

GiNaC

1.8.7

Generated by Doxygen 1.10.0

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	7
3.1 Class List	7
4 File Index	15
4.1 File List	15
5 Namespace Documentation	19
5.1 GiNaC Namespace Reference	19
5.1.1 Typedef Documentation	58
5.1.1.1 archive_node_id	58
5.1.1.2 archive_atom	58
5.1.1.3 synthesize_func	58
5.1.1.4 unarchive_map_t	58
5.1.1.5 exvector	58
5.1.1.6 exset	58
5.1.1.7 exmap	59
5.1.1.8 evalffunctype	59
5.1.1.9 FUNCP_1P	59
5.1.1.10 FUNCP_2P	59
5.1.1.11 FUNCP_CUBA	59
5.1.1.12 epvector	59
5.1.1.13 epp	59
5.1.1.14 exprseq	59
5.1.1.15 paramset	60
5.1.1.16 eval_funcp	60
5.1.1.17 evalf_funcp	60
5.1.1.18 conjugate_funcp	60
5.1.1.19 real_part_funcp	60
5.1.1.20 imag_part_funcp	60
5.1.1.21 expand_funcp	60
5.1.1.22 derivative_funcp	60
5.1.1.23 expl_derivative_funcp	60
5.1.1.24 power_funcp	60
5.1.1.25 series_funcp	61
5.1.1.26 print_funcp	61
5.1.1.27 info_funcp	61
5.1.1.28 eval_funcp_1	61
5.1.1.29 evalf_funcp_1	61

5.1.1.30 conjugate_funcp_1	61
5.1.1.31 real_part_funcp_1	61
5.1.1.32 imag_part_funcp_1	61
5.1.1.33 expand_funcp_1	61
5.1.1.34 derivative_funcp_1	61
5.1.1.35 expl_derivative_funcp_1	62
5.1.1.36 power_funcp_1	62
5.1.1.37 series_funcp_1	62
5.1.1.38 print_funcp_1	62
5.1.1.39 info_funcp_1	62
5.1.1.40 eval_funcp_2	62
5.1.1.41 evalf_funcp_2	62
5.1.1.42 conjugate_funcp_2	62
5.1.1.43 real_part_funcp_2	62
5.1.1.44 imag_part_funcp_2	62
5.1.1.45 expand_funcp_2	63
5.1.1.46 derivative_funcp_2	63
5.1.1.47 expl_derivative_funcp_2	63
5.1.1.48 power_funcp_2	63
5.1.1.49 series_funcp_2	63
5.1.1.50 print_funcp_2	63
5.1.1.51 info_funcp_2	63
5.1.1.52 eval_funcp_3	63
5.1.1.53 evalf_funcp_3	63
5.1.1.54 conjugate_funcp_3	63
5.1.1.55 real_part_funcp_3	64
5.1.1.56 imag_part_funcp_3	64
5.1.1.57 expand_funcp_3	64
5.1.1.58 derivative_funcp_3	64
5.1.1.59 expl_derivative_funcp_3	64
5.1.1.60 power_funcp_3	64
5.1.1.61 series_funcp_3	64
5.1.1.62 print_funcp_3	64
5.1.1.63 info_funcp_3	64
5.1.1.64 eval_funcp_4	64
5.1.1.65 evalf_funcp_4	65
5.1.1.66 conjugate_funcp_4	65
5.1.1.67 real_part_funcp_4	65
5.1.1.68 imag_part_funcp_4	65
5.1.1.69 expand_funcp_4	65
5.1.1.70 derivative_funcp_4	65
5.1.1.71 expl_derivative_funcp_4	65

5.1.1.72 power_funcp_4	65
5.1.1.73 series_funcp_4	65
5.1.1.74 print_funcp_4	65
5.1.1.75 info_funcp_4	66
5.1.1.76 eval_funcp_5	66
5.1.1.77 evalf_funcp_5	66
5.1.1.78 conjugate_funcp_5	66
5.1.1.79 real_part_funcp_5	66
5.1.1.80 imag_part_funcp_5	66
5.1.1.81 expand_funcp_5	66
5.1.1.82 derivative_funcp_5	66
5.1.1.83 expl_derivative_funcp_5	66
5.1.1.84 power_funcp_5	67
5.1.1.85 series_funcp_5	67
5.1.1.86 print_funcp_5	67
5.1.1.87 info_funcp_5	67
5.1.1.88 eval_funcp_6	67
5.1.1.89 evalf_funcp_6	67
5.1.1.90 conjugate_funcp_6	67
5.1.1.91 real_part_funcp_6	67
5.1.1.92 imag_part_funcp_6	67
5.1.1.93 expand_funcp_6	68
5.1.1.94 derivative_funcp_6	68
5.1.1.95 expl_derivative_funcp_6	68
5.1.1.96 power_funcp_6	68
5.1.1.97 series_funcp_6	68
5.1.1.98 print_funcp_6	68
5.1.1.99 info_funcp_6	68
5.1.1.100 eval_funcp_7	68
5.1.1.101 evalf_funcp_7	68
5.1.1.102 conjugate_funcp_7	69
5.1.1.103 real_part_funcp_7	69
5.1.1.104 imag_part_funcp_7	69
5.1.1.105 expand_funcp_7	69
5.1.1.106 derivative_funcp_7	69
5.1.1.107 expl_derivative_funcp_7	69
5.1.1.108 power_funcp_7	69
5.1.1.109 series_funcp_7	69
5.1.1.110 print_funcp_7	69
5.1.1.111 info_funcp_7	70
5.1.1.112 eval_funcp_8	70
5.1.1.113 evalf_funcp_8	70

5.1.1.114 conjugate_funcp_8	70
5.1.1.115 real_part_funcp_8	70
5.1.1.116 imag_part_funcp_8	70
5.1.1.117 expand_funcp_8	70
5.1.1.118 derivative_funcp_8	70
5.1.1.119 expl_derivative_funcp_8	70
5.1.1.120 power_funcp_8	71
5.1.1.121 series_funcp_8	71
5.1.1.122 print_funcp_8	71
5.1.1.123 info_funcp_8	71
5.1.1.124 eval_funcp_9	71
5.1.1.125 evalf_funcp_9	71
5.1.1.126 conjugate_funcp_9	71
5.1.1.127 real_part_funcp_9	71
5.1.1.128 imag_part_funcp_9	71
5.1.1.129 expand_funcp_9	72
5.1.1.130 derivative_funcp_9	72
5.1.1.131 expl_derivative_funcp_9	72
5.1.1.132 power_funcp_9	72
5.1.1.133 series_funcp_9	72
5.1.1.134 print_funcp_9	72
5.1.1.135 info_funcp_9	72
5.1.1.136 eval_funcp_10	72
5.1.1.137 evalf_funcp_10	72
5.1.1.138 conjugate_funcp_10	73
5.1.1.139 real_part_funcp_10	73
5.1.1.140 imag_part_funcp_10	73
5.1.1.141 expand_funcp_10	73
5.1.1.142 derivative_funcp_10	73
5.1.1.143 expl_derivative_funcp_10	73
5.1.1.144 power_funcp_10	73
5.1.1.145 series_funcp_10	73
5.1.1.146 print_funcp_10	73
5.1.1.147 info_funcp_10	74
5.1.1.148 eval_funcp_11	74
5.1.1.149 evalf_funcp_11	74
5.1.1.150 conjugate_funcp_11	74
5.1.1.151 real_part_funcp_11	74
5.1.1.152 imag_part_funcp_11	74
5.1.1.153 expand_funcp_11	74
5.1.1.154 derivative_funcp_11	74
5.1.1.155 expl_derivative_funcp_11	74

5.1.1.156 power_funcp_11	75
5.1.1.157 series_funcp_11	75
5.1.1.158 print_funcp_11	75
5.1.1.159 info_funcp_11	75
5.1.1.160 eval_funcp_12	75
5.1.1.161 evalf_funcp_12	75
5.1.1.162 conjugate_funcp_12	75
5.1.1.163 real_part_funcp_12	75
5.1.1.164 imag_part_funcp_12	76
5.1.1.165 expand_funcp_12	76
5.1.1.166 derivative_funcp_12	76
5.1.1.167 expl_derivative_funcp_12	76
5.1.1.168 power_funcp_12	76
5.1.1.169 series_funcp_12	76
5.1.1.170 print_funcp_12	76
5.1.1.171 info_funcp_12	76
5.1.1.172 eval_funcp_13	77
5.1.1.173 evalf_funcp_13	77
5.1.1.174 conjugate_funcp_13	77
5.1.1.175 real_part_funcp_13	77
5.1.1.176 imag_part_funcp_13	77
5.1.1.177 expand_funcp_13	77
5.1.1.178 derivative_funcp_13	77
5.1.1.179 expl_derivative_funcp_13	77
5.1.1.180 power_funcp_13	78
5.1.1.181 series_funcp_13	78
5.1.1.182 print_funcp_13	78
5.1.1.183 info_funcp_13	78
5.1.1.184 eval_funcp_14	78
5.1.1.185 evalf_funcp_14	78
5.1.1.186 conjugate_funcp_14	78
5.1.1.187 real_part_funcp_14	78
5.1.1.188 imag_part_funcp_14	79
5.1.1.189 expand_funcp_14	79
5.1.1.190 derivative_funcp_14	79
5.1.1.191 expl_derivative_funcp_14	79
5.1.1.192 power_funcp_14	79
5.1.1.193 series_funcp_14	79
5.1.1.194 print_funcp_14	79
5.1.1.195 info_funcp_14	79
5.1.1.196 eval_funcp_exvector	80
5.1.1.197 evalf_funcp_exvector	80

5.1.1.198 conjugate_funcp_exvector	80
5.1.1.199 real_part_funcp_exvector	80
5.1.1.200 imag_part_funcp_exvector	80
5.1.1.201 expand_funcp_exvector	80
5.1.1.202 derivative_funcp_exvector	80
5.1.1.203 expl_derivative_funcp_exvector	80
5.1.1.204 power_funcp_exvector	80
5.1.1.205 series_funcp_exvector	80
5.1.1.206 print_funcp_exvector	81
5.1.1.207 info_funcp_exvector	81
5.1.1.208 exhashmap	81
5.1.1.209 spmap	81
5.1.1.210 lookup_map	81
5.1.1.211 lst	81
5.1.1.212 uintvector	81
5.1.1.213 unsignedvector	81
5.1.1.214 exvectorvector	81
5.1.1.215 sym_desc_vec	81
5.1.1.216 digits_changed_callback	82
5.1.1.217 print_context_class_info	82
5.1.1.218 registered_class_info	82
5.1.2 Enumeration Type Documentation	82
5.1.2.1 anonymous enum	82
5.1.3 Function Documentation	82
5.1.3.1 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [1/33]	82
5.1.3.2 GINAC_BIND_UNARCHIVER() [1/49]	82
5.1.3.3 GINAC_DECLARE_UNARCHIVER() [1/51]	83
5.1.3.4 write_unsigned()	83
5.1.3.5 read_unsigned()	83
5.1.3.6 operator<<() [1/16]	83
5.1.3.7 operator<<() [2/16]	83
5.1.3.8 operator>>() [1/3]	83
5.1.3.9 operator>>() [2/3]	84
5.1.3.10 find_factory_fcn()	84
5.1.3.11 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [2/33]	84
5.1.3.12 is_a() [1/3]	84
5.1.3.13 is_exactly_a() [1/2]	84
5.1.3.14 dynallocate() [1/2]	85
5.1.3.15 dynallocate() [2/2]	85
5.1.3.16 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [3/33]	85
5.1.3.17 print_func< print_dflt >() [1/3]	85
5.1.3.18 GINAC_BIND_UNARCHIVER() [2/49]	85

5.1.3.19 GINAC_BIND_UNARCHIVER() [3/49]	85
5.1.3.20 GINAC_BIND_UNARCHIVER() [4/49]	86
5.1.3.21 GINAC_BIND_UNARCHIVER() [5/49]	86
5.1.3.22 GINAC_BIND_UNARCHIVER() [6/49]	86
5.1.3.23 GINAC_BIND_UNARCHIVER() [7/49]	86
5.1.3.24 GINAC_BIND_UNARCHIVER() [8/49]	86
5.1.3.25 is_dirac_slash()	86
5.1.3.26 base_and_index()	86
5.1.3.27 dirac_ONE()	86
5.1.3.28 get_dim_uint()	87
5.1.3.29 clifford_unit()	87
5.1.3.30 dirac_gamma()	87
5.1.3.31 dirac_gamma5()	88
5.1.3.32 dirac_gammaL()	88
5.1.3.33 dirac_gammaR()	88
5.1.3.34 dirac_slash()	89
5.1.3.35 get_representation_label() [1/2]	89
5.1.3.36 trace_string()	89
5.1.3.37 dirac_trace() [1/3]	89
5.1.3.38 dirac_trace() [2/3]	90
5.1.3.39 dirac_trace() [3/3]	90
5.1.3.40 canonicalize_clifford()	90
5.1.3.41 clifford_star_bar()	91
5.1.3.42 clifford_prime()	91
5.1.3.43 remove_dirac_ONE()	91
5.1.3.44 clifford_max_label()	91
5.1.3.45 clifford_norm()	92
5.1.3.46 clifford_inverse()	92
5.1.3.47 lst_to_clifford() [1/2]	92
5.1.3.48 lst_to_clifford() [2/2]	92
5.1.3.49 get_clifford_comp()	93
5.1.3.50 clifford_to_lst()	93
5.1.3.51 clifford_moebius_map() [1/2]	93
5.1.3.52 clifford_moebius_map() [2/2]	94
5.1.3.53 GINAC_DECLARE_UNARCHIVER() [2/51]	94
5.1.3.54 GINAC_DECLARE_UNARCHIVER() [3/51]	94
5.1.3.55 GINAC_DECLARE_UNARCHIVER() [4/51]	95
5.1.3.56 GINAC_DECLARE_UNARCHIVER() [5/51]	95
5.1.3.57 GINAC_DECLARE_UNARCHIVER() [6/51]	95
5.1.3.58 GINAC_DECLARE_UNARCHIVER() [7/51]	95
5.1.3.59 GINAC_DECLARE_UNARCHIVER() [8/51]	95
5.1.3.60 is_clifford_tinfo()	95

5.1.3.61 clifford_bar()	95
5.1.3.62 clifford_star()	96
5.1.3.63 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [4/33]	96
5.1.3.64 print_func< print_dflt >() [2/3]	96
5.1.3.65 GINAC_BIND_UNARCHIVER() [9/49]	96
5.1.3.66 GINAC_BIND_UNARCHIVER() [10/49]	96
5.1.3.67 GINAC_BIND_UNARCHIVER() [11/49]	96
5.1.3.68 GINAC_BIND_UNARCHIVER() [12/49]	96
5.1.3.69 GINAC_BIND_UNARCHIVER() [13/49]	97
5.1.3.70 permute_free_index_to_front()	97
5.1.3.71 color_ONE()	97
5.1.3.72 color_T()	98
5.1.3.73 color_f()	98
5.1.3.74 color_d()	98
5.1.3.75 color_h()	99
5.1.3.76 is_color_tinfo()	99
5.1.3.77 get_representation_label() [2/2]	99
5.1.3.78 color_trace() [1/3]	99
5.1.3.79 color_trace() [2/3]	100
5.1.3.80 color_trace() [3/3]	100
5.1.3.81 GINAC_DECLARE_UNARCHIVER() [9/51]	100
5.1.3.82 GINAC_DECLARE_UNARCHIVER() [10/51]	100
5.1.3.83 GINAC_DECLARE_UNARCHIVER() [11/51]	101
5.1.3.84 GINAC_DECLARE_UNARCHIVER() [12/51]	101
5.1.3.85 GINAC_DECLARE_UNARCHIVER() [13/51]	101
5.1.3.86 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [5/33]	101
5.1.3.87 GINAC_BIND_UNARCHIVER() [14/49]	101
5.1.3.88 GINAC_DECLARE_UNARCHIVER() [14/51]	101
5.1.3.89 crc32()	101
5.1.3.90 are_ex_trivially_equal()	102
5.1.3.91 operator<<() [3/16]	102
5.1.3.92 operator<<() [4/16]	102
5.1.3.93 operator<<() [5/16]	102
5.1.3.94 nops() [1/2]	103
5.1.3.95 expand() [1/2]	103
5.1.3.96 conjugate()	103
5.1.3.97 real_part()	103
5.1.3.98 imag_part()	103
5.1.3.99 has()	104
5.1.3.100 find()	104
5.1.3.101 is_polynomial()	104
5.1.3.102 degree()	104

5.1.3.103 ldegree()	104
5.1.3.104 coeff()	105
5.1.3.105 numer() [1/2]	105
5.1.3.106 denom() [1/2]	105
5.1.3.107 numer_denom()	105
5.1.3.108 normal()	105
5.1.3.109 to_rational()	106
5.1.3.110 to_polynomial()	106
5.1.3.111 collect()	106
5.1.3.112 eval()	106
5.1.3.113 evalf() [1/2]	106
5.1.3.114 evalm()	106
5.1.3.115 eval_integ()	107
5.1.3.116 diff()	107
5.1.3.117 series()	107
5.1.3.118 match()	107
5.1.3.119 simplify_indexed() [1/3]	107
5.1.3.120 simplify_indexed() [2/3]	108
5.1.3.121 symmetrize() [1/4]	108
5.1.3.122 symmetrize() [2/4]	108
5.1.3.123 antisymmetrize() [1/4]	108
5.1.3.124 antisymmetrize() [2/4]	108
5.1.3.125 symmetrize_cyclic() [1/4]	108
5.1.3.126 symmetrize_cyclic() [2/4]	109
5.1.3.127 op()	109
5.1.3.128 lhs()	109
5.1.3.129 rhs()	109
5.1.3.130 is_zero() [1/2]	109
5.1.3.131 swap() [1/2]	110
5.1.3.132 subs() [1/3]	110
5.1.3.133 subs() [2/3]	110
5.1.3.134 subs() [3/3]	110
5.1.3.135 is_a() [2/3]	110
5.1.3.136 is_exactly_a() [2/2]	111
5.1.3.137 ex_to()	111
5.1.3.138 compile_ex() [1/3]	111
5.1.3.139 compile_ex() [2/3]	112
5.1.3.140 compile_ex() [3/3]	112
5.1.3.141 link_ex() [1/3]	112
5.1.3.142 link_ex() [2/3]	113
5.1.3.143 link_ex() [3/3]	113
5.1.3.144 unlink_ex()	113

5.1.3.145 swap() [2/2]	114
5.1.3.146 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [6/33]	114
5.1.3.147 conjugateepvector()	114
5.1.3.148 factor()	114
5.1.3.149 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [7/33]	115
5.1.3.150 GINAC_DECLARE_UNARCHIVER() [15/51]	115
5.1.3.151 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [8/33]	115
5.1.3.152 GINAC_BIND_UNARCHIVER() [15/49]	115
5.1.3.153 GINAC_DECLARE_UNARCHIVER() [16/51]	115
5.1.3.154 GINAC_BIND_UNARCHIVER() [16/49]	116
5.1.3.155 GINAC_DECLARE_UNARCHIVER() [17/51]	116
5.1.3.156 is_the_function()	116
5.1.3.157 make_hash_seed()	116
5.1.3.158 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [9/33]	116
5.1.3.159 print_func< print_context >()	116
5.1.3.160 GINAC_BIND_UNARCHIVER() [17/49]	117
5.1.3.161 GINAC_BIND_UNARCHIVER() [18/49]	117
5.1.3.162 GINAC_BIND_UNARCHIVER() [19/49]	117
5.1.3.163 is_dummy_pair() [1/2]	117
5.1.3.164 is_dummy_pair() [2/2]	117
5.1.3.165 find_free_and_dummy() [1/2]	117
5.1.3.166 minimal_dim()	118
5.1.3.167 GINAC_DECLARE_UNARCHIVER() [18/51]	118
5.1.3.168 GINAC_DECLARE_UNARCHIVER() [19/51]	118
5.1.3.169 GINAC_DECLARE_UNARCHIVER() [20/51]	118
5.1.3.170 find_free_and_dummy() [2/2]	118
5.1.3.171 find_dummy_indices()	119
5.1.3.172 count_dummy_indices()	119
5.1.3.173 count_free_indices()	119
5.1.3.174 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [10/33]	119
5.1.3.175 GINAC_BIND_UNARCHIVER() [20/49]	120
5.1.3.176 indices_consistent()	120
5.1.3.177 number_of_type()	120
5.1.3.178 rename_dummy_indices()	120
5.1.3.179 find_variant_indices()	120
5.1.3.180 reposition_dummy_indices()	121
5.1.3.181 product_to_exvector()	121
5.1.3.182 idx_symmetrization()	121
5.1.3.183 simplify_indexed() [3/3]	121
5.1.3.184 simplify_indexed_product()	122
5.1.3.185 hasindex()	122
5.1.3.186 get_all_dummy_indices_safely()	122

5.1.3.187 <code>get_all_dummy_indices()</code>	122
5.1.3.188 <code>rename_dummy_indices_uniquely()</code> [1/4]	123
5.1.3.189 <code>rename_dummy_indices_uniquely()</code> [2/4]	123
5.1.3.190 <code>rename_dummy_indices_uniquely()</code> [3/4]	123
5.1.3.191 <code>rename_dummy_indices_uniquely()</code> [4/4]	123
5.1.3.192 <code>expand_dummy_sum()</code>	123
5.1.3.193 <code>GINAC_DECLARE_UNARCHIVER()</code> [21/51]	124
5.1.3.194 <code>conjugate_evalf()</code>	124
5.1.3.195 <code>conjugate_eval()</code>	124
5.1.3.196 <code>conjugate_print_latex()</code>	124
5.1.3.197 <code>conjugate_conjugate()</code>	124
5.1.3.198 <code>conjugate_expl_derivative()</code>	124
5.1.3.199 <code>conjugate_real_part()</code>	125
5.1.3.200 <code>conjugate_imag_part()</code>	125
5.1.3.201 <code>func_arg_info()</code>	125
5.1.3.202 <code>conjugate_info()</code>	125
5.1.3.203 <code>REGISTER_FUNCTION()</code> [1/36]	125
5.1.3.204 <code>real_part_evalf()</code>	125
5.1.3.205 <code>real_part_eval()</code>	126
5.1.3.206 <code>real_part_print_latex()</code>	126
5.1.3.207 <code>real_part_conjugate()</code>	126
5.1.3.208 <code>real_part_real_part()</code>	126
5.1.3.209 <code>real_part_imag_part()</code>	126
5.1.3.210 <code>real_part_expl_derivative()</code>	126
5.1.3.211 <code>REGISTER_FUNCTION()</code> [2/36]	126
5.1.3.212 <code>imag_part_evalf()</code>	127
5.1.3.213 <code>imag_part_eval()</code>	127
5.1.3.214 <code>imag_part_print_latex()</code>	127
5.1.3.215 <code>imag_part_conjugate()</code>	127
5.1.3.216 <code>imag_part_real_part()</code>	127
5.1.3.217 <code>imag_part_imag_part()</code>	127
5.1.3.218 <code>imag_part_expl_derivative()</code>	127
5.1.3.219 <code>REGISTER_FUNCTION()</code> [3/36]	128
5.1.3.220 <code>abs_evalf()</code>	128
5.1.3.221 <code>abs_eval()</code>	128
5.1.3.222 <code>abs_expand()</code>	128
5.1.3.223 <code>abs_expl_derivative()</code>	128
5.1.3.224 <code>abs_print_latex()</code>	128
5.1.3.225 <code>abs_print_csrc_float()</code>	129
5.1.3.226 <code>abs_conjugate()</code>	129
5.1.3.227 <code>abs_real_part()</code>	129
5.1.3.228 <code>abs_imag_part()</code>	129

5.1.3.229 <code>abs_power()</code>	129
5.1.3.230 <code>abs_info()</code>	129
5.1.3.231 <code>REGISTER_FUNCTION()</code> [4/36]	130
5.1.3.232 <code>step_evalf()</code>	130
5.1.3.233 <code>step_eval()</code>	130
5.1.3.234 <code>step_series()</code>	130
5.1.3.235 <code>step_conjugate()</code>	130
5.1.3.236 <code>step_real_part()</code>	130
5.1.3.237 <code>step_imag_part()</code>	131
5.1.3.238 <code>REGISTER_FUNCTION()</code> [5/36]	131
5.1.3.239 <code>csgn_evalf()</code>	131
5.1.3.240 <code>csgn_eval()</code>	131
5.1.3.241 <code>csgn_series()</code>	131
5.1.3.242 <code>csgn_conjugate()</code>	131
5.1.3.243 <code>csgn_real_part()</code>	132
5.1.3.244 <code>csgn_imag_part()</code>	132
5.1.3.245 <code>csgn_power()</code>	132
5.1.3.246 <code>REGISTER_FUNCTION()</code> [6/36]	132
5.1.3.247 <code>eta_evalf()</code>	132
5.1.3.248 <code>eta_eval()</code>	132
5.1.3.249 <code>eta_series()</code>	133
5.1.3.250 <code>eta_conjugate()</code>	133
5.1.3.251 <code>eta_real_part()</code>	133
5.1.3.252 <code>eta_imag_part()</code>	133
5.1.3.253 <code>REGISTER_FUNCTION()</code> [7/36]	133
5.1.3.254 <code>Li2_evalf()</code>	133
5.1.3.255 <code>Li2_eval()</code>	134
5.1.3.256 <code>Li2_deriv()</code>	134
5.1.3.257 <code>Li2_series()</code> [1/2]	134
5.1.3.258 <code>Li2_conjugate()</code>	134
5.1.3.259 <code>REGISTER_FUNCTION()</code> [8/36]	134
5.1.3.260 <code>Li3_eval()</code>	135
5.1.3.261 <code>REGISTER_FUNCTION()</code> [9/36]	135
5.1.3.262 <code>zetaderiv_eval()</code>	135
5.1.3.263 <code>zetaderiv_deriv()</code>	135
5.1.3.264 <code>REGISTER_FUNCTION()</code> [10/36]	135
5.1.3.265 <code>factorial_evalf()</code>	135
5.1.3.266 <code>factorial_eval()</code>	135
5.1.3.267 <code>factorial_print_dflit_latex()</code>	136
5.1.3.268 <code>factorial_conjugate()</code>	136
5.1.3.269 <code>factorial_real_part()</code>	136
5.1.3.270 <code>factorial_imag_part()</code>	136

5.1.3.271 REGISTER_FUNCTION() [11/36]	136
5.1.3.272 binomial_evalf()	136
5.1.3.273 binomial_sym()	137
5.1.3.274 binomial_eval()	137
5.1.3.275 binomial_conjugate()	137
5.1.3.276 binomial_real_part()	137
5.1.3.277 binomial_imag_part()	137
5.1.3.278 REGISTER_FUNCTION() [12/36]	137
5.1.3.279 Order_eval()	138
5.1.3.280 Order_series()	138
5.1.3.281 Order_conjugate()	138
5.1.3.282 Order_real_part()	138
5.1.3.283 Order_imag_part()	138
5.1.3.284 Order_power()	138
5.1.3.285 Order_expl_derivative()	139
5.1.3.286 REGISTER_FUNCTION() [13/36]	139
5.1.3.287 lsolve()	139
5.1.3.288 fsolve()	139
5.1.3.289 zeta() [1/3]	140
5.1.3.290 zeta() [2/3]	140
5.1.3.291 is_the_function< zeta_SERIAL >()	140
5.1.3.292 G() [1/2]	140
5.1.3.293 G() [2/2]	141
5.1.3.294 is_the_function< G_SERIAL >()	141
5.1.3.295 psi() [1/4]	141
5.1.3.296 psi() [2/4]	141
5.1.3.297 is_the_function< psi_SERIAL >()	141
5.1.3.298 iterated_integral() [1/2]	142
5.1.3.299 iterated_integral() [2/2]	142
5.1.3.300 is_the_function< iterated_integral_SERIAL >()	142
5.1.3.301 is_order_function()	142
5.1.3.302 convert_H_to_Li()	142
5.1.3.303 EllipticK_evalf()	143
5.1.3.304 EllipticK_eval()	143
5.1.3.305 EllipticK_deriv()	143
5.1.3.306 EllipticK_series()	143
5.1.3.307 EllipticK_print_latex()	143
5.1.3.308 REGISTER_FUNCTION() [14/36]	143
5.1.3.309 EllipticE_evalf()	144
5.1.3.310 EllipticE_eval()	144
5.1.3.311 EllipticE_deriv()	144
5.1.3.312 EllipticE_series()	144

5.1.3.313 EllipticE_print_latex()	144
5.1.3.314 REGISTER_FUNCTION() [15/36]	144
5.1.3.315 iterated_integral_evalf_impl()	145
5.1.3.316 iterated_integral2_evalf()	145
5.1.3.317 iterated_integral3_evalf()	145
5.1.3.318 iterated_integral2_eval()	145
5.1.3.319 iterated_integral3_eval()	145
5.1.3.320 lgamma_evalf()	146
5.1.3.321 lgamma_eval()	146
5.1.3.322 lgamma_deriv()	146
5.1.3.323 lgamma_series()	146
5.1.3.324 lgamma_conjugate()	146
5.1.3.325 REGISTER_FUNCTION() [16/36]	147
5.1.3.326 tgamma_evalf()	147
5.1.3.327 tgamma_eval()	147
5.1.3.328 tgamma_deriv()	147
5.1.3.329 tgamma_series()	147
5.1.3.330 tgamma_conjugate()	148
5.1.3.331 REGISTER_FUNCTION() [17/36]	148
5.1.3.332 beta_evalf()	148
5.1.3.333 beta_eval()	148
5.1.3.334 beta_deriv()	148
5.1.3.335 beta_series()	148
5.1.3.336 REGISTER_FUNCTION() [18/36]	149
5.1.3.337 psi1_evalf()	149
5.1.3.338 psi1_eval()	149
5.1.3.339 psi1_deriv()	149
5.1.3.340 psi1_series()	149
5.1.3.341 psi2_evalf()	150
5.1.3.342 psi2_eval()	150
5.1.3.343 psi2_deriv()	150
5.1.3.344 psi2_series()	150
5.1.3.345 G2_evalf()	150
5.1.3.346 G2_eval()	151
5.1.3.347 G3_evalf()	151
5.1.3.348 G3_eval()	151
5.1.3.349 Li_evalf()	151
5.1.3.350 Li_eval()	151
5.1.3.351 Li_series()	152
5.1.3.352 Li_deriv()	152
5.1.3.353 Li_print_latex()	152
5.1.3.354 REGISTER_FUNCTION() [19/36]	152

5.1.3.355 S_evalf()	152
5.1.3.356 S_eval()	153
5.1.3.357 S_series()	153
5.1.3.358 S_deriv()	153
5.1.3.359 S_print_latex()	153
5.1.3.360 REGISTER_FUNCTION() [20/36]	153
5.1.3.361 H_evalf()	154
5.1.3.362 H_eval()	154
5.1.3.363 H_series()	154
5.1.3.364 H_deriv()	154
5.1.3.365 H_print_latex()	154
5.1.3.366 REGISTER_FUNCTION() [21/36]	155
5.1.3.367 zeta1_evalf()	155
5.1.3.368 zeta1_eval()	155
5.1.3.369 zeta1_deriv()	155
5.1.3.370 zeta1_print_latex()	155
5.1.3.371 zeta2_evalf()	155
5.1.3.372 zeta2_eval()	156
5.1.3.373 zeta2_deriv()	156
5.1.3.374 zeta2_print_latex()	156
5.1.3.375 exp_evalf()	156
5.1.3.376 exp_eval()	156
5.1.3.377 exp_expand()	156
5.1.3.378 exp_deriv()	157
5.1.3.379 exp_real_part()	157
5.1.3.380 exp_imag_part()	157
5.1.3.381 exp_conjugate()	157
5.1.3.382 exp_power()	157
5.1.3.383 exp_info()	157
5.1.3.384 REGISTER_FUNCTION() [22/36]	158
5.1.3.385 log_evalf()	158
5.1.3.386 log_eval()	158
5.1.3.387 log_deriv()	158
5.1.3.388 log_series()	158
5.1.3.389 log_real_part()	159
5.1.3.390 log_imag_part()	159
5.1.3.391 log_expand()	159
5.1.3.392 log_conjugate()	159
5.1.3.393 log_info()	159
5.1.3.394 REGISTER_FUNCTION() [23/36]	159
5.1.3.395 sin_evalf()	160
5.1.3.396 sin_eval()	160

5.1.3.397 sin_deriv()	160
5.1.3.398 sin_real_part()	160
5.1.3.399 sin_imag_part()	160
5.1.3.400 sin_conjugate()	160
5.1.3.401 trig_info()	161
5.1.3.402 REGISTER_FUNCTION() [24/36]	161
5.1.3.403 cos_evalf()	161
5.1.3.404 cos_eval()	161
5.1.3.405 cos_deriv()	161
5.1.3.406 cos_real_part()	161
5.1.3.407 cos_imag_part()	162
5.1.3.408 cos_conjugate()	162
5.1.3.409 REGISTER_FUNCTION() [25/36]	162
5.1.3.410 tan_evalf()	162
5.1.3.411 tan_eval()	162
5.1.3.412 tan_deriv()	162
5.1.3.413 tan_real_part()	163
5.1.3.414 tan_imag_part()	163
5.1.3.415 tan_series()	163
5.1.3.416 tan_conjugate()	163
5.1.3.417 REGISTER_FUNCTION() [26/36]	163
5.1.3.418 asin_evalf()	163
5.1.3.419 asin_eval()	164
5.1.3.420 asin_deriv()	164
5.1.3.421 asin_conjugate()	164
5.1.3.422 asin_info()	164
5.1.3.423 REGISTER_FUNCTION() [27/36]	164
5.1.3.424 acos_evalf()	164
5.1.3.425 acos_eval()	165
5.1.3.426 acos_deriv()	165
5.1.3.427 acos_conjugate()	165
5.1.3.428 REGISTER_FUNCTION() [28/36]	165
5.1.3.429 atan_evalf()	165
5.1.3.430 atan_eval()	165
5.1.3.431 atan_deriv()	166
5.1.3.432 atan_series()	166
5.1.3.433 atan_conjugate()	166
5.1.3.434 atan_info()	166
5.1.3.435 REGISTER_FUNCTION() [29/36]	166
5.1.3.436 atan2_evalf()	166
5.1.3.437 atan2_eval()	167
5.1.3.438 atan2_deriv()	167

5.1.3.439 atan2_info()	167
5.1.3.440 REGISTER_FUNCTION() [30/36]	167
5.1.3.441 sinh_evalf()	167
5.1.3.442 sinh_eval()	167
5.1.3.443 sinh_deriv()	168
5.1.3.444 sinh_real_part()	168
5.1.3.445 sinh_imag_part()	168
5.1.3.446 sinh_conjugate()	168
5.1.3.447 REGISTER_FUNCTION() [31/36]	168
5.1.3.448 cosh_evalf()	168
5.1.3.449 cosh_eval()	169
5.1.3.450 cosh_deriv()	169
5.1.3.451 cosh_real_part()	169
5.1.3.452 cosh_imag_part()	169
5.1.3.453 cosh_conjugate()	169
5.1.3.454 REGISTER_FUNCTION() [32/36]	169
5.1.3.455 tanh_evalf()	170
5.1.3.456 tanh_eval()	170
5.1.3.457 tanh_deriv()	170
5.1.3.458 tanh_series()	170
5.1.3.459 tanh_real_part()	170
5.1.3.460 tanh_imag_part()	170
5.1.3.461 tanh_conjugate()	171
5.1.3.462 REGISTER_FUNCTION() [33/36]	171
5.1.3.463 asinh_evalf()	171
5.1.3.464 asinh_eval()	171
5.1.3.465 asinh_deriv()	171
5.1.3.466 asinh_conjugate()	171
5.1.3.467 REGISTER_FUNCTION() [34/36]	172
5.1.3.468 acosh_evalf()	172
5.1.3.469 acosh_eval()	172
5.1.3.470 acosh_deriv()	172
5.1.3.471 acosh_conjugate()	172
5.1.3.472 REGISTER_FUNCTION() [35/36]	172
5.1.3.473 atanh_evalf()	172
5.1.3.474 atanh_eval()	173
5.1.3.475 atanh_deriv()	173
5.1.3.476 atanh_series()	173
5.1.3.477 atanh_conjugate()	173
5.1.3.478 REGISTER_FUNCTION() [36/36]	173
5.1.3.479 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [11/33]	173
5.1.3.480 subsvalue()	174

5.1.3.481 adaptivesimpson()	174
5.1.3.482 GINAC_BIND_UNARCHIVER() [21/49]	174
5.1.3.483 GINAC_DECLARE_UNARCHIVER() [22/51]	174
5.1.3.484 ifactor()	174
5.1.3.485 is_discriminant_of_quadratic_number_field()	175
5.1.3.486 kronecker_symbol()	175
5.1.3.487 primitive_dirichlet_character()	175
5.1.3.488 dirichlet_character()	176
5.1.3.489 generalised_Bernoulli_number()	176
5.1.3.490 Bernoulli_polynomial()	176
5.1.3.491 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [12/33]	176
5.1.3.492 GINAC_BIND_UNARCHIVER() [22/49]	177
5.1.3.493 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [13/33]	177
5.1.3.494 GINAC_BIND_UNARCHIVER() [23/49]	177
5.1.3.495 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [14/33]	177
5.1.3.496 GINAC_BIND_UNARCHIVER() [24/49]	177
5.1.3.497 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [15/33]	177
5.1.3.498 GINAC_BIND_UNARCHIVER() [25/49]	177
5.1.3.499 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [16/33]	178
5.1.3.500 GINAC_BIND_UNARCHIVER() [26/49]	178
5.1.3.501 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [17/33]	178
5.1.3.502 GINAC_BIND_UNARCHIVER() [27/49]	178
5.1.3.503 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [18/33]	178
5.1.3.504 GINAC_BIND_UNARCHIVER() [28/49]	178
5.1.3.505 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [19/33]	178
5.1.3.506 GINAC_BIND_UNARCHIVER() [29/49]	178
5.1.3.507 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [20/33]	179
5.1.3.508 GINAC_BIND_UNARCHIVER() [30/49]	179
5.1.3.509 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [21/33]	179
5.1.3.510 GINAC_BIND_UNARCHIVER() [31/49]	179
5.1.3.511 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [22/33]	179
5.1.3.512 GINAC_BIND_UNARCHIVER() [32/49]	179
5.1.3.513 GINAC_DECLARE_UNARCHIVER() [23/51]	179
5.1.3.514 GINAC_DECLARE_UNARCHIVER() [24/51]	179
5.1.3.515 GINAC_DECLARE_UNARCHIVER() [25/51]	180
5.1.3.516 GINAC_DECLARE_UNARCHIVER() [26/51]	180
5.1.3.517 GINAC_DECLARE_UNARCHIVER() [27/51]	180
5.1.3.518 GINAC_DECLARE_UNARCHIVER() [28/51]	180
5.1.3.519 GINAC_DECLARE_UNARCHIVER() [29/51]	180
5.1.3.520 GINAC_DECLARE_UNARCHIVER() [30/51]	180
5.1.3.521 GINAC_DECLARE_UNARCHIVER() [31/51]	180
5.1.3.522 GINAC_DECLARE_UNARCHIVER() [32/51]	180

5.1.3.523 GINAC_DECLARE_UNARCHIVER() [33/51]	180
5.1.3.524 GINAC_DECLARE_UNARCHIVER() [34/51]	181
5.1.3.525 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [23/33]	181
5.1.3.526 GINAC_BIND_UNARCHIVER() [33/49]	181
5.1.3.527 lst_to_matrix()	181
5.1.3.528 diag_matrix() [1/2]	181
5.1.3.529 diag_matrix() [2/2]	181
5.1.3.530 unit_matrix() [1/2]	182
5.1.3.531 symbolic_matrix() [1/2]	182
5.1.3.532 reduced_matrix()	182
5.1.3.533 sub_matrix()	182
5.1.3.534 GINAC_DECLARE_UNARCHIVER() [35/51]	183
5.1.3.535 nops() [2/2]	183
5.1.3.536 expand() [2/2]	183
5.1.3.537 evalf() [2/2]	183
5.1.3.538 rows()	183
5.1.3.539 cols()	183
5.1.3.540 transpose()	184
5.1.3.541 determinant()	184
5.1.3.542 trace()	184
5.1.3.543 charpoly()	184
5.1.3.544 inverse() [1/3]	184
5.1.3.545 inverse() [2/3]	184
5.1.3.546 rank() [1/2]	185
5.1.3.547 rank() [2/2]	185
5.1.3.548 unit_matrix() [2/2]	185
5.1.3.549 symbolic_matrix() [2/2]	185
5.1.3.550 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [24/33]	185
5.1.3.551 tryfactsubs()	186
5.1.3.552 algebraic_match_mul_with_mul()	186
5.1.3.553 GINAC_BIND_UNARCHIVER() [34/49]	186
5.1.3.554 GINAC_DECLARE_UNARCHIVER() [36/51]	186
5.1.3.555 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [25/33]	186
5.1.3.556 reeval_ncmul()	187
5.1.3.557 hold_ncmul()	187
5.1.3.558 GINAC_BIND_UNARCHIVER() [35/49]	187
5.1.3.559 GINAC_DECLARE_UNARCHIVER() [37/51]	187
5.1.3.560 get_first_symbol()	187
5.1.3.561 add_symbol()	188
5.1.3.562 collect_symbols()	188
5.1.3.563 get_symbol_stats()	188
5.1.3.564 lcmcoeff()	188

5.1.3.565 lcm_of_coefficients_denominators()	189
5.1.3.566 multiply_lcm()	189
5.1.3.567 quo()	189
5.1.3.568 rem()	190
5.1.3.569 decomp_rational()	190
5.1.3.570 prem()	191
5.1.3.571 sprem()	191
5.1.3.572 divide()	192
5.1.3.573 divide_in_z()	192
5.1.3.574 sr_gcd()	193
5.1.3.575 interpolate()	194
5.1.3.576 heur_gcd_z()	194
5.1.3.577 heur_gcd()	195
5.1.3.578 gcd_pf_pow()	195
5.1.3.579 gcd_pf_mul()	196
5.1.3.580 gcd() $[1/2]$	196
5.1.3.581 gcd_pf_pow_pow()	197
5.1.3.582 lcm() $[1/2]$	197
5.1.3.583 sqrfree_yun()	197
5.1.3.584 sqrfree()	198
5.1.3.585 sqrfree_parfrac()	198
5.1.3.586 replace_with_symbol() $[1/2]$	199
5.1.3.587 replace_with_symbol() $[2/2]$	200
5.1.3.588 frac_cancel()	200
5.1.3.589 find_common_factor()	200
5.1.3.590 collect_common_factors()	201
5.1.3.591 resultant()	201
5.1.3.592 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() $[26/33]$	201
5.1.3.593 make_real_float()	201
5.1.3.594 read_real_float()	202
5.1.3.595 GINAC_BIND_UNARCHIVER() $[36/49]$	202
5.1.3.596 write_real_float()	202
5.1.3.597 print_real_number()	202
5.1.3.598 print_integer_csrc()	203
5.1.3.599 print_real_csrc()	203
5.1.3.600 coerce()	203
5.1.3.601 coerce< int, cln::cl_I >()	203
5.1.3.602 coerce< unsigned int, cln::cl_I >()	204
5.1.3.603 print_real_cl_N()	204
5.1.3.604 exp()	204
5.1.3.605 log()	204
5.1.3.606 sin()	205

5.1.3.607 cos()	205
5.1.3.608 tan()	206
5.1.3.609 asin()	206
5.1.3.610 acos()	206
5.1.3.611 atan() ^[1/2]	206
5.1.3.612 atan() ^[2/2]	207
5.1.3.613 sinh()	207
5.1.3.614 cosh()	208
5.1.3.615 tanh()	208
5.1.3.616 asinh()	208
5.1.3.617 acosh()	209
5.1.3.618 atanh()	209
5.1.3.619 Li2_series() ^[2/2]	209
5.1.3.620 Li2_projection()	209
5.1.3.621 Li2_()	210
5.1.3.622 Li2()	210
5.1.3.623 zeta() ^[3/3]	210
5.1.3.624 guess_precision()	210
5.1.3.625 lgamma() ^[1/2]	211
5.1.3.626 lgamma() ^[2/2]	211
5.1.3.627 tgamma() ^[1/2]	211
5.1.3.628 tgamma() ^[2/2]	211
5.1.3.629 psi() ^[3/4]	211
5.1.3.630 psi() ^[4/4]	212
5.1.3.631 factorial()	212
5.1.3.632 doublefactorial()	212
5.1.3.633 binomial()	213
5.1.3.634 bernoulli()	213
5.1.3.635 fibonacci()	213
5.1.3.636 abs()	214
5.1.3.637 mod()	214
5.1.3.638 smod()	215
5.1.3.639 irem() ^[1/2]	215
5.1.3.640 irem() ^[2/2]	215
5.1.3.641 iquo() ^[1/2]	216
5.1.3.642 iquo() ^[2/2]	216
5.1.3.643 gcd() ^[2/2]	217
5.1.3.644 lcm() ^[2/2]	217
5.1.3.645 sqrt() ^[1/2]	217
5.1.3.646 isqrt()	218
5.1.3.647 PiEvalf()	218
5.1.3.648 EulerEvalf()	218

5.1.3.649 CatalanEvalf()	218
5.1.3.650 operator<<() [6/16]	218
5.1.3.651 GINAC_DECLARE_UNARCHIVER() [38/51]	219
5.1.3.652 pow() [1/3]	219
5.1.3.653 inverse() [3/3]	219
5.1.3.654 step()	219
5.1.3.655 csgn()	219
5.1.3.656 is_zero() [2/2]	220
5.1.3.657 is_positive()	220
5.1.3.658 is_negative()	220
5.1.3.659 is_integer()	220
5.1.3.660 is_pos_integer()	220
5.1.3.661 is_nonneg_integer()	220
5.1.3.662 is_even()	221
5.1.3.663 is_odd()	221
5.1.3.664 is_prime()	221
5.1.3.665 is_rational()	221
5.1.3.666 is_real()	221
5.1.3.667 is_cinteger()	221
5.1.3.668 is_crational()	222
5.1.3.669 to_int()	222
5.1.3.670 to_long()	222
5.1.3.671 to_double()	222
5.1.3.672 real()	222
5.1.3.673 imag()	222
5.1.3.674 numer() [2/2]	223
5.1.3.675 denom() [2/2]	223
5.1.3.676 exadd()	223
5.1.3.677 exmul()	223
5.1.3.678 exminus()	223
5.1.3.679 operator+() [1/4]	224
5.1.3.680 operator-() [1/4]	224
5.1.3.681 operator*() [1/2]	224
5.1.3.682 operator/() [1/2]	224
5.1.3.683 operator+() [2/4]	224
5.1.3.684 operator-() [2/4]	224
5.1.3.685 operator*() [2/2]	225
5.1.3.686 operator/() [2/2]	225
5.1.3.687 operator+=() [1/2]	225
5.1.3.688 operator-=() [1/2]	225
5.1.3.689 operator*=() [1/2]	225
5.1.3.690 operator/=() [1/2]	225

5.1.3.691 operator+=() [2/2]	226
5.1.3.692 operator-=() [2/2]	226
5.1.3.693 operator*=() [2/2]	226
5.1.3.694 operator/=() [2/2]	226
5.1.3.695 operator+() [3/4]	226
5.1.3.696 operator-() [3/4]	226
5.1.3.697 operator+() [4/4]	226
5.1.3.698 operator-() [4/4]	227
5.1.3.699 operator++() [1/4]	227
5.1.3.700 operator--() [1/4]	227
5.1.3.701 operator++() [2/4]	227
5.1.3.702 operator--() [2/4]	227
5.1.3.703 operator++() [3/4]	228
5.1.3.704 operator--() [3/4]	228
5.1.3.705 operator++() [4/4]	228
5.1.3.706 operator--() [4/4]	228
5.1.3.707 operator==()	228
5.1.3.708 operator!=()	229
5.1.3.709 operator<()	229
5.1.3.710 operator<=()	229
5.1.3.711 operator>()	229
5.1.3.712 operator>=()	229
5.1.3.713 my_ios_index()	229
5.1.3.714 my_ios_callback()	230
5.1.3.715 get_print_context()	230
5.1.3.716 set_print_context()	230
5.1.3.717 get_print_options()	230
5.1.3.718 set_print_options()	230
5.1.3.719 operator<<() [7/16]	231
5.1.3.720 operator>>() [3/3]	231
5.1.3.721 dflit()	231
5.1.3.722 latex()	231
5.1.3.723 python()	231
5.1.3.724 python_repr()	231
5.1.3.725 tree()	231
5.1.3.726 csrc()	232
5.1.3.727 csrc_float()	232
5.1.3.728 csrc_double()	232
5.1.3.729 csrc_cl_N()	232
5.1.3.730 index_dimensions()	232
5.1.3.731 no_index_dimensions()	232
5.1.3.732 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [27/33]	232

5.1.3.733 <code>print_sym_pow()</code>	233
5.1.3.734 <code>GINAC_BIND_UNARCHIVER()</code> [37/49]	233
5.1.3.735 <code>GINAC_DECLARE_UNARCHIVER()</code> [39/51]	233
5.1.3.736 <code>pow()</code> [2/3]	233
5.1.3.737 <code>pow()</code> [3/3]	233
5.1.3.738 <code>sqrt()</code> [2/2]	233
5.1.3.739 <code>is_a()</code> [3/3]	234
5.1.3.740 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [28/33]	234
5.1.3.741 <code>GINAC_BIND_UNARCHIVER()</code> [38/49]	234
5.1.3.742 <code>GINAC_DECLARE_UNARCHIVER()</code> [40/51]	234
5.1.3.743 <code>series_to_poly()</code>	234
5.1.3.744 <code>is_terminating()</code>	235
5.1.3.745 <code>make_return_type_t()</code>	235
5.1.3.746 <code>set_print_func()</code> [1/2]	235
5.1.3.747 <code>set_print_func()</code> [2/2]	235
5.1.3.748 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [29/33]	236
5.1.3.749 <code>GINAC_BIND_UNARCHIVER()</code> [39/49]	236
5.1.3.750 <code>print_operator()</code>	236
5.1.3.751 <code>GINAC_DECLARE_UNARCHIVER()</code> [41/51]	236
5.1.3.752 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [30/33]	236
5.1.3.753 <code>get_default_TeX_name()</code>	236
5.1.3.754 <code>GINAC_BIND_UNARCHIVER()</code> [40/49]	237
5.1.3.755 <code>GINAC_BIND_UNARCHIVER()</code> [41/49]	237
5.1.3.756 <code>GINAC_BIND_UNARCHIVER()</code> [42/49]	237
5.1.3.757 <code>GINAC_DECLARE_UNARCHIVER()</code> [42/51]	237
5.1.3.758 <code>GINAC_DECLARE_UNARCHIVER()</code> [43/51]	237
5.1.3.759 <code>GINAC_DECLARE_UNARCHIVER()</code> [44/51]	237
5.1.3.760 <code>GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()</code> [31/33]	237
5.1.3.761 <code>GINAC_BIND_UNARCHIVER()</code> [43/49]	237
5.1.3.762 <code>index0()</code>	238
5.1.3.763 <code>index1()</code>	238
5.1.3.764 <code>index2()</code>	238
5.1.3.765 <code>index3()</code>	238
5.1.3.766 <code>not_symmetric()</code>	238
5.1.3.767 <code>symmetric2()</code>	238
5.1.3.768 <code>symmetric3()</code>	238
5.1.3.769 <code>symmetric4()</code>	239
5.1.3.770 <code>antisymmetric2()</code>	239
5.1.3.771 <code>antisymmetric3()</code>	239
5.1.3.772 <code>antisymmetric4()</code>	239
5.1.3.773 <code>canonicalize()</code>	239
5.1.3.774 <code>symm()</code>	240

5.1.3.775 symmetrize() [3/4]	240
5.1.3.776 antisymmetrize() [3/4]	240
5.1.3.777 symmetrize_cyclic() [3/4]	240
5.1.3.778 GINAC_DECLARE_UNARCHIVER() [45/51]	240
5.1.3.779 sy_none() [1/4]	241
5.1.3.780 sy_none() [2/4]	241
5.1.3.781 sy_none() [3/4]	241
5.1.3.782 sy_none() [4/4]	241
5.1.3.783 sy_symm() [1/4]	241
5.1.3.784 sy_symm() [2/4]	241
5.1.3.785 sy_symm() [3/4]	242
5.1.3.786 sy_symm() [4/4]	242
5.1.3.787 sy_anti() [1/4]	242
5.1.3.788 sy_anti() [2/4]	242
5.1.3.789 sy_anti() [3/4]	242
5.1.3.790 sy_anti() [4/4]	242
5.1.3.791 sy_cycl() [1/4]	243
5.1.3.792 sy_cycl() [2/4]	243
5.1.3.793 sy_cycl() [3/4]	243
5.1.3.794 sy_cycl() [4/4]	243
5.1.3.795 symmetrize() [4/4]	243
5.1.3.796 antisymmetrize() [4/4]	243
5.1.3.797 symmetrize_cyclic() [4/4]	244
5.1.3.798 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [32/33]	244
5.1.3.799 print_func< print_dflt >() [3/3]	244
5.1.3.800 GINAC_BIND_UNARCHIVER() [44/49]	244
5.1.3.801 GINAC_BIND_UNARCHIVER() [45/49]	244
5.1.3.802 GINAC_BIND_UNARCHIVER() [46/49]	244
5.1.3.803 GINAC_BIND_UNARCHIVER() [47/49]	244
5.1.3.804 GINAC_BIND_UNARCHIVER() [48/49]	245
5.1.3.805 delta_tensor()	245
5.1.3.806 metric_tensor()	245
5.1.3.807 lorentz_g()	246
5.1.3.808 spinor_metric()	246
5.1.3.809 epsilon_tensor() [1/2]	247
5.1.3.810 epsilon_tensor() [2/2]	247
5.1.3.811 lorentz_eps()	247
5.1.3.812 GINAC_DECLARE_UNARCHIVER() [46/51]	248
5.1.3.813 GINAC_DECLARE_UNARCHIVER() [47/51]	248
5.1.3.814 GINAC_DECLARE_UNARCHIVER() [48/51]	248
5.1.3.815 GINAC_DECLARE_UNARCHIVER() [49/51]	248
5.1.3.816 GINAC_DECLARE_UNARCHIVER() [50/51]	248

5.1.3.817 log2()	249
5.1.3.818 multinomial_coefficient()	249
5.1.3.819 rotate_left()	249
5.1.3.820 compare_pointers()	249
5.1.3.821 golden_ratio_hash()	250
5.1.3.822 permutation_sign() [1/2]	250
5.1.3.823 permutation_sign() [2/2]	250
5.1.3.824 shaker_sort()	250
5.1.3.825 cyclic_permutation()	251
5.1.3.826 format_index_value() [1/2]	251
5.1.3.827 format_index_value() [2/2]	251
5.1.3.828 operator<<() [8/16]	251
5.1.3.829 operator<<() [9/16]	252
5.1.3.830 operator<<() [10/16]	252
5.1.3.831 operator<<() [11/16]	252
5.1.3.832 operator<<() [12/16]	252
5.1.3.833 operator<<() [13/16]	253
5.1.3.834 operator<<() [14/16]	253
5.1.3.835 operator<<() [15/16]	253
5.1.3.836 operator<<() [16/16]	253
5.1.3.837 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [33/33]	254
5.1.3.838 GINAC_BIND_UNARCHIVER() [49/49]	254
5.1.3.839 haswild()	254
5.1.3.840 GINAC_DECLARE_UNARCHIVER() [51/51]	254
5.1.3.841 wild()	254
5.1.4 Variable Documentation	254
5.1.4.1 unarch_table_instance	254
5.1.4.2 map_evalm	254
5.1.4.3 map_eval_integ	255
5.1.4.4 tensor	255
5.1.4.5 Pi	255
5.1.4.6 Euler	255
5.1.4.7 Catalan	255
5.1.4.8 crctab	256
5.1.4.9 library_initializer	256
5.1.4.10 _num0_bp	256
5.1.4.11 idx	256
5.1.4.12 force_include_tgamma	256
5.1.4.13 force_include_zeta1	256
5.1.4.14 GINAC_BIND_UNARCHIVER	256
5.1.4.15 I	257
5.1.4.16 Digits	257

5.1.4.17 next_print_context_id	257
5.1.4.18 version_major	257
5.1.4.19 version_minor	257
5.1.4.20 version_micro	257
5.1.4.21 _num_120_p	258
5.1.4.22 _ex_120	258
5.1.4.23 _num_60_p	258
5.1.4.24 _ex_60	258
5.1.4.25 _num_48_p	258
5.1.4.26 _ex_48	258
5.1.4.27 _num_30_p	258
5.1.4.28 _ex_30	258
5.1.4.29 _num_25_p	259
5.1.4.30 _ex_25	259
5.1.4.31 _num_24_p	259
5.1.4.32 _ex_24	259
5.1.4.33 _num_20_p	259
5.1.4.34 _ex_20	259
5.1.4.35 _num_18_p	259
5.1.4.36 _ex_18	259
5.1.4.37 _num_15_p	260
5.1.4.38 _ex_15	260
5.1.4.39 _num_12_p	260
5.1.4.40 _ex_12	260
5.1.4.41 _num_11_p	260
5.1.4.42 _ex_11	260
5.1.4.43 _num_10_p	260
5.1.4.44 _ex_10	260
5.1.4.45 _num_9_p	261
5.1.4.46 _ex_9	261
5.1.4.47 _num_8_p	261
5.1.4.48 _ex_8	261
5.1.4.49 _num_7_p	261
5.1.4.50 _ex_7	261
5.1.4.51 _num_6_p	261
5.1.4.52 _ex_6	261
5.1.4.53 _num_5_p	262
5.1.4.54 _ex_5	262
5.1.4.55 _num_4_p	262
5.1.4.56 _ex_4	262
5.1.4.57 _num_3_p	262
5.1.4.58 _ex_3	262

5.1.4.59 _num_2_p	262
5.1.4.60 _ex_2	263
5.1.4.61 _num_1_p	263
5.1.4.62 _ex_1	263
5.1.4.63 _num_1_2_p	263
5.1.4.64 _ex_1_2	263
5.1.4.65 _num_1_3_p	263
5.1.4.66 _ex_1_3	264
5.1.4.67 _num_1_4_p	264
5.1.4.68 _ex_1_4	264
5.1.4.69 _num0_p	264
5.1.4.70 _ex0	264
5.1.4.71 _num1_4_p	265
5.1.4.72 _ex1_4	265
5.1.4.73 _num1_3_p	265
5.1.4.74 _ex1_3	265
5.1.4.75 _num1_2_p	265
5.1.4.76 _ex1_2	265
5.1.4.77 _num1_p	265
5.1.4.78 _ex1	266
5.1.4.79 _num2_p	266
5.1.4.80 _ex2	266
5.1.4.81 _num3_p	266
5.1.4.82 _ex3	267
5.1.4.83 _num4_p	267
5.1.4.84 _ex4	267
5.1.4.85 _num5_p	267
5.1.4.86 _ex5	267
5.1.4.87 _num6_p	267
5.1.4.88 _ex6	267
5.1.4.89 _num7_p	268
5.1.4.90 _ex7	268
5.1.4.91 _num8_p	268
5.1.4.92 _ex8	268
5.1.4.93 _num9_p	268
5.1.4.94 _ex9	268
5.1.4.95 _num10_p	268
5.1.4.96 _ex10	269
5.1.4.97 _num11_p	269
5.1.4.98 _ex11	269
5.1.4.99 _num12_p	269
5.1.4.100 _ex12	269

5.1.4.101 _num15_p	269
5.1.4.102 _ex15	269
5.1.4.103 _num18_p	270
5.1.4.104 _ex18	270
5.1.4.105 _num20_p	270
5.1.4.106 _ex20	270
5.1.4.107 _num24_p	270
5.1.4.108 _ex24	270
5.1.4.109 _num25_p	270
5.1.4.110 _ex25	270
5.1.4.111 _num30_p	271
5.1.4.112 _ex30	271
5.1.4.113 _num48_p	271
5.1.4.114 _ex48	271
5.1.4.115 _num60_p	271
5.1.4.116 _ex60	271
5.1.4.117 _num120_p	271
5.1.4.118 _ex120	271
5.2 GiNaC::internal Namespace Reference	272
5.3 std Namespace Reference	272
5.3.1 Function Documentation	272
5.3.1.1 swap()	272
6 Class Documentation	273
6.1 GiNaC::internal::_iter_rep Struct Reference	273
6.1.1 Constructor & Destructor Documentation	274
6.1.1.1 _iter_rep()	274
6.1.2 Member Function Documentation	274
6.1.2.1 operator==()	274
6.1.2.2 operator!=()	274
6.1.3 Member Data Documentation	274
6.1.3.1 e	274
6.1.3.2 i	275
6.1.3.3 i_end	275
6.2 GiNaC::_numeric_digits Class Reference	275
6.2.1 Detailed Description	276
6.2.2 Constructor & Destructor Documentation	276
6.2.2.1 _numeric_digits()	276
6.2.3 Member Function Documentation	276
6.2.3.1 operator=()	276
6.2.3.2 operator long()	276
6.2.3.3 print()	276

6.2.3.4 add_callback()	277
6.2.4 Member Data Documentation	277
6.2.4.1 digits	277
6.2.4.2 too_late	277
6.2.4.3 callbacklist	277
6.3 GiNaC::add Class Reference	278
6.3.1 Detailed Description	284
6.3.2 Constructor & Destructor Documentation	284
6.3.2.1 add() [1/6]	284
6.3.2.2 add() [2/6]	284
6.3.2.3 add() [3/6]	285
6.3.2.4 add() [4/6]	285
6.3.2.5 add() [5/6]	285
6.3.2.6 add() [6/6]	285
6.3.3 Member Function Documentation	285
6.3.3.1 precedence()	285
6.3.3.2 info()	286
6.3.3.3 is_polynomial()	286
6.3.3.4 degree()	286
6.3.3.5 ldegree()	286
6.3.3.6 coeff()	287
6.3.3.7 eval()	287
6.3.3.8 evalm()	287
6.3.3.9 series()	288
6.3.3.10 normal()	288
6.3.3.11 integer_content()	288
6.3.3.12 smod()	288
6.3.3.13 max_coefficient()	289
6.3.3.14 conjugate()	289
6.3.3.15 real_part()	289
6.3.3.16 imag_part()	290
6.3.3.17 get_free_indices()	290
6.3.3.18 eval_ncmul()	290
6.3.3.19 derivative()	290
6.3.3.20 return_type()	290
6.3.3.21 return_type_tinfo()	291
6.3.3.22 thisexpairseq() [1/2]	291
6.3.3.23 thisexpairseq() [2/2]	291
6.3.3.24 split_ex_to_pair()	291
6.3.3.25 combine_ex_with_coeff_to_pair()	292
6.3.3.26 combine_pair_with_coeff_to_pair()	292
6.3.3.27 recombine_pair_to_ex()	292

6.3.3.28 expand()	292
6.3.3.29 print_add()	293
6.3.3.30 do_print()	293
6.3.3.31 do_print_latex()	293
6.3.3.32 do_print_csrc()	293
6.3.3.33 do_print_python_repr()	293
6.3.4 Friends And Related Symbol Documentation	294
6.3.4.1 mul	294
6.3.4.2 power	294
6.4 GiNaC::archive Class Reference	294
6.4.1 Detailed Description	295
6.4.2 Member Typedef Documentation	296
6.4.2.1 inv_at_cit	296
6.4.3 Constructor & Destructor Documentation	296
6.4.3.1 archive() [1/3]	296
6.4.3.2 ~archive()	296
6.4.3.3 archive() [2/3]	296
6.4.3.4 archive() [3/3]	296
6.4.4 Member Function Documentation	296
6.4.4.1 archive_ex()	296
6.4.4.2 unarchive_ex() [1/3]	297
6.4.4.3 unarchive_ex() [2/3]	297
6.4.4.4 unarchive_ex() [3/3]	297
6.4.4.5 num_expressions()	299
6.4.4.6 get_top_node()	299
6.4.4.7 clear()	299
6.4.4.8 add_node()	299
6.4.4.9 get_node()	300
6.4.4.10 forget()	300
6.4.4.11 printraw()	300
6.4.4.12 atomize()	300
6.4.4.13 unatomize()	301
6.4.5 Friends And Related Symbol Documentation	301
6.4.5.1 operator<<	301
6.4.5.2 operator>>	301
6.4.6 Member Data Documentation	301
6.4.6.1 nodes	301
6.4.6.2 exprs	301
6.4.6.3 atoms	302
6.4.6.4 inverse_atoms	302
6.4.6.5 exprtable	302
6.5 GiNaC::archive_node Class Reference	302

6.5.1 Detailed Description	305
6.5.2 Member Typedef Documentation	305
6.5.2.1 propinfovector	305
6.5.2.2 archive_node_cit	305
6.5.3 Member Enumeration Documentation	305
6.5.3.1 property_type	305
6.5.4 Constructor & Destructor Documentation	306
6.5.4.1 archive_node() [1/2]	306
6.5.4.2 archive_node() [2/2]	306
6.5.5 Member Function Documentation	306
6.5.5.1 operator=()	306
6.5.5.2 add_bool()	306
6.5.5.3 add_unsigned()	306
6.5.5.4 add_string()	307
6.5.5.5 add_ex()	307
6.5.5.6 find_bool()	307
6.5.5.7 find_unsigned()	307
6.5.5.8 find_string()	308
6.5.5.9 find_first()	308
6.5.5.10 find_last()	308
6.5.5.11 find_property_range()	308
6.5.5.12 find_ex()	309
6.5.5.13 find_ex_by_loc()	309
6.5.5.14 find_ex_node()	309
6.5.5.15 get_properties()	309
6.5.5.16 unarchive()	310
6.5.5.17 has_same_ex_as()	310
6.5.5.18 has_ex()	310
6.5.5.19 get_ex()	310
6.5.5.20 forget()	310
6.5.5.21 printraw()	311
6.5.6 Friends And Related Symbol Documentation	311
6.5.6.1 operator<<	311
6.5.6.2 operator>>	311
6.5.7 Member Data Documentation	311
6.5.7.1 a	311
6.5.7.2 props	311
6.5.7.3 has_expression	312
6.5.7.4 e	312
6.6 GiNaC::archive_node::archive_node_cit_range Struct Reference	312
6.6.1 Member Data Documentation	313
6.6.1.1 begin	313

6.6.1.2 end	314
6.7 GiNaC::archive::archived_ex Struct Reference	314
6.7.1 Detailed Description	314
6.7.2 Constructor & Destructor Documentation	314
6.7.2.1 archived_ex() [1/2]	314
6.7.2.2 archived_ex() [2/2]	314
6.7.3 Member Data Documentation	315
6.7.3.1 name	315
6.7.3.2 root	315
6.8 GiNaC::basic Class Reference	315
6.8.1 Detailed Description	320
6.8.2 Constructor & Destructor Documentation	320
6.8.2.1 basic() [1/2]	320
6.8.2.2 ~basic()	320
6.8.2.3 basic() [2/2]	320
6.8.3 Member Function Documentation	320
6.8.3.1 operator=()	320
6.8.3.2 duplicate()	321
6.8.3.3 eval()	321
6.8.3.4 evalf()	321
6.8.3.5 evalm()	321
6.8.3.6 eval_integ()	322
6.8.3.7 eval_ncmul()	322
6.8.3.8 eval_indexed()	322
6.8.3.9 print()	322
6.8.3.10 dbgprint()	323
6.8.3.11 dbgprinttree()	323
6.8.3.12 precedence()	323
6.8.3.13 info()	323
6.8.3.14 nops()	324
6.8.3.15 op()	324
6.8.3.16 operator[]() [1/4]	324
6.8.3.17 operator[]() [2/4]	324
6.8.3.18 let_op()	325
6.8.3.19 operator[]() [3/4]	325
6.8.3.20 operator[]() [4/4]	325
6.8.3.21 has()	325
6.8.3.22 match()	326
6.8.3.23 match_same_type()	326
6.8.3.24 subs()	326
6.8.3.25 map()	327
6.8.3.26 accept()	327

6.8.3.27 is_polynomial()	327
6.8.3.28 degree()	327
6.8.3.29 ldegree()	327
6.8.3.30 coeff()	328
6.8.3.31 expand()	328
6.8.3.32 collect()	328
6.8.3.33 derivative()	328
6.8.3.34 series()	329
6.8.3.35 normal()	329
6.8.3.36 to_rational()	330
6.8.3.37 to_polynomial()	330
6.8.3.38 integer_content()	330
6.8.3.39 smod()	330
6.8.3.40 max_coefficient()	331
6.8.3.41 get_free_indices()	331
6.8.3.42 add_indexed()	331
6.8.3.43 scalar_mul_indexed()	332
6.8.3.44 contract_with()	332
6.8.3.45 return_type()	333
6.8.3.46 return_type_tinfo()	333
6.8.3.47 conjugate()	333
6.8.3.48 real_part()	333
6.8.3.49 imag_part()	334
6.8.3.50 compare_same_type()	334
6.8.3.51 is_equal_same_type()	334
6.8.3.52 calchash()	334
6.8.3.53 print_dispatch() [1/2]	335
6.8.3.54 print_dispatch() [2/2]	335
6.8.3.55 archive()	335
6.8.3.56 read_archive()	336
6.8.3.57 subs_one_level()	336
6.8.3.58 diff()	336
6.8.3.59 compare()	337
6.8.3.60 is_equal()	337
6.8.3.61 hold()	337
6.8.3.62 gethash()	338
6.8.3.63 setflag()	338
6.8.3.64 clearflag()	339
6.8.3.65 ensure_if_modifiable()	339
6.8.3.66 do_print()	339
6.8.3.67 do_print_tree()	339
6.8.3.68 do_print_python_repr()	339

6.8.4 Friends And Related Symbol Documentation	340
6.8.4.1 ex	340
6.8.5 Member Data Documentation	340
6.8.5.1 flags	340
6.8.5.2 hashvalue	340
6.9 GiNaC::basic_log_kernel Class Reference	341
6.9.1 Detailed Description	345
6.9.2 Member Function Documentation	346
6.9.2.1 series_coeff_impl()	346
6.9.2.2 do_print()	346
6.10 GiNaC::basic_multi_iterator< T > Class Template Reference	346
6.10.1 Detailed Description	348
6.10.2 Constructor & Destructor Documentation	348
6.10.2.1 basic_multi_iterator() [1/3]	348
6.10.2.2 basic_multi_iterator() [2/3]	348
6.10.2.3 basic_multi_iterator() [3/3]	348
6.10.2.4 ~basic_multi_iterator()	349
6.10.3 Member Function Documentation	349
6.10.3.1 size()	349
6.10.3.2 overflow()	349
6.10.3.3 get_vector()	349
6.10.3.4 operator[]() [1/2]	349
6.10.3.5 operator[]() [2/2]	350
6.10.3.6 operator()() [1/2]	350
6.10.3.7 operator()() [2/2]	350
6.10.3.8 init()	350
6.10.3.9 operator++()	350
6.10.4 Friends And Related Symbol Documentation	351
6.10.4.1 operator<<	351
6.10.5 Member Data Documentation	351
6.10.5.1 N	351
6.10.5.2 B	351
6.10.5.3 v	351
6.10.5.4 flag_overflow	351
6.11 GiNaC::basic_partition_generator Class Reference	352
6.11.1 Detailed Description	353
6.11.2 Constructor & Destructor Documentation	353
6.11.2.1 basic_partition_generator()	353
6.11.3 Member Data Documentation	353
6.11.3.1 mpngen	353
6.12 GiNaC::class_info< OPT > Class Template Reference	353
6.12.1 Constructor & Destructor Documentation	354

6.12.1.1 class_info()	354
6.12.2 Member Function Documentation	355
6.12.2.1 get_parent()	355
6.12.2.2 find()	355
6.12.2.3 dump_hierarchy()	355
6.12.2.4 dump_tree()	355
6.12.2.5 identify_parents()	355
6.12.3 Member Data Documentation	356
6.12.3.1 options	356
6.12.3.2 first	356
6.12.3.3 next	356
6.12.3.4 parent	356
6.12.3.5 parents_identified	356
6.13 GiNaC::clifford Class Reference	357
6.13.1 Detailed Description	365
6.13.2 Constructor & Destructor Documentation	365
6.13.2.1 clifford() [1/4]	365
6.13.2.2 clifford() [2/4]	365
6.13.2.3 clifford() [3/4]	365
6.13.2.4 clifford() [4/4]	366
6.13.3 Member Function Documentation	366
6.13.3.1 precedence()	366
6.13.3.2 archive()	366
6.13.3.3 read_archive()	366
6.13.3.4 eval_ncmul()	367
6.13.3.5 match_same_type()	367
6.13.3.6 thiscontainer() [1/2]	367
6.13.3.7 thiscontainer() [2/2]	367
6.13.3.8 return_type()	367
6.13.3.9 return_type_tinfo()	368
6.13.3.10 get_representation_label()	368
6.13.3.11 get_metric() [1/2]	368
6.13.3.12 get_metric() [2/2]	368
6.13.3.13 same_metric()	368
6.13.3.14 get_commutator_sign()	368
6.13.3.15 nops()	369
6.13.3.16 op()	369
6.13.3.17 let_op()	369
6.13.3.18 subs()	369
6.13.3.19 do_print_dflt()	370
6.13.3.20 do_print_latex()	370
6.13.3.21 do_print_tree()	370

6.13.4 Member Data Documentation	370
6.13.4.1 representation_label	370
6.13.4.2 metric	370
6.13.4.3 commutator_sign	371
6.14 GiNaC::cliffordunit Class Reference	371
6.14.1 Detailed Description	375
6.14.2 Member Function Documentation	376
6.14.2.1 contract_with()	376
6.14.2.2 do_print()	376
6.14.2.3 do_print_latex()	376
6.15 GiNaC::color Class Reference	376
6.15.1 Detailed Description	385
6.15.2 Constructor & Destructor Documentation	385
6.15.2.1 color() [1/4]	385
6.15.2.2 color() [2/4]	385
6.15.2.3 color() [3/4]	385
6.15.2.4 color() [4/4]	386
6.15.3 Member Function Documentation	386
6.15.3.1 archive()	386
6.15.3.2 read_archive()	386
6.15.3.3 eval_ncmul()	386
6.15.3.4 match_same_type()	387
6.15.3.5 thiscontainer() [1/2]	387
6.15.3.6 thiscontainer() [2/2]	387
6.15.3.7 return_type()	387
6.15.3.8 return_type_tinfo()	387
6.15.3.9 get_representation_label()	388
6.15.4 Member Data Documentation	388
6.15.4.1 representation_label	388
6.16 GiNaC::compare_all_equal< T > Class Template Reference	388
6.16.1 Detailed Description	388
6.16.2 Constructor & Destructor Documentation	389
6.16.2.1 ~compare_all_equal()	389
6.16.3 Member Function Documentation	389
6.16.3.1 struct_is_equal()	389
6.16.3.2 struct_compare()	389
6.17 GiNaC::compare_bitwise< T > Class Template Reference	389
6.17.1 Detailed Description	389
6.17.2 Constructor & Destructor Documentation	390
6.17.2.1 ~compare_bitwise()	390
6.17.3 Member Function Documentation	390
6.17.3.1 struct_is_equal()	390

6.17.3.2 struct_compare()	390
6.18 GiNaC::compare_std_less< T > Class Template Reference	390
6.18.1 Detailed Description	390
6.18.2 Constructor & Destructor Documentation	391
6.18.2.1 ~compare_std_less()	391
6.18.3 Member Function Documentation	391
6.18.3.1 struct_is_equal()	391
6.18.3.2 struct_compare()	391
6.19 GiNaC::composition_generator Class Reference	391
6.19.1 Detailed Description	392
6.19.2 Constructor & Destructor Documentation	393
6.19.2.1 composition_generator()	393
6.19.3 Member Function Documentation	393
6.19.3.1 get()	393
6.19.3.2 next()	393
6.19.4 Member Data Documentation	393
6.19.4.1 cmgen	393
6.19.4.2 atend	393
6.19.4.3 trivial	393
6.19.4.4 composition	394
6.19.4.5 current_updated	394
6.20 GiNaC::const_iterator Class Reference	394
6.20.1 Member Typedef Documentation	395
6.20.1.1 iterator_category	395
6.20.1.2 value_type	396
6.20.1.3 difference_type	396
6.20.1.4 pointer	396
6.20.1.5 reference	396
6.20.2 Constructor & Destructor Documentation	396
6.20.2.1 const_iterator() [1/2]	396
6.20.2.2 const_iterator() [2/2]	396
6.20.3 Member Function Documentation	396
6.20.3.1 operator*()	396
6.20.3.2 operator->()	396
6.20.3.3 operator[]()	397
6.20.3.4 operator++() [1/2]	397
6.20.3.5 operator++() [2/2]	397
6.20.3.6 operator+=()	397
6.20.3.7 operator+()	397
6.20.3.8 operator--() [1/2]	397
6.20.3.9 operator--() [2/2]	397
6.20.3.10 operator-=()	398

6.20.3.11 operator-()	398
6.20.3.12 operator==(())	398
6.20.3.13 operator"!=(())	398
6.20.3.14 operator<()	398
6.20.3.15 operator>()	398
6.20.3.16 operator<=()	398
6.20.3.17 operator>=()	398
6.20.4 Friends And Related Symbol Documentation	399
6.20.4.1 ex	399
6.20.4.2 const_preorder_iterator	399
6.20.4.3 const_postorder_iterator	399
6.20.4.4 operator+	399
6.20.4.5 operator-	399
6.20.5 Member Data Documentation	399
6.20.5.1 e	399
6.20.5.2 i	399
6.21 GiNaC::const_postorder_iterator Class Reference	400
6.21.1 Member Typedef Documentation	400
6.21.1.1 iterator_category	400
6.21.1.2 value_type	400
6.21.1.3 difference_type	400
6.21.1.4 pointer	401
6.21.1.5 reference	401
6.21.2 Constructor & Destructor Documentation	401
6.21.2.1 const_postorder_iterator() [1/2]	401
6.21.2.2 const_postorder_iterator() [2/2]	401
6.21.3 Member Function Documentation	401
6.21.3.1 operator*()	401
6.21.3.2 operator->()	401
6.21.3.3 operator++() [1/2]	401
6.21.3.4 operator++() [2/2]	402
6.21.3.5 operator==(())	402
6.21.3.6 operator"!=(())	402
6.21.3.7 descend()	402
6.21.3.8 increment()	402
6.21.4 Member Data Documentation	402
6.21.4.1 s	402
6.22 GiNaC::const_preorder_iterator Class Reference	403
6.22.1 Member Typedef Documentation	403
6.22.1.1 iterator_category	403
6.22.1.2 value_type	403
6.22.1.3 difference_type	403

6.22.1.4 pointer	404
6.22.1.5 reference	404
6.22.2 Constructor & Destructor Documentation	404
6.22.2.1 const_preorder_iterator() [1/2]	404
6.22.2.2 const_preorder_iterator() [2/2]	404
6.22.3 Member Function Documentation	404
6.22.3.1 operator*()	404
6.22.3.2 operator->()	404
6.22.3.3 operator++() [1/2]	404
6.22.3.4 operator++() [2/2]	405
6.22.3.5 operator==()	405
6.22.3.6 operator!=()	405
6.22.3.7 increment()	405
6.22.4 Member Data Documentation	405
6.22.4.1 s	405
6.23 GiNaC::constant Class Reference	406
6.23.1 Detailed Description	411
6.23.2 Constructor & Destructor Documentation	411
6.23.2.1 constant() [1/2]	411
6.23.2.2 constant() [2/2]	411
6.23.3 Member Function Documentation	411
6.23.3.1 info()	411
6.23.3.2 evalf()	412
6.23.3.3 is_polynomial()	412
6.23.3.4 conjugate()	412
6.23.3.5 real_part()	412
6.23.3.6 imag_part()	412
6.23.3.7 archive()	413
6.23.3.8 read_archive()	413
6.23.3.9 derivative()	413
6.23.3.10 is_equal_same_type()	414
6.23.3.11 calchash()	414
6.23.3.12 do_print()	414
6.23.3.13 do_print_tree()	414
6.23.3.14 do_print_latex()	414
6.23.3.15 do_print_python_repr()	415
6.23.4 Member Data Documentation	415
6.23.4.1 name	415
6.23.4.2 TeX_name	415
6.23.4.3 ef	415
6.23.4.4 number	415
6.23.4.5 serial	415

6.23.4.6 next_serial	416
6.23.4.7 domain	416
6.24 GiNaC::container< C > Class Template Reference	416
6.24.1 Detailed Description	422
6.24.2 Member Typedef Documentation	422
6.24.2.1 STLT	422
6.24.2.2 const_iterator	422
6.24.2.3 const_reverse_iterator	422
6.24.3 Constructor & Destructor Documentation	422
6.24.3.1 container() [1/4]	422
6.24.3.2 container() [2/4]	422
6.24.3.3 container() [3/4]	423
6.24.3.4 container() [4/4]	423
6.24.4 Member Function Documentation	423
6.24.4.1 get_default_flags()	423
6.24.4.2 get_open_delim()	423
6.24.4.3 get_close_delim()	423
6.24.4.4 info()	423
6.24.4.5 precedence()	424
6.24.4.6 nops()	424
6.24.4.7 op()	424
6.24.4.8 let_op()	425
6.24.4.9 subs()	425
6.24.4.10 read_archive()	425
6.24.4.11 archive()	426
6.24.4.12 conjugate()	426
6.24.4.13 real_part()	426
6.24.4.14 imag_part()	426
6.24.4.15 is_equal_same_type()	427
6.24.4.16 thiscontainer() [1/2]	427
6.24.4.17 thiscontainer() [2/2]	427
6.24.4.18 printseq()	428
6.24.4.19 sort_() [1/2]	428
6.24.4.20 sort_() [2/2]	428
6.24.4.21 unique_() [1/2]	428
6.24.4.22 prepend()	428
6.24.4.23 append()	429
6.24.4.24 remove_first()	429
6.24.4.25 remove_last()	429
6.24.4.26 remove_all()	429
6.24.4.27 sort()	429
6.24.4.28 unique()	429

6.24.4.29 begin()	430
6.24.4.30 end()	430
6.24.4.31 rbegin()	430
6.24.4.32 rend()	430
6.24.4.33 do_print()	430
6.24.4.34 do_print_tree()	431
6.24.4.35 do_print_python()	431
6.24.4.36 do_print_python_repr()	431
6.24.4.37 subschildren()	431
6.24.4.38 unique_() [2/2]	431
6.25 GiNaC::container_storage< C > Class Template Reference	432
6.25.1 Detailed Description	433
6.25.2 Member Typedef Documentation	433
6.25.2.1 STLT	433
6.25.3 Constructor & Destructor Documentation	433
6.25.3.1 container_storage() [1/4]	433
6.25.3.2 container_storage() [2/4]	433
6.25.3.3 container_storage() [3/4]	433
6.25.3.4 container_storage() [4/4]	433
6.25.3.5 ~container_storage()	433
6.25.4 Member Function Documentation	434
6.25.4.1 reserve() [1/4]	434
6.25.4.2 reserve() [2/4]	434
6.25.4.3 reserve() [3/4]	434
6.25.4.4 reserve() [4/4]	434
6.25.5 Member Data Documentation	434
6.25.5.1 seq	434
6.26 GiNaC::composition_generator::coolmulti Struct Reference	435
6.26.1 Constructor & Destructor Documentation	435
6.26.1.1 coolmulti()	435
6.26.1.2 ~coolmulti()	436
6.26.2 Member Function Documentation	436
6.26.2.1 next_permutation()	436
6.26.2.2 finished()	436
6.26.3 Member Data Documentation	436
6.26.3.1 head	436
6.26.3.2 i	436
6.26.3.3 after_i	436
6.27 GiNaC::derivative_map_function Struct Reference	437
6.27.1 Detailed Description	438
6.27.2 Constructor & Destructor Documentation	438
6.27.2.1 derivative_map_function()	438

6.27.3 Member Function Documentation	438
6.27.3.1 operator()()	438
6.27.4 Member Data Documentation	438
6.27.4.1 s	438
6.28 GiNaC::determinant_algo Class Reference	439
6.28.1 Detailed Description	439
6.28.2 Member Enumeration Documentation	439
6.28.2.1 anonymous enum	439
6.29 GiNaC::diracgamma Class Reference	440
6.29.1 Detailed Description	445
6.29.2 Member Function Documentation	445
6.29.2.1 contract_with()	445
6.29.2.2 do_print()	445
6.29.2.3 do_print_latex()	445
6.30 GiNaC::diracgamma5 Class Reference	445
6.30.1 Detailed Description	450
6.30.2 Member Function Documentation	450
6.30.2.1 conjugate()	450
6.30.2.2 do_print()	450
6.30.2.3 do_print_latex()	450
6.31 GiNaC::diracgammaL Class Reference	451
6.31.1 Detailed Description	455
6.31.2 Member Function Documentation	455
6.31.2.1 conjugate()	455
6.31.2.2 do_print()	455
6.31.2.3 do_print_latex()	455
6.32 GiNaC::diracgammaR Class Reference	456
6.32.1 Detailed Description	460
6.32.2 Member Function Documentation	460
6.32.2.1 conjugate()	460
6.32.2.2 do_print()	460
6.32.2.3 do_print_latex()	460
6.33 GiNaC::diracone Class Reference	461
6.33.1 Detailed Description	465
6.33.2 Member Function Documentation	465
6.33.2.1 do_print()	465
6.33.2.2 do_print_latex()	465
6.34 GiNaC::do_taylor Class Reference	466
6.34.1 Detailed Description	466
6.35 GiNaC::domain Class Reference	466
6.35.1 Detailed Description	466
6.35.2 Member Enumeration Documentation	466

6.35.2.1 anonymous enum	466
6.36 GiNaC::dunno Class Reference	467
6.36.1 Detailed Description	467
6.37 GiNaC::Ebar_kernel Class Reference	467
6.37.1 Detailed Description	472
6.37.2 Constructor & Destructor Documentation	473
6.37.2.1 Ebar_kernel()	473
6.37.3 Member Function Documentation	473
6.37.3.1 nops()	473
6.37.3.2 op()	473
6.37.3.3 let_op()	473
6.37.3.4 is_numeric()	473
6.37.3.5 get_numerical_value()	474
6.37.3.6 series_coeff_impl()	474
6.37.3.7 do_print()	474
6.37.4 Member Data Documentation	474
6.37.4.1 n	474
6.37.4.2 m	474
6.37.4.3 x	475
6.37.4.4 y	475
6.38 GiNaC::Eisenstein_h_kernel Class Reference	475
6.38.1 Detailed Description	480
6.38.2 Constructor & Destructor Documentation	481
6.38.2.1 Eisenstein_h_kernel()	481
6.38.3 Member Function Documentation	481
6.38.3.1 series()	481
6.38.3.2 nops()	481
6.38.3.3 op()	481
6.38.3.4 let_op()	482
6.38.3.5 is_numeric()	482
6.38.3.6 Laurent_series()	482
6.38.3.7 get_numerical_value()	482
6.38.3.8 uses_Laurent_series()	483
6.38.3.9 coefficient_a0()	483
6.38.3.10 coefficient_an()	483
6.38.3.11 q_expansion_modular_form()	483
6.38.3.12 do_print()	484
6.38.4 Member Data Documentation	484
6.38.4.1 k	484
6.38.4.2 N	484
6.38.4.3 r	484
6.38.4.4 s	484

6.38.4.5 C_norm	484
6.39 GiNaC::Eisenstein_kernel Class Reference	485
6.39.1 Detailed Description	491
6.39.2 Constructor & Destructor Documentation	491
6.39.2.1 Eisenstein_kernel()	491
6.39.3 Member Function Documentation	491
6.39.3.1 series()	491
6.39.3.2 nops()	492
6.39.3.3 op()	492
6.39.3.4 let_op()	492
6.39.3.5 is_numeric()	492
6.39.3.6 Laurent_series()	492
6.39.3.7 get_numerical_value()	493
6.39.3.8 uses_Laurent_series()	493
6.39.3.9 q_expansion_modular_form()	493
6.39.3.10 do_print()	493
6.39.4 Member Data Documentation	493
6.39.4.1 k	493
6.39.4.2 N	494
6.39.4.3 a	494
6.39.4.4 b	494
6.39.4.5 K	494
6.39.4.6 C_norm	494
6.40 GiNaC::composition_generator::coolmulti::element Struct Reference	494
6.40.1 Constructor & Destructor Documentation	495
6.40.1.1 element()	495
6.40.1.2 ~element()	495
6.40.2 Member Data Documentation	495
6.40.2.1 value	495
6.40.2.2 next	495
6.41 GiNaC::ELi_kernel Class Reference	496
6.41.1 Detailed Description	501
6.41.2 Constructor & Destructor Documentation	502
6.41.2.1 ELi_kernel()	502
6.41.3 Member Function Documentation	502
6.41.3.1 nops()	502
6.41.3.2 op()	502
6.41.3.3 let_op()	502
6.41.3.4 is_numeric()	502
6.41.3.5 get_numerical_value()	503
6.41.3.6 series_coeff_impl()	503
6.41.3.7 do_print()	503

6.41.4 Member Data Documentation	503
6.41.4.1 n	503
6.41.4.2 m	503
6.41.4.3 x	504
6.41.4.4 y	504
6.42 std::equal_to< GiNaC::ex > Struct Reference	504
6.42.1 Detailed Description	504
6.42.2 Member Function Documentation	504
6.42.2.1 operator()()	504
6.43 GiNaC::error_and_integral Struct Reference	505
6.43.1 Constructor & Destructor Documentation	505
6.43.1.1 error_and_integral()	505
6.43.2 Member Data Documentation	506
6.43.2.1 error	506
6.43.2.2 integral	506
6.44 GiNaC::error_and_integral_is_less Struct Reference	506
6.44.1 Member Function Documentation	506
6.44.1.1 operator()()	506
6.45 GiNaC::eval_integ_map_function Struct Reference	507
6.45.1 Detailed Description	508
6.45.2 Member Function Documentation	508
6.45.2.1 operator()()	508
6.46 GiNaC::evalf_map_function Struct Reference	508
6.46.1 Detailed Description	509
6.46.2 Member Function Documentation	509
6.46.2.1 operator()()	509
6.47 GiNaC::evalm_map_function Struct Reference	510
6.47.1 Detailed Description	511
6.47.2 Member Function Documentation	511
6.47.2.1 operator()()	511
6.48 GiNaC::ex Class Reference	511
6.48.1 Detailed Description	515
6.48.2 Constructor & Destructor Documentation	515
6.48.2.1 ex() [1/10]	515
6.48.2.2 ex() [2/10]	515
6.48.2.3 ex() [3/10]	515
6.48.2.4 ex() [4/10]	516
6.48.2.5 ex() [5/10]	516
6.48.2.6 ex() [6/10]	516
6.48.2.7 ex() [7/10]	516
6.48.2.8 ex() [8/10]	516
6.48.2.9 ex() [9/10]	516

6.48.2.10 ex() [10/10]	517
6.48.3 Member Function Documentation	517
6.48.3.1 swap()	517
6.48.3.2 begin()	517
6.48.3.3 end()	517
6.48.3.4 preorder_begin()	517
6.48.3.5 preorder_end()	518
6.48.3.6 postorder_begin()	518
6.48.3.7 postorder_end()	518
6.48.3.8 eval()	518
6.48.3.9 evalf()	518
6.48.3.10 evalm()	518
6.48.3.11 eval_ncmul()	519
6.48.3.12 eval_integ()	519
6.48.3.13 print()	519
6.48.3.14 dbgprint()	520
6.48.3.15 dbgprinttree()	520
6.48.3.16 info()	520
6.48.3.17 nops()	521
6.48.3.18 op()	521
6.48.3.19 operator[]() [1/4]	521
6.48.3.20 operator[]() [2/4]	522
6.48.3.21 let_op()	522
6.48.3.22 operator[]() [3/4]	522
6.48.3.23 operator[]() [4/4]	522
6.48.3.24 lhs()	522
6.48.3.25 rhs()	522
6.48.3.26 conjugate()	523
6.48.3.27 real_part()	523
6.48.3.28 imag_part()	523
6.48.3.29 has()	523
6.48.3.30 find()	524
6.48.3.31 match() [1/2]	524
6.48.3.32 match() [2/2]	524
6.48.3.33 subs() [1/3]	524
6.48.3.34 subs() [2/3]	525
6.48.3.35 subs() [3/3]	525
6.48.3.36 map() [1/2]	525
6.48.3.37 map() [2/2]	525
6.48.3.38 accept()	525
6.48.3.39 traverse_preorder()	526
6.48.3.40 traverse_postorder()	526

6.48.3.41	traverse()	526
6.48.3.42	is_polynomial()	526
6.48.3.43	degree()	526
6.48.3.44	ldegree()	527
6.48.3.45	coeff()	527
6.48.3.46	lcoeff()	527
6.48.3.47	tcoeff()	527
6.48.3.48	expand()	527
6.48.3.49	collect()	528
6.48.3.50	diff()	528
6.48.3.51	series()	529
6.48.3.52	normal()	529
6.48.3.53	to_rational()	530
6.48.3.54	to_polynomial()	530
6.48.3.55	numer()	530
6.48.3.56	denom()	531
6.48.3.57	numer_denom()	531
6.48.3.58	unit()	531
6.48.3.59	content()	532
6.48.3.60	integer_content()	532
6.48.3.61	primpart() [1/2]	533
6.48.3.62	primpart() [2/2]	533
6.48.3.63	unitcontprim()	534
6.48.3.64	smod()	534
6.48.3.65	max_coefficient()	534
6.48.3.66	get_free_indices()	535
6.48.3.67	simplify_indexed() [1/2]	535
6.48.3.68	simplify_indexed() [2/2]	535
6.48.3.69	compare()	536
6.48.3.70	is_equal()	536
6.48.3.71	is_zero()	537
6.48.3.72	is_zero_matrix()	537
6.48.3.73	symmetrize() [1/2]	537
6.48.3.74	symmetrize() [2/2]	537
6.48.3.75	antisymmetrize() [1/2]	538
6.48.3.76	antisymmetrize() [2/2]	538
6.48.3.77	symmetrize_cyclic() [1/2]	538
6.48.3.78	symmetrize_cyclic() [2/2]	538
6.48.3.79	return_type()	538
6.48.3.80	return_type_tinfo()	539
6.48.3.81	gethash()	539
6.48.3.82	construct_from_basic()	539

6.48.3.83 construct_from_int()	539
6.48.3.84 construct_from_uint()	540
6.48.3.85 construct_from_long()	540
6.48.3.86 construct_from_ulong()	540
6.48.3.87 construct_from_longlong()	540
6.48.3.88 construct_from_ulonglong()	540
6.48.3.89 construct_from_double()	540
6.48.3.90 construct_from_string_and_lst()	541
6.48.3.91 makewriteable()	541
6.48.3.92 share()	541
6.48.4 Friends And Related Symbol Documentation	541
6.48.4.1 archive_node	541
6.48.4.2 are_ex_trivially_equal	541
6.48.4.3 ex_to	541
6.48.4.4 is_a	542
6.48.4.5 is_exactly_a	542
6.48.5 Member Data Documentation	542
6.48.5.1 bp	542
6.49 GiNaC::ex_base_is_less Struct Reference	543
6.49.1 Member Function Documentation	543
6.49.1.1 operator>()	543
6.50 GiNaC::ex_is_equal Struct Reference	543
6.50.1 Member Function Documentation	543
6.50.1.1 operator>()	543
6.51 GiNaC::ex_is_less Struct Reference	543
6.51.1 Member Function Documentation	544
6.51.1.1 operator>()	544
6.52 GiNaC::ex_swap Struct Reference	544
6.52.1 Member Function Documentation	544
6.52.1.1 operator>()	544
6.53 GiNaC::expair Class Reference	545
6.53.1 Detailed Description	546
6.53.2 Constructor & Destructor Documentation	546
6.53.2.1 expair() [1/2]	546
6.53.2.2 expair() [2/2]	546
6.53.3 Member Function Documentation	546
6.53.3.1 is_equal()	546
6.53.3.2 is_less()	547
6.53.3.3 compare()	547
6.53.3.4 print()	547
6.53.3.5 is_canonical_numeric()	547
6.53.3.6 swap()	547

6.53.3.7 conjugate()	547
6.53.4 Member Data Documentation	548
6.53.4.1 rest	548
6.53.4.2 coeff	548
6.54 GiNaC::expair_is_less Struct Reference	548
6.54.1 Detailed Description	548
6.54.2 Member Function Documentation	549
6.54.2.1 operator()	549
6.55 GiNaC::expair_rest_is_less Struct Reference	549
6.55.1 Detailed Description	549
6.55.2 Member Function Documentation	549
6.55.2.1 operator()	549
6.56 GiNaC::expair_swap Struct Reference	550
6.56.1 Member Function Documentation	550
6.56.1.1 operator()	550
6.57 GiNaC::expairseq Class Reference	550
6.57.1 Detailed Description	555
6.57.2 Constructor & Destructor Documentation	555
6.57.2.1 expairseq() [1/4]	555
6.57.2.2 expairseq() [2/4]	556
6.57.2.3 expairseq() [3/4]	556
6.57.2.4 expairseq() [4/4]	556
6.57.3 Member Function Documentation	556
6.57.3.1 precedence()	556
6.57.3.2 info()	556
6.57.3.3 nops()	557
6.57.3.4 op()	557
6.57.3.5 map()	557
6.57.3.6 eval()	557
6.57.3.7 to_rational()	558
6.57.3.8 to_polynomial()	558
6.57.3.9 match()	558
6.57.3.10 subs()	558
6.57.3.11 conjugate()	559
6.57.3.12 archive()	559
6.57.3.13 read_archive()	559
6.57.3.14 is_equal_same_type()	560
6.57.3.15 return_type()	560
6.57.3.16 calchash()	560
6.57.3.17 expand()	560
6.57.3.18 thisexpairseq() [1/2]	561
6.57.3.19 thisexpairseq() [2/2]	561

6.57.3.20	printseq()	561
6.57.3.21	printpair()	561
6.57.3.22	split_ex_to_pair()	562
6.57.3.23	combine_ex_with_coeff_to_pair()	562
6.57.3.24	combine_pair_with_coeff_to_pair()	562
6.57.3.25	recombine_pair_to_ex()	562
6.57.3.26	expair_needs_further_processing()	563
6.57.3.27	default_overall_coeff()	563
6.57.3.28	combine_overall_coeff() [1/2]	563
6.57.3.29	combine_overall_coeff() [2/2]	563
6.57.3.30	can_make_flat()	563
6.57.3.31	do_print()	564
6.57.3.32	do_print_tree()	564
6.57.3.33	construct_from_2_ex()	564
6.57.3.34	construct_from_2_expairseq()	564
6.57.3.35	construct_from_expairseq_ex()	564
6.57.3.36	construct_from_exvector()	565
6.57.3.37	construct_from_epvector() [1/2]	565
6.57.3.38	construct_from_epvector() [2/2]	565
6.57.3.39	make_flat() [1/2]	565
6.57.3.40	make_flat() [2/2]	565
6.57.3.41	canonicalize()	566
6.57.3.42	combine_same_terms_sorted_seq()	566
6.57.3.43	is_canonical()	566
6.57.3.44	expandchildren()	566
6.57.3.45	evalchildren()	567
6.57.3.46	subchildren()	567
6.57.4	Member Data Documentation	567
6.57.4.1	seq	567
6.57.4.2	overall_coeff	568
6.58	GiNaC::expand_map_function Struct Reference	568
6.58.1	Detailed Description	569
6.58.2	Constructor & Destructor Documentation	569
6.58.2.1	expand_map_function()	569
6.58.3	Member Function Documentation	570
6.58.3.1	operator>()	570
6.58.4	Member Data Documentation	570
6.58.4.1	options	570
6.59	GiNaC::expand_options Class Reference	570
6.59.1	Detailed Description	570
6.59.2	Member Enumeration Documentation	570
6.59.2.1	anonymous enum	570

6.60 GiNaC::factor_options Class Reference	571
6.60.1 Detailed Description	571
6.60.2 Member Enumeration Documentation	571
6.60.2.1 anonymous enum	571
6.61 GiNaC::fail Class Reference	571
6.61.1 Member Function Documentation	575
6.61.1.1 return_type()	575
6.61.1.2 do_print()	576
6.62 GiNaC::fderivative Class Reference	576
6.62.1 Detailed Description	584
6.62.2 Constructor & Destructor Documentation	584
6.62.2.1 fderivative() [1/3]	584
6.62.2.2 fderivative() [2/3]	584
6.62.2.3 fderivative() [3/3]	585
6.62.3 Member Function Documentation	585
6.62.3.1 print()	585
6.62.3.2 eval()	585
6.62.3.3 series()	586
6.62.3.4 thiscontainer() [1/2]	586
6.62.3.5 thiscontainer() [2/2]	586
6.62.3.6 archive()	586
6.62.3.7 read_archive()	587
6.62.3.8 derivative()	587
6.62.3.9 is_equal_same_type()	587
6.62.3.10 match_same_type()	588
6.62.3.11 derivatives()	588
6.62.3.12 do_print()	588
6.62.3.13 do_print_latex()	588
6.62.3.14 do_print_csrc()	589
6.62.3.15 do_print_tree()	589
6.62.4 Member Data Documentation	589
6.62.4.1 parameter_set	589
6.63 GiNaC::function Class Reference	589
6.63.1 Detailed Description	597
6.63.2 Constructor & Destructor Documentation	597
6.63.2.1 function() [1/18]	597
6.63.2.2 function() [2/18]	597
6.63.2.3 function() [3/18]	597
6.63.2.4 function() [4/18]	597
6.63.2.5 function() [5/18]	597
6.63.2.6 function() [6/18]	598
6.63.2.7 function() [7/18]	598

6.63.2.8 function() [8/18]	598
6.63.2.9 function() [9/18]	598
6.63.2.10 function() [10/18]	599
6.63.2.11 function() [11/18]	599
6.63.2.12 function() [12/18]	599
6.63.2.13 function() [13/18]	599
6.63.2.14 function() [14/18]	600
6.63.2.15 function() [15/18]	600
6.63.2.16 function() [16/18]	600
6.63.2.17 function() [17/18]	601
6.63.2.18 function() [18/18]	601
6.63.3 Member Function Documentation	601
6.63.3.1 print()	601
6.63.3.2 precedence()	601
6.63.3.3 expand()	602
6.63.3.4 eval()	602
6.63.3.5 evalf()	602
6.63.3.6 eval_ncmul()	602
6.63.3.7 calchash()	603
6.63.3.8 series()	603
6.63.3.9 thiscontainer() [1/2]	603
6.63.3.10 thiscontainer() [2/2]	603
6.63.3.11 conjugate()	604
6.63.3.12 real_part()	604
6.63.3.13 imag_part()	604
6.63.3.14 archive()	604
6.63.3.15 read_archive()	605
6.63.3.16 info()	605
6.63.3.17 derivative()	605
6.63.3.18 is_equal_same_type()	605
6.63.3.19 match_same_type()	606
6.63.3.20 return_type()	606
6.63.3.21 return_type_tinfo()	606
6.63.3.22 pderivative()	606
6.63.3.23 expl_derivative()	607
6.63.3.24 registered_functions()	607
6.63.3.25 lookup_remember_table()	607
6.63.3.26 store_remember_table()	607
6.63.3.27 power()	607
6.63.3.28 register_new()	607
6.63.3.29 find_function()	608
6.63.3.30 get_registered_functions()	608

6.63.3.31	get_serial()	608
6.63.3.32	get_name()	608
6.63.4	Friends And Related Symbol Documentation	608
6.63.4.1	remember_table_entry	608
6.63.5	Member Data Documentation	608
6.63.5.1	current_serial	608
6.63.5.2	serial	609
6.64	GiNaC::function_options Class Reference	609
6.64.1	Constructor & Destructor Documentation	615
6.64.1.1	function_options() [1/3]	615
6.64.1.2	function_options() [2/3]	615
6.64.1.3	function_options() [3/3]	616
6.64.1.4	~function_options()	616
6.64.2	Member Function Documentation	616
6.64.2.1	initialize()	616
6.64.2.2	dummy()	616
6.64.2.3	set_name()	616
6.64.2.4	latex_name()	616
6.64.2.5	eval_func() [1/15]	617
6.64.2.6	eval_func() [2/15]	617
6.64.2.7	eval_func() [3/15]	617
6.64.2.8	eval_func() [4/15]	617
6.64.2.9	eval_func() [5/15]	617
6.64.2.10	eval_func() [6/15]	617
6.64.2.11	eval_func() [7/15]	617
6.64.2.12	eval_func() [8/15]	618
6.64.2.13	eval_func() [9/15]	618
6.64.2.14	eval_func() [10/15]	618
6.64.2.15	eval_func() [11/15]	618
6.64.2.16	eval_func() [12/15]	618
6.64.2.17	eval_func() [13/15]	618
6.64.2.18	eval_func() [14/15]	618
6.64.2.19	evalf_func() [1/15]	619
6.64.2.20	evalf_func() [2/15]	619
6.64.2.21	evalf_func() [3/15]	619
6.64.2.22	evalf_func() [4/15]	619
6.64.2.23	evalf_func() [5/15]	619
6.64.2.24	evalf_func() [6/15]	619
6.64.2.25	evalf_func() [7/15]	619
6.64.2.26	evalf_func() [8/15]	620
6.64.2.27	evalf_func() [9/15]	620
6.64.2.28	evalf_func() [10/15]	620

6.64.2.29 evalf_func() [11/15]	620
6.64.2.30 evalf_func() [12/15]	620
6.64.2.31 evalf_func() [13/15]	620
6.64.2.32 evalf_func() [14/15]	620
6.64.2.33 conjugate_func() [1/15]	621
6.64.2.34 conjugate_func() [2/15]	621
6.64.2.35 conjugate_func() [3/15]	621
6.64.2.36 conjugate_func() [4/15]	621
6.64.2.37 conjugate_func() [5/15]	621
6.64.2.38 conjugate_func() [6/15]	621
6.64.2.39 conjugate_func() [7/15]	621
6.64.2.40 conjugate_func() [8/15]	622
6.64.2.41 conjugate_func() [9/15]	622
6.64.2.42 conjugate_func() [10/15]	622
6.64.2.43 conjugate_func() [11/15]	622
6.64.2.44 conjugate_func() [12/15]	622
6.64.2.45 conjugate_func() [13/15]	622
6.64.2.46 conjugate_func() [14/15]	622
6.64.2.47 real_part_func() [1/15]	623
6.64.2.48 real_part_func() [2/15]	623
6.64.2.49 real_part_func() [3/15]	623
6.64.2.50 real_part_func() [4/15]	623
6.64.2.51 real_part_func() [5/15]	623
6.64.2.52 real_part_func() [6/15]	623
6.64.2.53 real_part_func() [7/15]	623
6.64.2.54 real_part_func() [8/15]	624
6.64.2.55 real_part_func() [9/15]	624
6.64.2.56 real_part_func() [10/15]	624
6.64.2.57 real_part_func() [11/15]	624
6.64.2.58 real_part_func() [12/15]	624
6.64.2.59 real_part_func() [13/15]	624
6.64.2.60 real_part_func() [14/15]	624
6.64.2.61 imag_part_func() [1/15]	625
6.64.2.62 imag_part_func() [2/15]	625
6.64.2.63 imag_part_func() [3/15]	625
6.64.2.64 imag_part_func() [4/15]	625
6.64.2.65 imag_part_func() [5/15]	625
6.64.2.66 imag_part_func() [6/15]	625
6.64.2.67 imag_part_func() [7/15]	625
6.64.2.68 imag_part_func() [8/15]	626
6.64.2.69 imag_part_func() [9/15]	626
6.64.2.70 imag_part_func() [10/15]	626

6.64.2.71 imag_part_func() [11/15]	626
6.64.2.72 imag_part_func() [12/15]	626
6.64.2.73 imag_part_func() [13/15]	626
6.64.2.74 imag_part_func() [14/15]	626
6.64.2.75 expand_func() [1/15]	627
6.64.2.76 expand_func() [2/15]	627
6.64.2.77 expand_func() [3/15]	627
6.64.2.78 expand_func() [4/15]	627
6.64.2.79 expand_func() [5/15]	627
6.64.2.80 expand_func() [6/15]	627
6.64.2.81 expand_func() [7/15]	627
6.64.2.82 expand_func() [8/15]	628
6.64.2.83 expand_func() [9/15]	628
6.64.2.84 expand_func() [10/15]	628
6.64.2.85 expand_func() [11/15]	628
6.64.2.86 expand_func() [12/15]	628
6.64.2.87 expand_func() [13/15]	628
6.64.2.88 expand_func() [14/15]	628
6.64.2.89 derivative_func() [1/15]	629
6.64.2.90 derivative_func() [2/15]	629
6.64.2.91 derivative_func() [3/15]	629
6.64.2.92 derivative_func() [4/15]	629
6.64.2.93 derivative_func() [5/15]	629
6.64.2.94 derivative_func() [6/15]	629
6.64.2.95 derivative_func() [7/15]	629
6.64.2.96 derivative_func() [8/15]	630
6.64.2.97 derivative_func() [9/15]	630
6.64.2.98 derivative_func() [10/15]	630
6.64.2.99 derivative_func() [11/15]	630
6.64.2.100 derivative_func() [12/15]	630
6.64.2.101 derivative_func() [13/15]	630
6.64.2.102 derivative_func() [14/15]	630
6.64.2.103 expl_derivative_func() [1/15]	631
6.64.2.104 expl_derivative_func() [2/15]	631
6.64.2.105 expl_derivative_func() [3/15]	631
6.64.2.106 expl_derivative_func() [4/15]	631
6.64.2.107 expl_derivative_func() [5/15]	631
6.64.2.108 expl_derivative_func() [6/15]	631
6.64.2.109 expl_derivative_func() [7/15]	631
6.64.2.110 expl_derivative_func() [8/15]	632
6.64.2.111 expl_derivative_func() [9/15]	632
6.64.2.112 expl_derivative_func() [10/15]	632

6.64.2.113 expl_derivative_func() [11/15]	632
6.64.2.114 expl_derivative_func() [12/15]	632
6.64.2.115 expl_derivative_func() [13/15]	632
6.64.2.116 expl_derivative_func() [14/15]	632
6.64.2.117 power_func() [1/15]	633
6.64.2.118 power_func() [2/15]	633
6.64.2.119 power_func() [3/15]	633
6.64.2.120 power_func() [4/15]	633
6.64.2.121 power_func() [5/15]	633
6.64.2.122 power_func() [6/15]	633
6.64.2.123 power_func() [7/15]	633
6.64.2.124 power_func() [8/15]	634
6.64.2.125 power_func() [9/15]	634
6.64.2.126 power_func() [10/15]	634
6.64.2.127 power_func() [11/15]	634
6.64.2.128 power_func() [12/15]	634
6.64.2.129 power_func() [13/15]	634
6.64.2.130 power_func() [14/15]	634
6.64.2.131 series_func() [1/15]	635
6.64.2.132 series_func() [2/15]	635
6.64.2.133 series_func() [3/15]	635
6.64.2.134 series_func() [4/15]	635
6.64.2.135 series_func() [5/15]	635
6.64.2.136 series_func() [6/15]	635
6.64.2.137 series_func() [7/15]	635
6.64.2.138 series_func() [8/15]	636
6.64.2.139 series_func() [9/15]	636
6.64.2.140 series_func() [10/15]	636
6.64.2.141 series_func() [11/15]	636
6.64.2.142 series_func() [12/15]	636
6.64.2.143 series_func() [13/15]	636
6.64.2.144 series_func() [14/15]	636
6.64.2.145 info_func() [1/15]	637
6.64.2.146 info_func() [2/15]	637
6.64.2.147 info_func() [3/15]	637
6.64.2.148 info_func() [4/15]	637
6.64.2.149 info_func() [5/15]	637
6.64.2.150 info_func() [6/15]	637
6.64.2.151 info_func() [7/15]	637
6.64.2.152 info_func() [8/15]	638
6.64.2.153 info_func() [9/15]	638
6.64.2.154 info_func() [10/15]	638

6.64.2.155 info_func() [11/15]	638
6.64.2.156 info_func() [12/15]	638
6.64.2.157 info_func() [13/15]	638
6.64.2.158 info_func() [14/15]	638
6.64.2.159 eval_func() [15/15]	639
6.64.2.160 evalf_func() [15/15]	639
6.64.2.161 conjugate_func() [15/15]	639
6.64.2.162 real_part_func() [15/15]	639
6.64.2.163 imag_part_func() [15/15]	639
6.64.2.164 expand_func() [15/15]	639
6.64.2.165 derivative_func() [15/15]	639
6.64.2.166 expl_derivative_func() [15/15]	640
6.64.2.167 power_func() [15/15]	640
6.64.2.168 series_func() [15/15]	640
6.64.2.169 info_func() [15/15]	640
6.64.2.170 print_func() [1/15]	640
6.64.2.171 print_func() [2/15]	640
6.64.2.172 print_func() [3/15]	641
6.64.2.173 print_func() [4/15]	641
6.64.2.174 print_func() [5/15]	641
6.64.2.175 print_func() [6/15]	641
6.64.2.176 print_func() [7/15]	641
6.64.2.177 print_func() [8/15]	641
6.64.2.178 print_func() [9/15]	642
6.64.2.179 print_func() [10/15]	642
6.64.2.180 print_func() [11/15]	642
6.64.2.181 print_func() [12/15]	642
6.64.2.182 print_func() [13/15]	642
6.64.2.183 print_func() [14/15]	642
6.64.2.184 print_func() [15/15]	643
6.64.2.185 set_return_type()	643
6.64.2.186 do_not_evalf_params()	643
6.64.2.187 remember()	643
6.64.2.188 overloaded()	643
6.64.2.189 set_symmetry()	643
6.64.2.190 get_name()	644
6.64.2.191 get_nparams()	644
6.64.2.192 has_derivative()	644
6.64.2.193 has_power()	644
6.64.2.194 test_and_set_nparams()	644
6.64.2.195 set_print_func()	645
6.64.3 Friends And Related Symbol Documentation	645

6.64.3.1 function	645
6.64.3.2 fderivative	645
6.64.4 Member Data Documentation	645
6.64.4.1 name	645
6.64.4.2 TeX_name	645
6.64.4.3 nparams	645
6.64.4.4 eval_f	646
6.64.4.5 evalf_f	646
6.64.4.6 conjugate_f	646
6.64.4.7 real_part_f	646
6.64.4.8 imag_part_f	646
6.64.4.9 expand_f	646
6.64.4.10 derivative_f	647
6.64.4.11 expl_derivative_f	647
6.64.4.12 power_f	647
6.64.4.13 series_f	647
6.64.4.14 print_dispatch_table	647
6.64.4.15 info_f	647
6.64.4.16 evalf_params_first	648
6.64.4.17 use_return_type	648
6.64.4.18 return_type	648
6.64.4.19 return_type_tinfo	648
6.64.4.20 use_remember	648
6.64.4.21 remember_size	648
6.64.4.22 remember_assoc_size	648
6.64.4.23 remember_strategy	648
6.64.4.24 eval_use_exvector_args	649
6.64.4.25 evalf_use_exvector_args	649
6.64.4.26 conjugate_use_exvector_args	649
6.64.4.27 real_part_use_exvector_args	649
6.64.4.28 imag_part_use_exvector_args	649
6.64.4.29 expand_use_exvector_args	649
6.64.4.30 derivative_use_exvector_args	649
6.64.4.31 expl_derivative_use_exvector_args	649
6.64.4.32 power_use_exvector_args	650
6.64.4.33 series_use_exvector_args	650
6.64.4.34 print_use_exvector_args	650
6.64.4.35 info_use_exvector_args	650
6.64.4.36 functions_with_same_name	650
6.64.4.37 symtree	650
6.65 GiNaC::G2_SERIAL Class Reference	650
6.65.1 Detailed Description	651

6.65.2 Member Data Documentation	651
6.65.2.1 serial	651
6.66 GiNaC::G3_SERIAL Class Reference	651
6.66.1 Detailed Description	651
6.66.2 Member Data Documentation	652
6.66.2.1 serial	652
6.67 GiNaC::gcd_options Struct Reference	652
6.67.1 Detailed Description	652
6.67.2 Member Enumeration Documentation	652
6.67.2.1 anonymous enum	652
6.68 GiNaC::gcdheu_failed Class Reference	653
6.68.1 Detailed Description	653
6.69 GiNaC::has_distance< T > Class Template Reference	653
6.69.1 Detailed Description	653
6.69.2 Member Typedef Documentation	654
6.69.2.1 yes_type	654
6.69.2.2 no_type	654
6.69.3 Member Enumeration Documentation	654
6.69.3.1 anonymous enum	654
6.69.4 Member Function Documentation	654
6.69.4.1 test() [1/2]	654
6.69.4.2 test() [2/2]	654
6.70 GiNaC::has_options Class Reference	655
6.70.1 Detailed Description	655
6.70.2 Member Enumeration Documentation	655
6.70.2.1 anonymous enum	655
6.71 std::hash< GiNaC::ex > Struct Reference	655
6.71.1 Detailed Description	655
6.71.2 Member Function Documentation	656
6.71.2.1 operator()()	656
6.72 GiNaC::idx Class Reference	656
6.72.1 Detailed Description	661
6.72.2 Constructor & Destructor Documentation	661
6.72.2.1 idx()	661
6.72.3 Member Function Documentation	661
6.72.3.1 info()	661
6.72.3.2 nops()	662
6.72.3.3 op()	662
6.72.3.4 map()	662
6.72.3.5 evalf()	662
6.72.3.6 subs()	663
6.72.3.7 archive()	663

6.72.3.8	read_archive()	663
6.72.3.9	derivative()	664
6.72.3.10	match_same_type()	664
6.72.3.11	calchash()	664
6.72.3.12	is_dummy_pair_same_type()	665
6.72.3.13	get_value()	665
6.72.3.14	is_numeric()	665
6.72.3.15	is_symbolic()	665
6.72.3.16	get_dim()	665
6.72.3.17	is_dim_numeric()	666
6.72.3.18	is_dim_symbolic()	666
6.72.3.19	replace_dim()	666
6.72.3.20	minimal_dim()	666
6.72.3.21	print_index()	666
6.72.3.22	do_print()	667
6.72.3.23	do_print_csrc()	667
6.72.3.24	do_print_latex()	667
6.72.3.25	do_print_tree()	667
6.72.4	Member Data Documentation	667
6.72.4.1	value	667
6.72.4.2	dim	668
6.73	GiNaC::idx_is_equal_ignore_dim Struct Reference	668
6.73.1	Member Function Documentation	668
6.73.1.1	operator()()	668
6.74	GiNaC::indexed Class Reference	669
6.74.1	Detailed Description	676
6.74.2	Constructor & Destructor Documentation	676
6.74.2.1	indexed() [1/13]	676
6.74.2.2	indexed() [2/13]	677
6.74.2.3	indexed() [3/13]	677
6.74.2.4	indexed() [4/13]	677
6.74.2.5	indexed() [5/13]	678
6.74.2.6	indexed() [6/13]	678
6.74.2.7	indexed() [7/13]	678
6.74.2.8	indexed() [8/13]	679
6.74.2.9	indexed() [9/13]	679
6.74.2.10	indexed() [10/13]	680
6.74.2.11	indexed() [11/13]	680
6.74.2.12	indexed() [12/13]	680
6.74.2.13	indexed() [13/13]	680
6.74.3	Member Function Documentation	681
6.74.3.1	precedence()	681

6.74.3.2	info()	681
6.74.3.3	eval()	681
6.74.3.4	real_part()	681
6.74.3.5	imag_part()	682
6.74.3.6	get_free_indices()	682
6.74.3.7	archive()	682
6.74.3.8	read_archive()	682
6.74.3.9	derivative()	683
6.74.3.10	thiscontainer() [1/2]	683
6.74.3.11	thiscontainer() [2/2]	683
6.74.3.12	return_type()	683
6.74.3.13	return_type_tinfo()	683
6.74.3.14	expand()	684
6.74.3.15	all_index_values_are()	684
6.74.3.16	get_indices()	684
6.74.3.17	get_dummy_indices() [1/2]	684
6.74.3.18	get_dummy_indices() [2/2]	684
6.74.3.19	has_dummy_index_for()	685
6.74.3.20	get_symmetry()	685
6.74.3.21	printindices()	685
6.74.3.22	print_indexed()	685
6.74.3.23	do_print()	685
6.74.3.24	do_print_latex()	686
6.74.3.25	do_print_tree()	686
6.74.3.26	validate()	686
6.74.4	Friends And Related Symbol Documentation	686
6.74.4.1	simplify_indexed	686
6.74.4.2	simplify_indexed_product	686
6.74.4.3	reposition_dummy_indices	687
6.74.5	Member Data Documentation	687
6.74.5.1	symtree	687
6.75	GiNaC::info_flags Class Reference	687
6.75.1	Detailed Description	688
6.75.2	Member Enumeration Documentation	688
6.75.2.1	anonymous enum	688
6.76	GiNaC::integral Class Reference	689
6.76.1	Detailed Description	694
6.76.2	Constructor & Destructor Documentation	694
6.76.2.1	integral()	694
6.76.3	Member Function Documentation	694
6.76.3.1	precedence()	694
6.76.3.2	eval()	694

6.76.3.3 evalf()	694
6.76.3.4 degree()	695
6.76.3.5 ldegree()	695
6.76.3.6 eval_ncmul()	695
6.76.3.7 nops()	695
6.76.3.8 op()	695
6.76.3.9 let_op()	696
6.76.3.10 expand()	696
6.76.3.11 get_free_indices()	696
6.76.3.12 return_type()	696
6.76.3.13 return_type_tinfo()	696
6.76.3.14 conjugate()	697
6.76.3.15 eval_integ()	697
6.76.3.16 archive()	697
6.76.3.17 read_archive()	697
6.76.3.18 derivative()	698
6.76.3.19 series()	698
6.76.3.20 do_print()	698
6.76.3.21 do_print_latex()	698
6.76.4 Member Data Documentation	699
6.76.4.1 max_integration_level	699
6.76.4.2 relative_integration_error	699
6.76.4.3 x	699
6.76.4.4 a	699
6.76.4.5 b	699
6.76.4.6 f	699
6.77 GiNaC::integration_kernel Class Reference	700
6.77.1 Detailed Description	704
6.77.2 Member Function Documentation	705
6.77.2.1 series()	705
6.77.2.2 has_trailing_zero()	705
6.77.2.3 is_numeric()	705
6.77.2.4 Laurent_series()	705
6.77.2.5 get_numerical_value()	706
6.77.2.6 uses_Laurent_series()	706
6.77.2.7 series_coeff_impl()	706
6.77.2.8 get_cache_size()	706
6.77.2.9 set_cache_step()	706
6.77.2.10 get_series_coeff()	707
6.77.2.11 series_coeff()	707
6.77.2.12 get_numerical_value_impl()	707
6.77.2.13 do_print()	707

6.77.3 Member Data Documentation	707
6.77.3.1 cache_step_size	707
6.77.3.2 series_vec	708
6.78 GiNaC::is_not_a_clifford Struct Reference	708
6.78.1 Detailed Description	708
6.78.2 Member Function Documentation	708
6.78.2.1 operator()	708
6.79 GiNaC::is_summation_idx Struct Reference	708
6.79.1 Member Function Documentation	709
6.79.1.1 operator()	709
6.80 GiNaC::iterated_integral2_SERIAL Class Reference	709
6.80.1 Detailed Description	709
6.80.2 Member Data Documentation	709
6.80.2.1 serial	709
6.81 GiNaC::iterated_integral3_SERIAL Class Reference	710
6.81.1 Detailed Description	710
6.81.2 Member Data Documentation	710
6.81.2.1 serial	710
6.82 GiNaC::Kronecker_dtau_kernel Class Reference	710
6.82.1 Detailed Description	716
6.82.2 Constructor & Destructor Documentation	717
6.82.2.1 Kronecker_dtau_kernel()	717
6.82.3 Member Function Documentation	717
6.82.3.1 nops()	717
6.82.3.2 op()	717
6.82.3.3 let_op()	717
6.82.3.4 is_numeric()	717
6.82.3.5 get_numerical_value()	718
6.82.3.6 series_coeff_impl()	718
6.82.3.7 do_print()	718
6.82.4 Member Data Documentation	718
6.82.4.1 n	718
6.82.4.2 z	718
6.82.4.3 K	719
6.82.4.4 C_norm	719
6.83 GiNaC::Kronecker_dz_kernel Class Reference	719
6.83.1 Detailed Description	724
6.83.2 Constructor & Destructor Documentation	725
6.83.2.1 Kronecker_dz_kernel()	725
6.83.3 Member Function Documentation	725
6.83.3.1 nops()	725
6.83.3.2 op()	725

6.83.3.3 let_op()	725
6.83.3.4 is_numeric()	725
6.83.3.5 get_numerical_value()	726
6.83.3.6 series_coeff_impl()	726
6.83.3.7 do_print()	726
6.83.4 Member Data Documentation	726
6.83.4.1 n	726
6.83.4.2 z_j	726
6.83.4.3 tau	727
6.83.4.4 K	727
6.83.4.5 C_norm	727
6.84 GiNaC::lanczos_coeffs Class Reference	727
6.84.1 Constructor & Destructor Documentation	727
6.84.1.1 lanczos_coeffs()	727
6.84.2 Member Function Documentation	728
6.84.2.1 sufficiently_accurate()	728
6.84.2.2 get_order()	728
6.84.2.3 calc_lanczos_A()	728
6.84.3 Member Data Documentation	728
6.84.3.1 coeffs	728
6.84.3.2 current_vector	728
6.85 std::less< GiNaC::ptr< T > > Struct Template Reference	729
6.85.1 Detailed Description	729
6.85.2 Member Function Documentation	729
6.85.2.1 operator>()	729
6.86 GiNaC::library_init Class Reference	729
6.86.1 Detailed Description	730
6.86.2 Constructor & Destructor Documentation	730
6.86.2.1 library_init()	730
6.86.2.2 ~library_init()	731
6.86.3 Member Function Documentation	731
6.86.3.1 init_unarchivers()	731
6.86.4 Member Data Documentation	731
6.86.4.1 count	731
6.87 GiNaC::make_flat_inserter Class Reference	731
6.87.1 Detailed Description	732
6.87.2 Constructor & Destructor Documentation	732
6.87.2.1 make_flat_inserter() [1/2]	732
6.87.2.2 make_flat_inserter() [2/2]	732
6.87.3 Member Function Documentation	732
6.87.3.1 handle_factor()	732
6.87.3.2 combine_indices()	733

6.87.4 Member Data Documentation	733
6.87.4.1 do_renaming	733
6.87.4.2 used_indices	733
6.88 GiNaC::map_function Struct Reference	733
6.88.1 Detailed Description	735
6.88.2 Member Typedef Documentation	735
6.88.2.1 argument_type	735
6.88.2.2 result_type	735
6.88.3 Constructor & Destructor Documentation	735
6.88.3.1 ~map_function()	735
6.88.4 Member Function Documentation	735
6.88.4.1 operator()()	735
6.89 GiNaC::matrix Class Reference	736
6.89.1 Detailed Description	741
6.89.2 Constructor & Destructor Documentation	741
6.89.2.1 matrix() [1/5]	741
6.89.2.2 matrix() [2/5]	741
6.89.2.3 matrix() [3/5]	742
6.89.2.4 matrix() [4/5]	742
6.89.2.5 matrix() [5/5]	742
6.89.3 Member Function Documentation	742
6.89.3.1 nops()	742
6.89.3.2 op()	743
6.89.3.3 let_op()	743
6.89.3.4 evalm()	743
6.89.3.5 subs()	743
6.89.3.6 eval_indexed()	744
6.89.3.7 add_indexed()	744
6.89.3.8 scalar_mul_indexed()	744
6.89.3.9 contract_with()	744
6.89.3.10 conjugate()	745
6.89.3.11 real_part()	745
6.89.3.12 imag_part()	745
6.89.3.13 archive()	745
6.89.3.14 read_archive()	745
6.89.3.15 match_same_type()	746
6.89.3.16 return_type()	746
6.89.3.17 rows()	746
6.89.3.18 cols()	746
6.89.3.19 add()	746
6.89.3.20 sub()	747
6.89.3.21 mul() [1/2]	747

6.89.3.22 mul() [2/2]	747
6.89.3.23 mul_scalar()	748
6.89.3.24 pow()	748
6.89.3.25 operator>() [1/2]	748
6.89.3.26 operator>() [2/2]	748
6.89.3.27 set()	749
6.89.3.28 transpose()	749
6.89.3.29 determinant()	749
6.89.3.30 trace()	750
6.89.3.31 charpoly()	750
6.89.3.32 inverse() [1/2]	751
6.89.3.33 inverse() [2/2]	751
6.89.3.34 solve()	752
6.89.3.35 rank() [1/2]	752
6.89.3.36 rank() [2/2]	753
6.89.3.37 is_zero_matrix()	753
6.89.3.38 determinant_minor()	753
6.89.3.39 echelon_form()	753
6.89.3.40 gauss_elimination()	753
6.89.3.41 division_free_elimination()	754
6.89.3.42 fraction_free_elimination()	754
6.89.3.43 markowitz_elimination()	755
6.89.3.44 pivot()	755
6.89.3.45 print_elements()	756
6.89.3.46 do_print()	756
6.89.3.47 do_print_latex()	756
6.89.3.48 do_print_python_repr()	756
6.89.4 Member Data Documentation	756
6.89.4.1 row	756
6.89.4.2 col	757
6.89.4.3 m	757
6.90 GiNaC::minkmetric Class Reference	757
6.90.1 Detailed Description	763
6.90.2 Constructor & Destructor Documentation	763
6.90.2.1 minkmetric()	763
6.90.3 Member Function Documentation	763
6.90.3.1 info()	763
6.90.3.2 eval_indexed()	763
6.90.3.3 archive()	764
6.90.3.4 read_archive()	764
6.90.3.5 return_type()	764
6.90.3.6 do_print()	764

6.90.3.7 do_print_latex()	764
6.90.4 Member Data Documentation	765
6.90.4.1 pos_sig	765
6.91 GiNaC::modular_form_kernel Class Reference	765
6.91.1 Detailed Description	770
6.91.2 Constructor & Destructor Documentation	770
6.91.2.1 modular_form_kernel()	770
6.91.3 Member Function Documentation	771
6.91.3.1 series()	771
6.91.3.2 nops()	771
6.91.3.3 op()	771
6.91.3.4 let_op()	771
6.91.3.5 is_numeric()	772
6.91.3.6 Laurent_series()	772
6.91.3.7 get_numerical_value()	772
6.91.3.8 uses_Laurent_series()	772
6.91.3.9 q_expansion_modular_form()	773
6.91.3.10 do_print()	773
6.91.4 Member Data Documentation	773
6.91.4.1 k	773
6.91.4.2 P	773
6.91.4.3 C_norm	773
6.92 GiNaC::basic_partition_generator::mpartition2 Struct Reference	773
6.92.1 Constructor & Destructor Documentation	774
6.92.1.1 mpartition2()	774
6.92.2 Member Function Documentation	774
6.92.2.1 next_partition()	774
6.92.3 Member Data Documentation	774
6.92.3.1 x	774
6.92.3.2 n	774
6.92.3.3 m	775
6.93 GiNaC::mul Class Reference	775
6.93.1 Detailed Description	781
6.93.2 Constructor & Destructor Documentation	781
6.93.2.1 mul() [1/7]	781
6.93.2.2 mul() [2/7]	781
6.93.2.3 mul() [3/7]	781
6.93.2.4 mul() [4/7]	782
6.93.2.5 mul() [5/7]	782
6.93.2.6 mul() [6/7]	782
6.93.2.7 mul() [7/7]	782
6.93.3 Member Function Documentation	782

6.93.3.1 precedence()	782
6.93.3.2 info()	783
6.93.3.3 is_polynomial()	783
6.93.3.4 degree()	783
6.93.3.5 ldegree()	783
6.93.3.6 coeff()	784
6.93.3.7 has()	784
6.93.3.8 eval()	784
6.93.3.9 evalf()	785
6.93.3.10 real_part()	785
6.93.3.11 imag_part()	785
6.93.3.12 evalm()	785
6.93.3.13 series()	785
6.93.3.14 normal()	786
6.93.3.15 integer_content()	786
6.93.3.16 smod()	786
6.93.3.17 max_coefficient()	787
6.93.3.18 get_free_indices()	787
6.93.3.19 conjugate()	787
6.93.3.20 derivative()	787
6.93.3.21 eval_ncmul()	788
6.93.3.22 return_type()	788
6.93.3.23 return_type_tinfo()	788
6.93.3.24 thisexpairseq() [1/2]	788
6.93.3.25 thisexpairseq() [2/2]	788
6.93.3.26 split_ex_to_pair()	789
6.93.3.27 combine_ex_with_coeff_to_pair()	789
6.93.3.28 combine_pair_with_coeff_to_pair()	789
6.93.3.29 recombine_pair_to_ex()	789
6.93.3.30 expair_needs_further_processing()	790
6.93.3.31 default_overall_coeff()	790
6.93.3.32 combine_overall_coeff() [1/2]	790
6.93.3.33 combine_overall_coeff() [2/2]	790
6.93.3.34 can_make_flat()	790
6.93.3.35 expand()	791
6.93.3.36 algebraic_subs_mul()	791
6.93.3.37 find_real_imag()	791
6.93.3.38 print_overall_coeff()	791
6.93.3.39 do_print()	792
6.93.3.40 do_print_latex()	792
6.93.3.41 do_print_csrc()	792
6.93.3.42 do_print_python_repr()	792

6.93.3.43 can_be_further_expanded()	792
6.93.3.44 expandchildren()	793
6.93.4 Friends And Related Symbol Documentation	793
6.93.4.1 add	793
6.93.4.2 ncmul	793
6.93.4.3 power	793
6.94 GiNaC::multi_iterator_counter< T > Class Template Reference	794
6.94.1 Detailed Description	795
6.94.2 Constructor & Destructor Documentation	796
6.94.2.1 multi_iterator_counter() [1/3]	796
6.94.2.2 multi_iterator_counter() [2/3]	796
6.94.2.3 multi_iterator_counter() [3/3]	796
6.94.3 Member Function Documentation	796
6.94.3.1 init()	796
6.94.3.2 operator++()	797
6.94.4 Friends And Related Symbol Documentation	797
6.94.4.1 operator<<	797
6.95 GiNaC::multi_iterator_counter_indv< T > Class Template Reference	797
6.95.1 Detailed Description	799
6.95.2 Constructor & Destructor Documentation	799
6.95.2.1 multi_iterator_counter_indv() [1/3]	799
6.95.2.2 multi_iterator_counter_indv() [2/3]	799
6.95.2.3 multi_iterator_counter_indv() [3/3]	800
6.95.3 Member Function Documentation	800
6.95.3.1 init()	800
6.95.3.2 operator++()	800
6.95.4 Friends And Related Symbol Documentation	800
6.95.4.1 operator<<	800
6.95.5 Member Data Documentation	801
6.95.5.1 Nv	801
6.96 GiNaC::multi_iterator_ordered< T > Class Template Reference	801
6.96.1 Detailed Description	803
6.96.2 Constructor & Destructor Documentation	803
6.96.2.1 multi_iterator_ordered() [1/3]	803
6.96.2.2 multi_iterator_ordered() [2/3]	803
6.96.2.3 multi_iterator_ordered() [3/3]	803
6.96.3 Member Function Documentation	804
6.96.3.1 init()	804
6.96.3.2 operator++()	804
6.96.4 Friends And Related Symbol Documentation	804
6.96.4.1 operator<<	804
6.97 GiNaC::multi_iterator_ordered_eq< T > Class Template Reference	805

6.97.1 Detailed Description	806
6.97.2 Constructor & Destructor Documentation	807
6.97.2.1 multi_iterator_ordered_eq() [1/3]	807
6.97.2.2 multi_iterator_ordered_eq() [2/3]	807
6.97.2.3 multi_iterator_ordered_eq() [3/3]	807
6.97.3 Member Function Documentation	807
6.97.3.1 init()	807
6.97.3.2 operator++()	808
6.97.4 Friends And Related Symbol Documentation	808
6.97.4.1 operator<<	808
6.98 GiNaC::multi_iterator_ordered_eq_indv< T > Class Template Reference	808
6.98.1 Detailed Description	810
6.98.2 Constructor & Destructor Documentation	810
6.98.2.1 multi_iterator_ordered_eq_indv() [1/3]	810
6.98.2.2 multi_iterator_ordered_eq_indv() [2/3]	810
6.98.2.3 multi_iterator_ordered_eq_indv() [3/3]	811
6.98.3 Member Function Documentation	811
6.98.3.1 init()	811
6.98.3.2 operator++()	811
6.98.4 Friends And Related Symbol Documentation	811
6.98.4.1 operator<<	811
6.98.5 Member Data Documentation	812
6.98.5.1 Nv	812
6.99 GiNaC::multi_iterator_permutation< T > Class Template Reference	812
6.99.1 Detailed Description	814
6.99.2 Constructor & Destructor Documentation	814
6.99.2.1 multi_iterator_permutation() [1/3]	814
6.99.2.2 multi_iterator_permutation() [2/3]	814
6.99.2.3 multi_iterator_permutation() [3/3]	814
6.99.3 Member Function Documentation	815
6.99.3.1 init()	815
6.99.3.2 operator++()	815
6.99.3.3 get_sign()	815
6.99.4 Friends And Related Symbol Documentation	816
6.99.4.1 operator<<	816
6.100 GiNaC::multi_iterator_shuffle< T > Class Template Reference	816
6.100.1 Detailed Description	818
6.100.2 Constructor & Destructor Documentation	818
6.100.2.1 multi_iterator_shuffle() [1/2]	818
6.100.2.2 multi_iterator_shuffle() [2/2]	818
6.100.3 Member Function Documentation	819
6.100.3.1 init()	819

6.100.3.2 operator++()	819
6.100.4 Friends And Related Symbol Documentation	819
6.100.4.1 operator<<	819
6.100.5 Member Data Documentation	819
6.100.5.1 N_internal	819
6.100.5.2 v_internal	820
6.100.5.3 v_orig	820
6.101 GiNaC::multi_iterator_shuffle_prime< T > Class Template Reference	820
6.101.1 Detailed Description	822
6.101.2 Constructor & Destructor Documentation	823
6.101.2.1 multi_iterator_shuffle_prime() [1/2]	823
6.101.2.2 multi_iterator_shuffle_prime() [2/2]	823
6.101.3 Member Function Documentation	823
6.101.3.1 init()	823
6.101.4 Friends And Related Symbol Documentation	823
6.101.4.1 operator<<	823
6.102 GiNaC::multiple_polylog_kernel Class Reference	824
6.102.1 Detailed Description	829
6.102.2 Constructor & Destructor Documentation	829
6.102.2.1 multiple_polylog_kernel()	829
6.102.3 Member Function Documentation	830
6.102.3.1 nops()	830
6.102.3.2 op()	830
6.102.3.3 let_op()	830
6.102.3.4 is_numeric()	830
6.102.3.5 series_coeff_impl()	830
6.102.3.6 do_print()	831
6.102.4 Member Data Documentation	831
6.102.4.1 z	831
6.103 GiNaC::ncmul Class Reference	831
6.103.1 Detailed Description	838
6.103.2 Constructor & Destructor Documentation	838
6.103.2.1 ncmul() [1/7]	838
6.103.2.2 ncmul() [2/7]	838
6.103.2.3 ncmul() [3/7]	838
6.103.2.4 ncmul() [4/7]	838
6.103.2.5 ncmul() [5/7]	838
6.103.2.6 ncmul() [6/7]	839
6.103.2.7 ncmul() [7/7]	839
6.103.3 Member Function Documentation	839
6.103.3.1 precedence()	839
6.103.3.2 info()	839

6.103.3.3 degree()	839
6.103.3.4 ldegree()	840
6.103.3.5 expand()	840
6.103.3.6 coeff()	840
6.103.3.7 eval()	840
6.103.3.8 evalm()	841
6.103.3.9 get_free_indices()	841
6.103.3.10 thiscontainer() [1/2]	841
6.103.3.11 thiscontainer() [2/2]	841
6.103.3.12 conjugate()	841
6.103.3.13 real_part()	841
6.103.3.14 imag_part()	842
6.103.3.15 derivative()	842
6.103.3.16 return_type()	842
6.103.3.17 return_type_tinfo()	842
6.103.3.18 do_print()	842
6.103.3.19 do_print_csrc()	843
6.103.3.20 count_factors()	843
6.103.3.21 append_factors()	843
6.103.3.22 expandchildren()	843
6.103.3.23 get_factors()	843
6.103.4 Friends And Related Symbol Documentation	843
6.103.4.1 power	843
6.103.4.2 reeval_ncmul	844
6.103.4.3 hold_ncmul	844
6.104 GiNaC::normal_map_function Struct Reference	844
6.104.1 Detailed Description	845
6.104.2 Member Function Documentation	845
6.104.2.1 operator()()	845
6.105 GiNaC::numeric Class Reference	846
6.105.1 Detailed Description	852
6.105.2 Constructor & Destructor Documentation	852
6.105.2.1 numeric() [1/10]	852
6.105.2.2 numeric() [2/10]	852
6.105.2.3 numeric() [3/10]	852
6.105.2.4 numeric() [4/10]	852
6.105.2.5 numeric() [5/10]	853
6.105.2.6 numeric() [6/10]	853
6.105.2.7 numeric() [7/10]	853
6.105.2.8 numeric() [8/10]	853
6.105.2.9 numeric() [9/10]	853
6.105.2.10 numeric() [10/10]	854

6.105.3 Member Function Documentation	854
6.105.3.1 precedence()	854
6.105.3.2 info()	854
6.105.3.3 is_polynomial()	854
6.105.3.4 degree()	855
6.105.3.5 ldegree()	855
6.105.3.6 coeff()	855
6.105.3.7 has()	855
6.105.3.8 eval()	856
6.105.3.9 evalf()	856
6.105.3.10 subs()	856
6.105.3.11 normal()	856
6.105.3.12 to_rational()	857
6.105.3.13 to_polynomial()	857
6.105.3.14 integer_content()	857
6.105.3.15 smod()	857
6.105.3.16 max_coefficient()	858
6.105.3.17 conjugate()	858
6.105.3.18 real_part()	858
6.105.3.19 imag_part()	858
6.105.3.20 archive()	859
6.105.3.21 read_archive()	859
6.105.3.22 derivative()	859
6.105.3.23 is_equal_same_type()	859
6.105.3.24 calchash()	860
6.105.3.25 add()	860
6.105.3.26 sub()	860
6.105.3.27 mul()	860
6.105.3.28 div()	860
6.105.3.29 power()	861
6.105.3.30 add_dyn()	861
6.105.3.31 sub_dyn()	861
6.105.3.32 mul_dyn()	862
6.105.3.33 div_dyn()	862
6.105.3.34 power_dyn()	862
6.105.3.35 operator=() [1/6]	862
6.105.3.36 operator=() [2/6]	863
6.105.3.37 operator=() [3/6]	863
6.105.3.38 operator=() [4/6]	863
6.105.3.39 operator=() [5/6]	863
6.105.3.40 operator=() [6/6]	863
6.105.3.41 inverse()	863

6.105.3.42 step()	864
6.105.3.43 csgn()	864
6.105.3.44 compare()	864
6.105.3.45 is_equal()	865
6.105.3.46 is_zero()	865
6.105.3.47 is_positive()	865
6.105.3.48 is_negative()	865
6.105.3.49 is_integer()	865
6.105.3.50 is_pos_integer()	866
6.105.3.51 is_nonneg_integer()	866
6.105.3.52 is_even()	866
6.105.3.53 is_odd()	866
6.105.3.54 is_prime()	866
6.105.3.55 is_rational()	867
6.105.3.56 is_real()	867
6.105.3.57 is_cinteger()	867
6.105.3.58 is_crational()	867
6.105.3.59 operator==(())	867
6.105.3.60 operator!=(())	867
6.105.3.61 operator<()	867
6.105.3.62 operator<=()	868
6.105.3.63 operator>()	868
6.105.3.64 operator>=()	868
6.105.3.65 to_int()	869
6.105.3.66 to_long()	869
6.105.3.67 to_double()	869
6.105.3.68 to_cl_N()	869
6.105.3.69 real()	870
6.105.3.70 imag()	870
6.105.3.71 numer()	870
6.105.3.72 denom()	870
6.105.3.73 int_length()	871
6.105.3.74 print_numeric()	871
6.105.3.75 do_print()	871
6.105.3.76 do_print_latex()	871
6.105.3.77 do_print_csrc()	872
6.105.3.78 do_print_csrc_cl_N()	872
6.105.3.79 do_print_tree()	872
6.105.3.80 do_print_python_repr()	872
6.105.4 Member Data Documentation	872
6.105.4.1 value	872
6.106 GiNaC::op0_is_equal Struct Reference	873

6.106.1 Member Function Documentation	873
6.106.1.1 operator()()	873
6.107 GiNaC::partition_generator Class Reference	873
6.107.1 Detailed Description	874
6.107.2 Constructor & Destructor Documentation	875
6.107.2.1 partition_generator()	875
6.107.3 Member Function Documentation	875
6.107.3.1 get()	875
6.107.3.2 next()	875
6.107.4 Member Data Documentation	875
6.107.4.1 partition	875
6.107.4.2 current_updated	875
6.108 GiNaC::partition_with_zero_parts_generator Class Reference	876
6.108.1 Detailed Description	877
6.108.2 Constructor & Destructor Documentation	877
6.108.2.1 partition_with_zero_parts_generator()	877
6.108.3 Member Function Documentation	877
6.108.3.1 get()	877
6.108.3.2 next()	877
6.108.4 Member Data Documentation	878
6.108.4.1 m	878
6.108.4.2 partition	878
6.108.4.3 current_updated	878
6.109 GiNaC::pointer_to_map_function Class Reference	878
6.109.1 Constructor & Destructor Documentation	880
6.109.1.1 pointer_to_map_function()	880
6.109.2 Member Function Documentation	880
6.109.2.1 operator()()	880
6.109.3 Member Data Documentation	880
6.109.3.1 ptr	880
6.110 GiNaC::pointer_to_map_function_1arg< T1 > Class Template Reference	880
6.110.1 Constructor & Destructor Documentation	882
6.110.1.1 pointer_to_map_function_1arg()	882
6.110.2 Member Function Documentation	882
6.110.2.1 operator()()	882
6.110.3 Member Data Documentation	882
6.110.3.1 ptr	882
6.110.3.2 arg1	882
6.111 GiNaC::pointer_to_map_function_2args< T1, T2 > Class Template Reference	883
6.111.1 Constructor & Destructor Documentation	884
6.111.1.1 pointer_to_map_function_2args()	884
6.111.2 Member Function Documentation	884

6.111.2.1 operator()()	884
6.111.3 Member Data Documentation	885
6.111.3.1 ptr	885
6.111.3.2 arg1	885
6.111.3.3 arg2	885
6.112 GiNaC::pointer_to_map_function_3args< T1, T2, T3 > Class Template Reference	885
6.112.1 Constructor & Destructor Documentation	887
6.112.1.1 pointer_to_map_function_3args()	887
6.112.2 Member Function Documentation	887
6.112.2.1 operator()()	887
6.112.3 Member Data Documentation	887
6.112.3.1 ptr	887
6.112.3.2 arg1	887
6.112.3.3 arg2	887
6.112.3.4 arg3	888
6.113 GiNaC::pointer_to_member_to_map_function< C > Class Template Reference	888
6.113.1 Constructor & Destructor Documentation	890
6.113.1.1 pointer_to_member_to_map_function()	890
6.113.2 Member Function Documentation	890
6.113.2.1 operator()()	890
6.113.3 Member Data Documentation	890
6.113.3.1 ptr	890
6.113.3.2 c	890
6.114 GiNaC::pointer_to_member_to_map_function_1arg< C, T1 > Class Template Reference	891
6.114.1 Constructor & Destructor Documentation	892
6.114.1.1 pointer_to_member_to_map_function_1arg()	892
6.114.2 Member Function Documentation	892
6.114.2.1 operator()()	892
6.114.3 Member Data Documentation	893
6.114.3.1 ptr	893
6.114.3.2 c	893
6.114.3.3 arg1	893
6.115 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 > Class Template Reference	893
6.115.1 Constructor & Destructor Documentation	895
6.115.1.1 pointer_to_member_to_map_function_2args()	895
6.115.2 Member Function Documentation	895
6.115.2.1 operator()()	895
6.115.3 Member Data Documentation	895
6.115.3.1 ptr	895
6.115.3.2 c	895
6.115.3.3 arg1	895
6.115.3.4 arg2	896

6.116 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 > Class Template Reference	896
6.116.1 Constructor & Destructor Documentation	898
6.116.1.1 pointer_to_member_to_map_function_3args()	898
6.116.2 Member Function Documentation	898
6.116.2.1 operator>()	898
6.116.3 Member Data Documentation	898
6.116.3.1 ptr	898
6.116.3.2 c	898
6.116.3.3 arg1	899
6.116.3.4 arg2	899
6.116.3.5 arg3	899
6.117 GiNaC::pole_error Class Reference	899
6.117.1 Detailed Description	900
6.117.2 Constructor & Destructor Documentation	900
6.117.2.1 pole_error()	900
6.117.3 Member Function Documentation	900
6.117.3.1 degree()	900
6.117.4 Member Data Documentation	901
6.117.4.1 deg	901
6.118 GiNaC::possymbol Class Reference	901
6.118.1 Detailed Description	906
6.118.2 Constructor & Destructor Documentation	906
6.118.2.1 possymbol() [1/3]	906
6.118.2.2 possymbol() [2/3]	906
6.118.2.3 possymbol() [3/3]	906
6.118.3 Member Function Documentation	906
6.118.3.1 get_domain()	906
6.118.3.2 duplicate()	907
6.119 GiNaC::power Class Reference	907
6.119.1 Detailed Description	912
6.119.2 Constructor & Destructor Documentation	912
6.119.2.1 power() [1/2]	912
6.119.2.2 power() [2/2]	912
6.119.3 Member Function Documentation	912
6.119.3.1 precedence()	912
6.119.3.2 info()	913
6.119.3.3 nops()	913
6.119.3.4 op()	913
6.119.3.5 map()	913
6.119.3.6 is_polynomial()	914
6.119.3.7 degree()	914
6.119.3.8 ldegree()	914

6.119.3.9 coeff()	914
6.119.3.10 eval()	915
6.119.3.11 evalf()	915
6.119.3.12 evalm()	916
6.119.3.13 series()	916
6.119.3.14 subs()	916
6.119.3.15 has()	917
6.119.3.16 normal()	917
6.119.3.17 to_rational()	917
6.119.3.18 to_polynomial()	918
6.119.3.19 conjugate()	918
6.119.3.20 real_part()	918
6.119.3.21 imag_part()	918
6.119.3.22 archive()	918
6.119.3.23 read_archive()	919
6.119.3.24 derivative()	919
6.119.3.25 eval_ncmul()	919
6.119.3.26 return_type()	919
6.119.3.27 return_type_tinfo()	919
6.119.3.28 expand()	920
6.119.3.29 print_power()	920
6.119.3.30 do_print_dflt()	920
6.119.3.31 do_print_latex()	920
6.119.3.32 do_print_csrc()	921
6.119.3.33 do_print_python()	921
6.119.3.34 do_print_python_repr()	921
6.119.3.35 do_print_csrc_cl_N()	921
6.119.3.36 expand_add()	921
6.119.3.37 expand_add_2()	922
6.119.3.38 expand_mul()	922
6.119.4 Friends And Related Symbol Documentation	922
6.119.4.1 mul	922
6.119.5 Member Data Documentation	923
6.119.5.1 basis	923
6.119.5.2 exponent	923
6.120 GiNaC::print_context Class Reference	923
6.120.1 Detailed Description	924
6.120.2 Constructor & Destructor Documentation	924
6.120.2.1 print_context()	924
6.120.2.2 ~print_context()	924
6.120.3 Member Data Documentation	924
6.120.3.1 s	924

6.120.3.2 options	924
6.121 GiNaC::print_context_options Class Reference	925
6.121.1 Detailed Description	925
6.121.2 Constructor & Destructor Documentation	925
6.121.2.1 print_context_options()	925
6.121.3 Member Function Documentation	925
6.121.3.1 get_name()	925
6.121.3.2 get_parent_name()	926
6.121.3.3 get_id()	926
6.121.4 Member Data Documentation	926
6.121.4.1 name	926
6.121.4.2 parent_name	926
6.121.4.3 id	926
6.122 GiNaC::print_csrc Class Reference	927
6.122.1 Detailed Description	928
6.122.2 Constructor & Destructor Documentation	928
6.122.2.1 print_csrc()	928
6.123 GiNaC::print_csrc_cl_N Class Reference	928
6.123.1 Detailed Description	929
6.123.2 Constructor & Destructor Documentation	930
6.123.2.1 print_csrc_cl_N()	930
6.124 GiNaC::print_csrc_double Class Reference	930
6.124.1 Detailed Description	931
6.124.2 Constructor & Destructor Documentation	932
6.124.2.1 print_csrc_double()	932
6.125 GiNaC::print_csrc_float Class Reference	932
6.125.1 Detailed Description	933
6.125.2 Constructor & Destructor Documentation	934
6.125.2.1 print_csrc_float()	934
6.126 GiNaC::print_dflt Class Reference	934
6.126.1 Detailed Description	935
6.126.2 Constructor & Destructor Documentation	935
6.126.2.1 print_dflt()	935
6.127 GiNaC::print_functor Class Reference	935
6.127.1 Detailed Description	936
6.127.2 Constructor & Destructor Documentation	936
6.127.2.1 print_functor() [1/5]	936
6.127.2.2 print_functor() [2/5]	936
6.127.2.3 print_functor() [3/5]	936
6.127.2.4 print_functor() [4/5]	936
6.127.2.5 print_functor() [5/5]	937
6.127.3 Member Function Documentation	937

6.127.3.1 operator=()	937
6.127.3.2 operator()()	937
6.127.3.3 is_valid()	937
6.127.4 Member Data Documentation	937
6.127.4.1 impl	937
6.128 GiNaC::print_functor_impl Class Reference	938
6.128.1 Detailed Description	938
6.128.2 Constructor & Destructor Documentation	938
6.128.2.1 ~print_functor_impl()	938
6.128.3 Member Function Documentation	938
6.128.3.1 duplicate()	938
6.128.3.2 operator()()	939
6.129 GiNaC::print_latex Class Reference	939
6.129.1 Detailed Description	940
6.129.2 Constructor & Destructor Documentation	940
6.129.2.1 print_latex()	940
6.130 GiNaC::print_memfun_handler< T, C > Class Template Reference	941
6.130.1 Detailed Description	942
6.130.2 Member Typedef Documentation	942
6.130.2.1 F	942
6.130.3 Constructor & Destructor Documentation	942
6.130.3.1 print_memfun_handler()	942
6.130.4 Member Function Documentation	942
6.130.4.1 duplicate()	942
6.130.4.2 operator()()	942
6.130.5 Member Data Documentation	943
6.130.5.1 f	943
6.131 GiNaC::print_options Class Reference	943
6.131.1 Detailed Description	943
6.131.2 Member Enumeration Documentation	943
6.131.2.1 anonymous enum	943
6.132 GiNaC::print_ptrfun_handler< T, C > Class Template Reference	943
6.132.1 Detailed Description	945
6.132.2 Member Typedef Documentation	945
6.132.2.1 F	945
6.132.3 Constructor & Destructor Documentation	945
6.132.3.1 print_ptrfun_handler()	945
6.132.4 Member Function Documentation	945
6.132.4.1 duplicate()	945
6.132.4.2 operator()()	945
6.132.5 Member Data Documentation	946
6.132.5.1 f	946

6.133 GiNaC::print_python Class Reference	946
6.133.1 Detailed Description	947
6.133.2 Constructor & Destructor Documentation	947
6.133.2.1 print_python()	947
6.134 GiNaC::print_python_repr Class Reference	948
6.134.1 Detailed Description	949
6.134.2 Constructor & Destructor Documentation	949
6.134.2.1 print_python_repr()	949
6.135 GiNaC::print_tree Class Reference	949
6.135.1 Detailed Description	950
6.135.2 Constructor & Destructor Documentation	950
6.135.2.1 print_tree() [1/2]	950
6.135.2.2 print_tree() [2/2]	950
6.135.3 Member Data Documentation	951
6.135.3.1 delta_indent	951
6.136 GiNaC::archive_node::property Struct Reference	951
6.136.1 Detailed Description	951
6.136.2 Constructor & Destructor Documentation	951
6.136.2.1 property() [1/2]	951
6.136.2.2 property() [2/2]	952
6.136.3 Member Data Documentation	952
6.136.3.1 type	952
6.136.3.2 name	952
6.136.3.3 value	952
6.137 GiNaC::archive_node::property_info Struct Reference	952
6.137.1 Detailed Description	953
6.137.2 Constructor & Destructor Documentation	953
6.137.2.1 property_info() [1/2]	953
6.137.2.2 property_info() [2/2]	953
6.137.3 Member Data Documentation	953
6.137.3.1 type	953
6.137.3.2 name	953
6.137.3.3 count	954
6.138 GiNaC::pseries Class Reference	954
6.138.1 Detailed Description	959
6.138.2 Constructor & Destructor Documentation	959
6.138.2.1 pseries() [1/2]	959
6.138.2.2 pseries() [2/2]	960
6.138.3 Member Function Documentation	960
6.138.3.1 precedence()	960
6.138.3.2 nops()	960
6.138.3.3 op()	960

6.138.3.4 degree()	961
6.138.3.5 ldegree()	961
6.138.3.6 coeff()	961
6.138.3.7 collect()	962
6.138.3.8 eval()	962
6.138.3.9 evalf()	962
6.138.3.10 series()	962
6.138.3.11 subs()	962
6.138.3.12 normal()	963
6.138.3.13 expand()	963
6.138.3.14 conjugate()	963
6.138.3.15 real_part()	963
6.138.3.16 imag_part()	964
6.138.3.17 eval_integ()	964
6.138.3.18 evalm()	964
6.138.3.19 archive()	964
6.138.3.20 read_archive()	964
6.138.3.21 derivative()	965
6.138.3.22 get_var()	965
6.138.3.23 get_point()	965
6.138.3.24 convert_to_poly()	965
6.138.3.25 is_compatible_to()	965
6.138.3.26 is_zero()	966
6.138.3.27 is_terminating()	966
6.138.3.28 coeffop()	966
6.138.3.29 exponop()	966
6.138.3.30 add_series()	966
6.138.3.31 mul_const()	967
6.138.3.32 mul_series()	967
6.138.3.33 power_const()	968
6.138.3.34 shift_exponents()	968
6.138.3.35 print_series()	968
6.138.3.36 do_print()	968
6.138.3.37 do_print_latex()	969
6.138.3.38 do_print_tree()	969
6.138.3.39 do_print_python()	969
6.138.3.40 do_print_python_repr()	969
6.138.4 Member Data Documentation	969
6.138.4.1 seq	969
6.138.4.2 var	970
6.138.4.3 point	970
6.139 GiNaC::psi1_SERIAL Class Reference	970

6.139.1 Detailed Description	970
6.139.2 Member Data Documentation	971
6.139.2.1 serial	971
6.140 GiNaC::psi2_SERIAL Class Reference	971
6.140.1 Detailed Description	971
6.140.2 Member Data Documentation	971
6.140.2.1 serial	971
6.141 GiNaC::ptr< T > Class Template Reference	972
6.141.1 Detailed Description	973
6.141.2 Constructor & Destructor Documentation	973
6.141.2.1 ptr() [1/3]	973
6.141.2.2 ptr() [2/3]	973
6.141.2.3 ptr() [3/3]	973
6.141.2.4 ~ptr()	974
6.141.3 Member Function Documentation	974
6.141.3.1 operator=()	974
6.141.3.2 operator*()	974
6.141.3.3 operator->()	974
6.141.3.4 makewritable()	974
6.141.3.5 swap()	974
6.141.3.6 operator==()	975
6.141.3.7 operator"!=()	975
6.141.4 Friends And Related Symbol Documentation	975
6.141.4.1 std::less< ptr< T > >	975
6.141.4.2 get_pointer	975
6.141.4.3 operator== [1/2]	975
6.141.4.4 operator"!= [1/2]	975
6.141.4.5 operator== [2/2]	976
6.141.4.6 operator"!= [2/2]	976
6.141.4.7 operator<<	976
6.141.5 Member Data Documentation	976
6.141.5.1 p	976
6.142 GiNaC::realsymbol Class Reference	977
6.142.1 Detailed Description	982
6.142.2 Constructor & Destructor Documentation	982
6.142.2.1 realsymbol() [1/3]	982
6.142.2.2 realsymbol() [2/3]	982
6.142.2.3 realsymbol() [3/3]	982
6.142.3 Member Function Documentation	982
6.142.3.1 get_domain()	982
6.142.3.2 conjugate()	982
6.142.3.3 real_part()	983

6.142.3.4 <code>imag_part()</code>	983
6.142.3.5 <code>duplicate()</code>	983
6.143 <code>GiNaC::refcounted</code> Class Reference	983
6.143.1 Detailed Description	985
6.143.2 Constructor & Destructor Documentation	985
6.143.2.1 <code>refcounted()</code>	985
6.143.3 Member Function Documentation	985
6.143.3.1 <code>add_reference()</code>	985
6.143.3.2 <code>remove_reference()</code>	985
6.143.3.3 <code>get_refcount()</code>	985
6.143.3.4 <code>set_refcount()</code>	985
6.143.4 Member Data Documentation	986
6.143.4.1 <code>refcount</code>	986
6.144 <code>GiNaC::registered_class_options</code> Class Reference	986
6.144.1 Detailed Description	986
6.144.2 Constructor & Destructor Documentation	987
6.144.2.1 <code>registered_class_options()</code>	987
6.144.3 Member Function Documentation	987
6.144.3.1 <code>get_name()</code>	987
6.144.3.2 <code>get_parent_name()</code>	987
6.144.3.3 <code>get_id()</code>	987
6.144.3.4 <code>get_print_dispatch_table()</code>	987
6.144.3.5 <code>print_func()</code> [1/3]	987
6.144.3.6 <code>print_func()</code> [2/3]	988
6.144.3.7 <code>print_func()</code> [3/3]	988
6.144.3.8 <code>set_print_func()</code>	988
6.144.4 Member Data Documentation	988
6.144.4.1 <code>name</code>	988
6.144.4.2 <code>parent_name</code>	988
6.144.4.3 <code>tinfo_key</code>	988
6.144.4.4 <code>print_dispatch_table</code>	989
6.145 <code>GiNaC::relational</code> Class Reference	989
6.145.1 Detailed Description	994
6.145.2 Member Typedef Documentation	994
6.145.2.1 <code>safe_bool</code>	994
6.145.3 Member Enumeration Documentation	994
6.145.3.1 <code>operators</code>	994
6.145.4 Constructor & Destructor Documentation	995
6.145.4.1 <code>relational()</code>	995
6.145.5 Member Function Documentation	995
6.145.5.1 <code>precedence()</code>	995
6.145.5.2 <code>info()</code>	995

6.145.5.3 nops()	995
6.145.5.4 op()	996
6.145.5.5 map()	996
6.145.5.6 subs()	996
6.145.5.7 archive()	996
6.145.5.8 read_archive()	997
6.145.5.9 canonical()	997
6.145.5.10 eval_ncmul()	997
6.145.5.11 match_same_type()	997
6.145.5.12 return_type()	998
6.145.5.13 return_type_tinfo()	998
6.145.5.14 calchash()	998
6.145.5.15 do_print()	998
6.145.5.16 do_print_python_repr()	998
6.145.5.17 lhs()	999
6.145.5.18 rhs()	999
6.145.5.19 make_safe_bool()	999
6.145.5.20 operator safe_bool()	999
6.145.5.21 operator"!()	999
6.145.6 Member Data Documentation	1000
6.145.6.1 lh	1000
6.145.6.2 rh	1000
6.145.6.3 o	1000
6.146 GiNaC::remember_strategies Class Reference	1000
6.146.1 Detailed Description	1000
6.146.2 Member Enumeration Documentation	1000
6.146.2.1 anonymous enum	1000
6.147 GiNaC::remember_table Class Reference	1001
6.147.1 Detailed Description	1002
6.147.2 Constructor & Destructor Documentation	1002
6.147.2.1 remember_table() [1/2]	1002
6.147.2.2 remember_table() [2/2]	1003
6.147.3 Member Function Documentation	1003
6.147.3.1 lookup_entry()	1003
6.147.3.2 add_entry()	1003
6.147.3.3 clear_all_entries()	1003
6.147.3.4 show_statistics()	1003
6.147.3.5 remember_tables()	1003
6.147.3.6 init_table()	1004
6.147.4 Member Data Documentation	1004
6.147.4.1 table_size	1004
6.147.4.2 max_assoc_size	1004

6.147.4.3 remember_strategy	1004
6.148 GiNaC::remember_table_entry Class Reference	1005
6.148.1 Detailed Description	1006
6.148.2 Constructor & Destructor Documentation	1006
6.148.2.1 remember_table_entry()	1006
6.148.3 Member Function Documentation	1006
6.148.3.1 is_equal()	1006
6.148.3.2 get_result()	1006
6.148.3.3 get_last_access()	1006
6.148.3.4 get_successful_hits()	1007
6.148.4 Member Data Documentation	1007
6.148.4.1 hashvalue	1007
6.148.4.2 seq	1007
6.148.4.3 result	1007
6.148.4.4 last_access	1007
6.148.4.5 successful_hits	1007
6.148.4.6 access_counter	1007
6.149 GiNaC::remember_table_list Class Reference	1008
6.149.1 Detailed Description	1009
6.149.2 Constructor & Destructor Documentation	1009
6.149.2.1 remember_table_list()	1009
6.149.3 Member Function Documentation	1009
6.149.3.1 add_entry()	1009
6.149.3.2 lookup_entry()	1009
6.149.4 Member Data Documentation	1009
6.149.4.1 max_assoc_size	1009
6.149.4.2 remember_strategy	1009
6.150 GiNaC::return_type_t Struct Reference	1010
6.150.1 Detailed Description	1010
6.150.2 Member Function Documentation	1010
6.150.2.1 operator<()	1010
6.150.2.2 operator==(())	1010
6.150.2.3 operator"!=(())	1011
6.150.3 Member Data Documentation	1011
6.150.3.1 tinfo	1011
6.150.3.2 rl	1011
6.151 GiNaC::return_types Class Reference	1011
6.151.1 Member Enumeration Documentation	1011
6.151.1.1 anonymous enum	1011
6.152 GiNaC::relational::safe_bool_helper Struct Reference	1012
6.152.1 Member Function Documentation	1012
6.152.1.1 nonnull()	1012

6.153 GiNaC::scalar_products Class Reference	1012
6.153.1 Detailed Description	1013
6.153.2 Member Function Documentation	1013
6.153.2.1 add() [1/2]	1013
6.153.2.2 add() [2/2]	1013
6.153.2.3 add_vectors()	1013
6.153.2.4 clear()	1014
6.153.2.5 is_defined()	1014
6.153.2.6 evaluate()	1014
6.153.2.7 debugprint()	1014
6.153.3 Member Data Documentation	1014
6.153.3.1 spm	1014
6.154 GiNaC::series_options Class Reference	1015
6.154.1 Detailed Description	1015
6.154.2 Member Enumeration Documentation	1015
6.154.2.1 anonymous enum	1015
6.155 GiNaC::solve_algo Class Reference	1015
6.155.1 Detailed Description	1015
6.155.2 Member Enumeration Documentation	1016
6.155.2.1 anonymous enum	1016
6.156 GiNaC::spinidx Class Reference	1016
6.156.1 Detailed Description	1023
6.156.2 Constructor & Destructor Documentation	1023
6.156.2.1 spinidx()	1023
6.156.3 Member Function Documentation	1024
6.156.3.1 is_dummy_pair_same_type()	1024
6.156.3.2 conjugate()	1024
6.156.3.3 archive()	1024
6.156.3.4 read_archive()	1024
6.156.3.5 match_same_type()	1025
6.156.3.6 is_dotted()	1025
6.156.3.7 is_undotted()	1025
6.156.3.8 toggle_dot()	1025
6.156.3.9 toggle_variance_dot()	1026
6.156.3.10 do_print()	1026
6.156.3.11 do_print_latex()	1026
6.156.3.12 do_print_tree()	1026
6.156.4 Member Data Documentation	1026
6.156.4.1 dotted	1026
6.157 GiNaC::spinmetric Class Reference	1027
6.157.1 Detailed Description	1032
6.157.2 Member Function Documentation	1032

6.157.2.1 info()	1032
6.157.2.2 eval_indexed()	1032
6.157.2.3 contract_with()	1033
6.157.2.4 do_print()	1033
6.157.2.5 do_print_latex()	1033
6.158 GiNaC::spmapkey Class Reference	1033
6.158.1 Constructor & Destructor Documentation	1034
6.158.1.1 spmapkey() [1/2]	1034
6.158.1.2 spmapkey() [2/2]	1035
6.158.2 Member Function Documentation	1035
6.158.2.1 operator==()	1035
6.158.2.2 operator<()	1035
6.158.2.3 debugprint()	1035
6.158.3 Member Data Documentation	1035
6.158.3.1 v1	1035
6.158.3.2 v2	1035
6.158.3.3 dim	1036
6.159 GiNaC::status_flags Class Reference	1036
6.159.1 Detailed Description	1036
6.159.2 Member Enumeration Documentation	1036
6.159.2.1 anonymous enum	1036
6.160 GiNaC::structure< T, ComparisonPolicy > Class Template Reference	1037
6.160.1 Detailed Description	1041
6.160.2 Constructor & Destructor Documentation	1041
6.160.2.1 structure()	1041
6.160.3 Member Function Documentation	1041
6.160.3.1 get_class_name()	1041
6.160.3.2 eval()	1042
6.160.3.3 evalm()	1042
6.160.3.4 eval_ncmul()	1042
6.160.3.5 eval_indexed()	1042
6.160.3.6 print()	1042
6.160.3.7 precedence()	1043
6.160.3.8 info()	1043
6.160.3.9 nops()	1043
6.160.3.10 op()	1043
6.160.3.11 operator[]() [1/4]	1044
6.160.3.12 operator[]() [2/4]	1044
6.160.3.13 let_op()	1044
6.160.3.14 operator[]() [3/4]	1044
6.160.3.15 operator[]() [4/4]	1044
6.160.3.16 has()	1044

6.160.3.17	match()	1045
6.160.3.18	match_same_type()	1045
6.160.3.19	subs()	1045
6.160.3.20	map()	1046
6.160.3.21	degree()	1046
6.160.3.22	ldegree()	1046
6.160.3.23	coeff()	1046
6.160.3.24	expand()	1046
6.160.3.25	collect()	1047
6.160.3.26	derivative()	1048
6.160.3.27	series()	1048
6.160.3.28	normal()	1049
6.160.3.29	to_rational()	1049
6.160.3.30	to_polynomial()	1049
6.160.3.31	integer_content()	1049
6.160.3.32	smod()	1049
6.160.3.33	max_coefficient()	1050
6.160.3.34	get_free_indices()	1050
6.160.3.35	add_indexed()	1050
6.160.3.36	scalar_mul_indexed()	1051
6.160.3.37	contract_with()	1051
6.160.3.38	return_type()	1052
6.160.3.39	return_type_tinfo()	1052
6.160.3.40	is_equal_same_type()	1053
6.160.3.41	calchash()	1053
6.160.3.42	operator->()	1053
6.160.3.43	get_struct() [1/2]	1053
6.160.3.44	get_struct() [2/2]	1053
6.160.4	Member Data Documentation	1054
6.160.4.1	obj	1054
6.161	GiNaC::su3d Class Reference	1054
6.161.1	Detailed Description	1058
6.161.2	Member Function Documentation	1059
6.161.2.1	eval_indexed()	1059
6.161.2.2	contract_with()	1059
6.161.2.3	return_type()	1059
6.161.2.4	do_print()	1059
6.161.2.5	do_print_latex()	1059
6.162	GiNaC::su3f Class Reference	1060
6.162.1	Detailed Description	1064
6.162.2	Member Function Documentation	1064
6.162.2.1	eval_indexed()	1064

6.162.2.2 contract_with()	1064
6.162.2.3 return_type()	1065
6.162.2.4 do_print()	1065
6.162.2.5 do_print_latex()	1065
6.163 GiNaC::su3one Class Reference	1065
6.163.1 Detailed Description	1069
6.163.2 Member Function Documentation	1069
6.163.2.1 do_print()	1069
6.163.2.2 do_print_latex()	1070
6.164 GiNaC::su3t Class Reference	1070
6.164.1 Detailed Description	1074
6.164.2 Member Function Documentation	1075
6.164.2.1 contract_with()	1075
6.164.2.2 do_print()	1075
6.164.2.3 do_print_latex()	1075
6.165 GiNaC::subs_options Class Reference	1075
6.165.1 Detailed Description	1075
6.165.2 Member Enumeration Documentation	1075
6.165.2.1 anonymous enum	1075
6.166 GiNaC::sy_is_less Class Reference	1076
6.166.1 Constructor & Destructor Documentation	1076
6.166.1.1 sy_is_less()	1076
6.166.2 Member Function Documentation	1076
6.166.2.1 operator>()	1076
6.166.3 Member Data Documentation	1077
6.166.3.1 v	1077
6.167 GiNaC::sy_swap Class Reference	1077
6.167.1 Constructor & Destructor Documentation	1077
6.167.1.1 sy_swap()	1077
6.167.2 Member Function Documentation	1077
6.167.2.1 operator>()	1077
6.167.3 Member Data Documentation	1078
6.167.3.1 v	1078
6.167.3.2 swapped	1078
6.168 GiNaC::sym_desc Struct Reference	1078
6.168.1 Detailed Description	1079
6.168.2 Constructor & Destructor Documentation	1079
6.168.2.1 sym_desc()	1079
6.168.3 Member Function Documentation	1080
6.168.3.1 operator<()	1080
6.168.4 Member Data Documentation	1080
6.168.4.1 sym	1080

6.168.4.2 deg_a	1080
6.168.4.3 deg_b	1080
6.168.4.4 ldeg_a	1080
6.168.4.5 ldeg_b	1080
6.168.4.6 max_deg	1081
6.168.4.7 max_lcnops	1081
6.169 GiNaC::symbol Class Reference	1081
6.169.1 Detailed Description	1085
6.169.2 Constructor & Destructor Documentation	1085
6.169.2.1 symbol() [1/2]	1085
6.169.2.2 symbol() [2/2]	1086
6.169.3 Member Function Documentation	1086
6.169.3.1 info()	1086
6.169.3.2 eval()	1086
6.169.3.3 evalf()	1086
6.169.3.4 series()	1087
6.169.3.5 subs()	1087
6.169.3.6 normal()	1087
6.169.3.7 to_rational()	1088
6.169.3.8 to_polynomial()	1088
6.169.3.9 conjugate()	1088
6.169.3.10 real_part()	1088
6.169.3.11 imag_part()	1088
6.169.3.12 is_polynomial()	1088
6.169.3.13 archive()	1089
6.169.3.14 read_archive()	1089
6.169.3.15 derivative()	1089
6.169.3.16 is_equal_same_type()	1090
6.169.3.17 calchash()	1090
6.169.3.18 get_domain()	1090
6.169.3.19 set_name()	1090
6.169.3.20 set_TeX_name()	1091
6.169.3.21 get_name()	1091
6.169.3.22 get_TeX_name()	1091
6.169.3.23 do_print()	1091
6.169.3.24 do_print_latex()	1091
6.169.3.25 do_print_tree()	1091
6.169.3.26 do_print_python_repr()	1092
6.169.4 Member Data Documentation	1092
6.169.4.1 serial	1092
6.169.4.2 name	1092
6.169.4.3 TeX_name	1092

6.169.4.4 next_serial	1092
6.170 GiNaC::symbolset Class Reference	1093
6.170.1 Constructor & Destructor Documentation	1093
6.170.1.1 symbolset()	1093
6.170.2 Member Function Documentation	1093
6.170.2.1 insert_symbols()	1093
6.170.2.2 has()	1093
6.170.3 Member Data Documentation	1094
6.170.3.1 s	1094
6.171 GiNaC::symmetry Class Reference	1094
6.171.1 Detailed Description	1099
6.171.2 Member Enumeration Documentation	1099
6.171.2.1 symmetry_type	1099
6.171.3 Constructor & Destructor Documentation	1099
6.171.3.1 symmetry() [1/2]	1099
6.171.3.2 symmetry() [2/2]	1099
6.171.4 Member Function Documentation	1100
6.171.4.1 archive()	1100
6.171.4.2 read_archive()	1100
6.171.4.3 calchash()	1100
6.171.4.4 get_type()	1101
6.171.4.5 set_type()	1101
6.171.4.6 add()	1101
6.171.4.7 validate()	1101
6.171.4.8 has_symmetry()	1101
6.171.4.9 has_nonsymmetric()	1102
6.171.4.10 has_cyclic()	1102
6.171.4.11 do_print()	1102
6.171.4.12 do_print_tree()	1102
6.171.5 Friends And Related Symbol Documentation	1102
6.171.5.1 sy_is_less	1102
6.171.5.2 sy_swap	1102
6.171.5.3 canonicalize	1102
6.171.6 Member Data Documentation	1103
6.171.6.1 type	1103
6.171.6.2 indices	1103
6.171.6.3 children	1103
6.172 GiNaC::symminfo Class Reference	1104
6.172.1 Detailed Description	1105
6.172.2 Constructor & Destructor Documentation	1105
6.172.2.1 symminfo() [1/2]	1105
6.172.2.2 symminfo() [2/2]	1105

6.172.3 Member Data Documentation	1105
6.172.3.1 symmterm	1105
6.172.3.2 coeff	1105
6.172.3.3 orig	1105
6.172.3.4 num	1106
6.173 GiNaC::symminfo_is_less_by_orig Class Reference	1106
6.173.1 Member Function Documentation	1106
6.173.1.1 operator()	1106
6.174 GiNaC::symminfo_is_less_by_symmterm Class Reference	1106
6.174.1 Member Function Documentation	1106
6.174.1.1 operator()	1106
6.175 GiNaC::tensdelta Class Reference	1107
6.175.1 Detailed Description	1111
6.175.2 Member Function Documentation	1111
6.175.2.1 info()	1111
6.175.2.2 eval_indexed()	1111
6.175.2.3 contract_with()	1112
6.175.2.4 return_type()	1112
6.175.2.5 do_print()	1112
6.175.2.6 do_print_latex()	1112
6.176 GiNaC::tensepsilon Class Reference	1113
6.176.1 Detailed Description	1117
6.176.2 Constructor & Destructor Documentation	1117
6.176.2.1 tensepsilon()	1117
6.176.3 Member Function Documentation	1118
6.176.3.1 info()	1118
6.176.3.2 eval_indexed()	1118
6.176.3.3 contract_with()	1118
6.176.3.4 archive()	1118
6.176.3.5 read_archive()	1119
6.176.3.6 return_type()	1119
6.176.3.7 do_print()	1119
6.176.3.8 do_print_latex()	1119
6.176.4 Member Data Documentation	1119
6.176.4.1 minkowski	1119
6.176.4.2 pos_sig	1120
6.177 GiNaC::tensmetric Class Reference	1120
6.177.1 Detailed Description	1124
6.177.2 Member Function Documentation	1125
6.177.2.1 info()	1125
6.177.2.2 eval_indexed()	1125
6.177.2.3 contract_with()	1125

6.177.2.4 return_type()	1125
6.177.2.5 do_print()	1126
6.178 GiNaC::tensor Class Reference	1126
6.178.1 Detailed Description	1130
6.178.2 Member Function Documentation	1130
6.178.2.1 return_type()	1130
6.178.2.2 replace_contr_index()	1131
6.179 GiNaC::terminfo Class Reference	1131
6.179.1 Detailed Description	1132
6.179.2 Constructor & Destructor Documentation	1132
6.179.2.1 terminfo()	1132
6.179.3 Member Data Documentation	1132
6.179.3.1 orig	1132
6.179.3.2 symm	1132
6.180 GiNaC::terminfo_is_less Class Reference	1133
6.180.1 Member Function Documentation	1133
6.180.1.1 operator()	1133
6.181 GiNaC::class_info< OPT >::tree_node Struct Reference	1133
6.181.1 Constructor & Destructor Documentation	1134
6.181.1.1 tree_node()	1134
6.181.2 Member Function Documentation	1134
6.181.2.1 add_child()	1134
6.181.3 Member Data Documentation	1134
6.181.3.1 children	1134
6.181.3.2 info	1134
6.182 GiNaC::unarchive_table_t Class Reference	1134
6.182.1 Constructor & Destructor Documentation	1135
6.182.1.1 unarchive_table_t()	1135
6.182.1.2 ~unarchive_table_t()	1135
6.182.2 Member Function Documentation	1135
6.182.2.1 find()	1135
6.182.2.2 insert()	1135
6.182.3 Member Data Documentation	1135
6.182.3.1 usecount	1135
6.182.3.2 unarch_map	1136
6.183 GiNaC::user_defined_kernel Class Reference	1136
6.183.1 Detailed Description	1141
6.183.2 Constructor & Destructor Documentation	1142
6.183.2.1 user_defined_kernel()	1142
6.183.3 Member Function Documentation	1142
6.183.3.1 nops()	1142
6.183.3.2 op()	1142

6.183.3.3 let_op()	1142
6.183.3.4 is_numeric()	1142
6.183.3.5 Laurent_series()	1143
6.183.3.6 uses_Laurent_series()	1143
6.183.3.7 do_print()	1143
6.183.4 Member Data Documentation	1143
6.183.4.1 f	1143
6.183.4.2 x	1143
6.184 GiNaC::varidx Class Reference	1144
6.184.1 Detailed Description	1149
6.184.2 Constructor & Destructor Documentation	1149
6.184.2.1 varidx()	1149
6.184.3 Member Function Documentation	1150
6.184.3.1 is_dummy_pair_same_type()	1150
6.184.3.2 archive()	1150
6.184.3.3 read_archive()	1150
6.184.3.4 match_same_type()	1151
6.184.3.5 is_covariant()	1151
6.184.3.6 is_contravariant()	1151
6.184.3.7 toggle_variance()	1151
6.184.3.8 do_print()	1152
6.184.3.9 do_print_tree()	1152
6.184.4 Member Data Documentation	1152
6.184.4.1 covariant	1152
6.185 GiNaC::visitor Class Reference	1152
6.185.1 Detailed Description	1153
6.185.2 Constructor & Destructor Documentation	1153
6.185.2.1 ~visitor()	1153
6.186 GiNaC::wildcard Class Reference	1153
6.186.1 Detailed Description	1157
6.186.2 Constructor & Destructor Documentation	1157
6.186.2.1 wildcard()	1157
6.186.3 Member Function Documentation	1158
6.186.3.1 match()	1158
6.186.3.2 archive()	1158
6.186.3.3 read_archive()	1158
6.186.3.4 calchash()	1158
6.186.3.5 get_label()	1159
6.186.3.6 do_print()	1159
6.186.3.7 do_print_tree()	1159
6.186.3.8 do_print_python_repr()	1159
6.186.4 Member Data Documentation	1159

6.186.4.1 label	1159
6.187 GiNaC::zeta1_SERIAL Class Reference	1160
6.187.1 Detailed Description	1160
6.187.2 Member Data Documentation	1160
6.187.2.1 serial	1160
6.188 GiNaC::zeta2_SERIAL Class Reference	1160
6.188.1 Detailed Description	1161
6.188.2 Member Data Documentation	1161
6.188.2.1 serial	1161
7 File Documentation	1163
7.1 add.cpp File Reference	1163
7.1.1 Detailed Description	1163
7.2 add.h File Reference	1164
7.2.1 Detailed Description	1164
7.3 archive.cpp File Reference	1164
7.3.1 Detailed Description	1165
7.4 archive.h File Reference	1165
7.4.1 Detailed Description	1166
7.4.2 Macro Definition Documentation	1166
7.4.2.1 GINAC_DECLARE_UNARCHIVER	1166
7.4.2.2 GINAC_BIND_UNARCHIVER	1167
7.5 assertion.h File Reference	1167
7.5.1 Detailed Description	1168
7.5.2 Macro Definition Documentation	1168
7.5.2.1 GINAC_ASSERT	1168
7.6 basic.cpp File Reference	1169
7.6.1 Detailed Description	1170
7.7 basic.h File Reference	1170
7.7.1 Detailed Description	1171
7.8 class_info.h File Reference	1171
7.8.1 Detailed Description	1172
7.9 clifford.cpp File Reference	1172
7.9.1 Detailed Description	1174
7.10 clifford.h File Reference	1174
7.10.1 Detailed Description	1176
7.11 color.cpp File Reference	1176
7.11.1 Detailed Description	1177
7.11.2 Macro Definition Documentation	1177
7.11.2.1 TEST_PERMUTATION	1177
7.11.2.2 CMPINDICES	1178
7.12 color.h File Reference	1178

7.12.1 Detailed Description	1179
7.13 compiler.h File Reference	1179
7.13.1 Detailed Description	1179
7.13.2 Macro Definition Documentation	1179
7.13.2.1 unlikely	1179
7.13.2.2 likely	1179
7.13.2.3 attribute_deprecated	1180
7.14 constant.cpp File Reference	1180
7.14.1 Detailed Description	1180
7.15 constant.h File Reference	1181
7.15.1 Detailed Description	1181
7.16 container.h File Reference	1181
7.16.1 Detailed Description	1182
7.17 crc32.h File Reference	1182
7.17.1 Detailed Description	1182
7.18 ex.cpp File Reference	1183
7.18.1 Detailed Description	1183
7.19 ex.h File Reference	1183
7.19.1 Detailed Description	1185
7.20 excompiler.cpp File Reference	1185
7.20.1 Detailed Description	1186
7.21 excompiler.h File Reference	1186
7.21.1 Detailed Description	1187
7.22 expair.cpp File Reference	1187
7.22.1 Detailed Description	1187
7.23 expair.h File Reference	1188
7.23.1 Detailed Description	1188
7.24 expairseq.cpp File Reference	1188
7.24.1 Detailed Description	1189
7.25 expairseq.h File Reference	1189
7.25.1 Detailed Description	1190
7.26 exprseq.cpp File Reference	1190
7.26.1 Detailed Description	1190
7.27 exprseq.h File Reference	1190
7.27.1 Detailed Description	1190
7.28 factor.cpp File Reference	1191
7.28.1 Detailed Description	1191
7.28.2 Macro Definition Documentation	1192
7.28.2.1 DCOUT	1192
7.28.2.2 DCOUTVAR	1192
7.28.2.3 DCOUT2	1192
7.28.2.4 USE_SAME_DEGREE_FACTOR	1192

7.28.3 Variable Documentation	1192
7.28.3.1 value	1192
7.28.3.2 r	1192
7.28.3.3 c	1193
7.28.3.4 m	1193
7.28.3.5 lr	1194
7.28.3.6 cache	1194
7.28.3.7 factors	1194
7.28.3.8 one	1194
7.28.3.9 n	1195
7.28.3.10 len	1195
7.28.3.11 last	1195
7.28.3.12 k	1196
7.28.3.13 poly	1196
7.28.3.14 x	1196
7.28.3.15 evalpoint	1197
7.28.3.16 R	1197
7.28.3.17 syms_wox	1197
7.28.3.18 unit	1197
7.28.3.19 cont	1197
7.28.3.20 pp	1198
7.28.3.21 vn	1198
7.28.3.22 vnlst	1198
7.28.3.23 modulus	1198
7.28.3.24 syms	1198
7.28.3.25 options	1198
7.29 factor.h File Reference	1199
7.29.1 Detailed Description	1199
7.30 fail.cpp File Reference	1199
7.30.1 Detailed Description	1199
7.31 fail.h File Reference	1200
7.31.1 Detailed Description	1200
7.32 fderivative.cpp File Reference	1200
7.32.1 Detailed Description	1201
7.33 fderivative.h File Reference	1201
7.33.1 Detailed Description	1201
7.34 flags.h File Reference	1201
7.34.1 Detailed Description	1202
7.35 function.cpp File Reference	1202
7.35.1 Detailed Description	1203
7.36 function.h File Reference	1203
7.36.1 Detailed Description	1210

7.36.2 Macro Definition Documentation	1210
7.36.2.1 DECLARE_FUNCTION_1P	1210
7.36.2.2 DECLARE_FUNCTION_2P	1210
7.36.2.3 DECLARE_FUNCTION_3P	1210
7.36.2.4 DECLARE_FUNCTION_4P	1211
7.36.2.5 DECLARE_FUNCTION_5P	1211
7.36.2.6 DECLARE_FUNCTION_6P	1211
7.36.2.7 DECLARE_FUNCTION_7P	1211
7.36.2.8 DECLARE_FUNCTION_8P	1212
7.36.2.9 DECLARE_FUNCTION_9P	1212
7.36.2.10 DECLARE_FUNCTION_10P	1212
7.36.2.11 DECLARE_FUNCTION_11P	1212
7.36.2.12 DECLARE_FUNCTION_12P	1213
7.36.2.13 DECLARE_FUNCTION_13P	1213
7.36.2.14 DECLARE_FUNCTION_14P	1213
7.36.2.15 REGISTER_FUNCTION	1213
7.36.2.16 is_ex_the_function	1214
7.37 ginac.h File Reference	1214
7.37.1 Detailed Description	1215
7.38 hash_map.h File Reference	1215
7.38.1 Detailed Description	1215
7.39 hash_seed.h File Reference	1215
7.39.1 Detailed Description	1216
7.40 idx.cpp File Reference	1216
7.40.1 Detailed Description	1217
7.41 idx.h File Reference	1217
7.41.1 Detailed Description	1218
7.42 indexed.cpp File Reference	1218
7.42.1 Detailed Description	1219
7.43 indexed.h File Reference	1220
7.43.1 Detailed Description	1220
7.44 inifcns.cpp File Reference	1221
7.44.1 Detailed Description	1224
7.45 inifcns.h File Reference	1224
7.45.1 Detailed Description	1225
7.46 inifcns_elliptic.cpp File Reference	1225
7.46.1 Detailed Description	1226
7.47 inifcns_gamma.cpp File Reference	1227
7.47.1 Detailed Description	1228
7.48 inifcns_nstdsums.cpp File Reference	1228
7.48.1 Detailed Description	1229
7.49 inifcns_trans.cpp File Reference	1230

7.49.1 Detailed Description	1233
7.50 integral.cpp File Reference	1233
7.50.1 Detailed Description	1234
7.51 integral.h File Reference	1234
7.51.1 Detailed Description	1234
7.52 integration_kernel.cpp File Reference	1234
7.52.1 Detailed Description	1236
7.52.2 Variable Documentation	1236
7.52.2.1 qbar	1236
7.52.2.2 order	1236
7.52.2.3 cache_vec	1236
7.52.2.4 x	1236
7.53 integration_kernel.h File Reference	1236
7.53.1 Detailed Description	1238
7.54 lst.cpp File Reference	1238
7.54.1 Detailed Description	1238
7.55 lst.h File Reference	1238
7.55.1 Detailed Description	1239
7.56 matrix.cpp File Reference	1239
7.56.1 Detailed Description	1240
7.57 matrix.h File Reference	1240
7.57.1 Detailed Description	1241
7.58 mul.cpp File Reference	1241
7.58.1 Detailed Description	1242
7.59 mul.h File Reference	1242
7.59.1 Detailed Description	1243
7.60 ncmul.cpp File Reference	1243
7.60.1 Detailed Description	1243
7.61 ncmul.h File Reference	1244
7.61.1 Detailed Description	1244
7.62 normal.cpp File Reference	1244
7.62.1 Detailed Description	1246
7.62.2 Macro Definition Documentation	1246
7.62.2.1 FAST_COMPARE	1246
7.62.2.2 USE_REMEMBER	1247
7.62.2.3 USE_TRIAL_DIVISION	1247
7.62.2.4 STATISTICS	1247
7.63 normal.h File Reference	1247
7.63.1 Detailed Description	1248
7.64 numeric.cpp File Reference	1248
7.64.1 Detailed Description	1251
7.65 numeric.h File Reference	1251

7.65.1 Detailed Description	1254
7.66 operators.cpp File Reference	1254
7.66.1 Detailed Description	1256
7.67 operators.h File Reference	1256
7.67.1 Detailed Description	1257
7.68 power.cpp File Reference	1258
7.68.1 Detailed Description	1258
7.69 power.h File Reference	1258
7.69.1 Detailed Description	1259
7.70 print.cpp File Reference	1259
7.70.1 Detailed Description	1259
7.71 print.h File Reference	1260
7.71.1 Detailed Description	1261
7.71.2 Macro Definition Documentation	1261
7.71.2.1 GINAC_DECLARE_PRINT_CONTEXT_COMMON	1261
7.71.2.2 GINAC_DECLARE_PRINT_CONTEXT_BASE	1261
7.71.2.3 GINAC_DECLARE_PRINT_CONTEXT	1262
7.71.2.4 GINAC_IMPLEMENT_PRINT_CONTEXT	1262
7.72 pseries.cpp File Reference	1262
7.72.1 Detailed Description	1263
7.73 pseries.h File Reference	1263
7.73.1 Detailed Description	1263
7.74 ptr.h File Reference	1264
7.74.1 Detailed Description	1264
7.75 registrar.cpp File Reference	1264
7.75.1 Detailed Description	1264
7.76 registrar.h File Reference	1265
7.76.1 Detailed Description	1266
7.76.2 Macro Definition Documentation	1266
7.76.2.1 GINAC_DECLARE_REGISTERED_CLASS_COMMON	1266
7.76.2.2 GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS	1266
7.76.2.3 GINAC_DECLARE_REGISTERED_CLASS	1267
7.76.2.4 GINAC_IMPLEMENT_REGISTERED_CLASS	1267
7.76.2.5 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT	1267
7.76.2.6 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T	1268
7.77 relational.cpp File Reference	1268
7.77.1 Detailed Description	1268
7.78 relational.h File Reference	1268
7.78.1 Detailed Description	1269
7.79 remember.cpp File Reference	1269
7.79.1 Detailed Description	1269
7.80 remember.h File Reference	1269

7.80.1 Detailed Description	1270
7.81 structure.h File Reference	1270
7.81.1 Detailed Description	1270
7.82 symbol.cpp File Reference	1271
7.82.1 Detailed Description	1271
7.83 symbol.h File Reference	1271
7.83.1 Detailed Description	1272
7.84 symmetry.cpp File Reference	1272
7.84.1 Detailed Description	1273
7.85 symmetry.h File Reference	1273
7.85.1 Detailed Description	1274
7.86 tensor.cpp File Reference	1275
7.86.1 Detailed Description	1276
7.87 tensor.h File Reference	1276
7.87.1 Detailed Description	1277
7.88 utils.cpp File Reference	1277
7.88.1 Detailed Description	1279
7.89 utils.h File Reference	1279
7.89.1 Detailed Description	1280
7.89.2 Macro Definition Documentation	1281
7.89.2.1 DEFAULT_CTOR	1281
7.89.2.2 DEFAULT_COMPARE	1281
7.89.2.3 DEFAULT_PRINT	1281
7.89.2.4 DEFAULT_PRINT_LATEX	1281
7.90 utils_multi_iterator.h File Reference	1281
7.90.1 Detailed Description	1283
7.91 version.h File Reference	1283
7.91.1 Detailed Description	1283
7.91.2 Macro Definition Documentation	1284
7.91.2.1 GINACLIB_MAJOR_VERSION	1284
7.91.2.2 GINACLIB_MINOR_VERSION	1284
7.91.2.3 GINACLIB_MICRO_VERSION	1284
7.91.2.4 GINAC_LT_CURRENT	1284
7.91.2.5 GINAC_LT_REVISION	1284
7.91.2.6 GINAC_LT_AGE	1284
7.91.2.7 GINACLIB_ARCHIVE_VERSION	1284
7.91.2.8 GINACLIB_ARCHIVE_AGE	1284
7.91.2.9 GINACLIB_STR_HELPER	1284
7.91.2.10 GINACLIB_STR	1284
7.91.2.11 GINACLIB_VERSION	1285
7.92 wildcard.cpp File Reference	1285
7.92.1 Detailed Description	1285

7.93 wildcard.h File Reference	1285
7.93.1 Detailed Description	1286

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

GiNaC	19
GiNaC::internal	272
std	272

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

GiNaC::internal::_iter_rep	273
GiNaC::_numeric_digits	275
GiNaC::archive	294
GiNaC::archive_node	302
GiNaC::archive_node::archive_node_cit_range	312
GiNaC::archive::archived_ex	314
GiNaC::basic_multi_iterator< T >	346
GiNaC::multi_iterator_counter< T >	794
GiNaC::multi_iterator_counter_indv< T >	797
GiNaC::multi_iterator_ordered< T >	801
GiNaC::multi_iterator_ordered_eq< T >	805
GiNaC::multi_iterator_ordered_eq_indv< T >	808
GiNaC::multi_iterator_permutation< T >	812
GiNaC::multi_iterator_shuffle< T >	816
GiNaC::multi_iterator_shuffle_prime< T >	820
GiNaC::basic_partition_generator	352
GiNaC::partition_generator	873
GiNaC::partition_with_zero_parts_generator	876
GiNaC::class_info< OPT >	353
GiNaC::compare_all_equal< T >	388
GiNaC::compare_bitwise< T >	389
GiNaC::compare_std_less< T >	390
ComparisonPolicy	
GiNaC::structure< T, ComparisonPolicy >	1037
GiNaC::composition_generator	391
GiNaC::const_iterator	394
GiNaC::const_postorder_iterator	400
GiNaC::const_preorder_iterator	403
GiNaC::container_storage< C >	432
GiNaC::container< C >	416
GiNaC::function	589
GiNaC::fderivative	576
GiNaC::indexed	669
GiNaC::clifford	357

GiNaC::color	376
GiNaC::ncmul	831
GiNaC::composition_generator::coolmulti	435
GiNaC::determinant_algo	439
GiNaC::do_taylor	466
GiNaC::domain	466
std::domain_error	
GiNaC::pole_error	899
GiNaC::dunno	467
GiNaC::composition_generator::coolmulti::element	494
std::equal_to< GiNaC::ex >	504
GiNaC::error_and_integral	505
GiNaC::error_and_integral_is_less	506
GiNaC::ex	511
GiNaC::ex_base_is_less	543
GiNaC::ex_is_equal	543
GiNaC::ex_is_less	543
GiNaC::ex_swap	544
GiNaC::expair	545
GiNaC::expair_is_less	548
GiNaC::expair_rest_is_less	549
GiNaC::expair_swap	550
GiNaC::expand_options	570
GiNaC::factor_options	571
GiNaC::function_options	609
GiNaC::G2_SERIAL	650
GiNaC::G3_SERIAL	651
GiNaC::gcd_options	652
GiNaC::gcdheu_failed	653
GiNaC::has_distance< T >	653
GiNaC::has_options	655
std::hash< GiNaC::ex >	655
GiNaC::idx_is_equal_ignore_dim	668
GiNaC::info_flags	687
GiNaC::is_not_a_clifford	708
GiNaC::is_summation_idx	708
GiNaC::iterated_integral2_SERIAL	709
GiNaC::iterated_integral3_SERIAL	710
GiNaC::lanczos_coeffs	727
std::less< GiNaC::ptr< T > >	729
GiNaC::library_init	729
std::list	
GiNaC::remember_table_list	1008
GiNaC::make_flat_inserter	731
GiNaC::map_function	733
GiNaC::derivative_map_function	437
GiNaC::eval_integ_map_function	507
GiNaC::evalf_map_function	508
GiNaC::evalm_map_function	510
GiNaC::expand_map_function	568
GiNaC::normal_map_function	844
GiNaC::pointer_to_map_function	878
GiNaC::pointer_to_map_function_1arg< T1 >	880
GiNaC::pointer_to_map_function_2args< T1, T2 >	883
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >	885
GiNaC::pointer_to_member_to_map_function< C >	888
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >	891
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >	893

GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >	896
GiNaC::basic_partition_generator::mpartition2	773
GiNaC::op0_is_equal	873
GiNaC::print_context	923
GiNaC::print_csrc	927
GiNaC::print_csrc_cl_N	928
GiNaC::print_csrc_double	930
GiNaC::print_csrc_float	932
GiNaC::print_dflt	934
GiNaC::print_latex	939
GiNaC::print_python	946
GiNaC::print_python_repr	948
GiNaC::print_tree	949
GiNaC::print_context_options	925
GiNaC::print_functor	935
GiNaC::print_functor_impl	938
GiNaC::print_memfun_handler< T, C >	941
GiNaC::print_ptrfun_handler< T, C >	943
GiNaC::print_options	943
GiNaC::archive_node::property	951
GiNaC::archive_node::property_info	952
GiNaC::psi1_SERIAL	970
GiNaC::psi2_SERIAL	971
GiNaC::ptr< T >	972
GiNaC::ptr< GiNaC::basic >	972
GiNaC::refcounted	983
GiNaC::basic	315
GiNaC::constant	406
GiNaC::container< C >	416
GiNaC::expairseq	550
GiNaC::add	278
GiNaC::mul	775
GiNaC::fail	571
GiNaC::idx	656
GiNaC::varidx	1144
GiNaC::spinidx	1016
GiNaC::integral	689
GiNaC::integration_kernel	700
GiNaC::ELi_kernel	496
GiNaC::Ebar_kernel	467
GiNaC::Eisenstein_h_kernel	475
GiNaC::Eisenstein_kernel	485
GiNaC::Kronecker_dtau_kernel	710
GiNaC::Kronecker_dz_kernel	719
GiNaC::basic_log_kernel	341
GiNaC::modular_form_kernel	765
GiNaC::multiple_polylog_kernel	824
GiNaC::user_defined_kernel	1136
GiNaC::matrix	736
GiNaC::numeric	846
GiNaC::power	907
GiNaC::pseries	954
GiNaC::relational	989
GiNaC::structure< T, ComparisonPolicy >	1037
GiNaC::symbol	1081
GiNaC::realsymbol	977
GiNaC::possymbol	901

GiNaC::symmetry	1094
GiNaC::tensor	1126
GiNaC::cliffordunit	371
GiNaC::diracgamma	440
GiNaC::diracgamma5	445
GiNaC::diracgammaL	451
GiNaC::diracgammaR	456
GiNaC::diracone	461
GiNaC::su3d	1054
GiNaC::su3f	1060
GiNaC::su3one	1065
GiNaC::su3t	1070
GiNaC::tensdelta	1107
GiNaC::tensepsilon	1113
GiNaC::tensmetric	1120
GiNaC::minkmetric	757
GiNaC::spinmetric	1027
GiNaC::wildcard	1153
GiNaC::registered_class_options	986
GiNaC::remember_strategies	1000
GiNaC::remember_table_entry	1005
GiNaC::return_type_t	1010
GiNaC::return_types	1011
GiNaC::relational::safe_bool_helper	1012
GiNaC::scalar_products	1012
GiNaC::series_options	1015
GiNaC::solve_algo	1015
GiNaC::spmapkey	1033
GiNaC::status_flags	1036
GiNaC::subs_options	1075
GiNaC::sy_is_less	1076
GiNaC::sy_swap	1077
GiNaC::sym_desc	1078
GiNaC::symbolset	1093
GiNaC::symminfo	1104
GiNaC::symminfo_is_less_by_orig	1106
GiNaC::symminfo_is_less_by_symmterm	1106
GiNaC::terminfo	1131
GiNaC::terminfo_is_less	1133
GiNaC::class_info< OPT >::tree_node	1133
GiNaC::unarchive_table_t	1134
std::vector	
GiNaC::remember_table	1001
GiNaC::visitor	1152
GiNaC::zeta1_SERIAL	1160
GiNaC::zeta2_SERIAL	1160

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

GiNaC::internal::_iter_rep	273
GiNaC::_numeric_digits	
This class is used to instantiate a global singleton object Digits which behaves just like Maple's	
Digits	275
GiNaC::add	
Sum of expressions	278
GiNaC::archive	
This class holds archived versions of GiNaC expressions (class ex)	294
GiNaC::archive_node	
This class stores all properties needed to record/retrieve the state of one object of class basic	
(or a derived class)	302
GiNaC::archive_node::archive_node_cit_range	312
GiNaC::archive::archived_ex	
Archived expression descriptor	314
GiNaC::basic	
This class is the ABC (abstract base class) of GiNaC 's class hierarchy	315
GiNaC::basic_log_kernel	
The basic integration kernel with a logarithmic singularity at the origin	341
GiNaC::basic_multi_iterator< T >	
Basic_multi_iterator is a base class	346
GiNaC::basic_partition_generator	
Base class for generating all bounded combinatorial partitions of an integer n with exactly m	
parts in non-decreasing order	352
GiNaC::class_info< OPT >	353
GiNaC::clifford	
This class holds an object representing an element of the Clifford algebra (the Dirac gamma	
matrices)	357
GiNaC::cliffordunit	
This class represents the Clifford algebra generators (units)	371
GiNaC::color	
This class holds a generator T_a or the unity element of the Lie algebra of SU(3), as used for	
calculations in quantum chromodynamics	376
GiNaC::compare_all_equal< T >	
Comparison policy: all structures of one type are equal	388
GiNaC::compare_bitwise< T >	
Comparison policy: use bit-wise comparison to compare structures	389

GiNaC::compare_std_less< T >	
Comparison policy: use <code>std::equal_to</code> / <code>std::less</code> (defaults to operators <code>==</code> and <code><</code>) to compare structures	390
GiNaC::composition_generator	
Generate all compositions of a partition of an integer n , starting with the compositions which has non-decreasing order	391
GiNaC::const_iterator	394
GiNaC::const_postorder_iterator	400
GiNaC::const_preorder_iterator	403
GiNaC::constant	
This class holds constants, symbols with specific numerical value	406
GiNaC::container< C >	
Wrapper template for making GiNaC classes out of STL containers	416
GiNaC::container_storage< C >	
Helper template for encapsulating the <code>reserve()</code> mechanics of STL containers	432
GiNaC::composition_generator::coolmulti	435
GiNaC::derivative_map_function	
Function object to be applied by <code>basic::derivative()</code>	437
GiNaC::determinant_algo	
Switch to control algorithm for determinant computation	439
GiNaC::diracgamma	
This class represents the Dirac gamma Lorentz vector	440
GiNaC::diracgamma5	
This class represents the Dirac gamma5 object which anticommutes with all other gammas	445
GiNaC::diracgammaL	
This class represents the Dirac gammaL object which behaves like $1/2 (1-\text{gamma}5)$	451
GiNaC::diracgammaR	
This class represents the Dirac gammaL object which behaves like $1/2 (1+\text{gamma}5)$	456
GiNaC::diracone	
This class represents the Clifford algebra unity element	461
GiNaC::do_taylor	
Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe	466
GiNaC::domain	
Domain of an object	466
GiNaC::dunno	
Exception class thrown by functions to signal unimplemented functionality so the expression may just be <code>.hold()</code>	467
GiNaC::Ebar_kernel	
The Ebar-kernel	467
GiNaC::Eisenstein_h_kernel	
The kernel corresponding to the Eisenstein series $h_{k,N,r,s}(\tau)$	475
GiNaC::Eisenstein_kernel	
The kernel corresponding to the Eisenstein series $E_{k,N,a,b,K}(\tau)$	485
GiNaC::composition_generator::coolmulti::element	494
GiNaC::ELi_kernel	
The ELi-kernel	496
std::equal_to< GiNaC::ex >	
Specialization of <code>std::equal_to()</code> for <code>ex</code> objects	504
GiNaC::error_and_integral	505
GiNaC::error_and_integral_is_less	506
GiNaC::eval_integ_map_function	
Function object to be applied by <code>basic::eval_integ()</code>	507
GiNaC::evalf_map_function	
Function object to be applied by <code>basic::evalf()</code>	508
GiNaC::evalm_map_function	
Function object to be applied by <code>basic::evalm()</code>	510

GiNaC::ex	
Lightweight wrapper for GiNaC 's symbolic objects	511
GiNaC::ex_base_is_less	543
GiNaC::ex_is_equal	543
GiNaC::ex_is_less	543
GiNaC::ex_swap	544
GiNaC::expair	
A pair of expressions	545
GiNaC::expair_is_less	
Function object for insertion into third argument of STL's sort() etc	548
GiNaC::expair_rest_is_less	
Function object not caring about the numerical coefficients for insertion into third argument of STL's sort()	549
GiNaC::expair_swap	550
GiNaC::expairseq	
A sequence of class expair	550
GiNaC::expand_map_function	
Function object to be applied by basic::expand()	568
GiNaC::expand_options	
Flags to control the behavior of expand()	570
GiNaC::factor_options	
Flags to control the polynomial factorization	571
GiNaC::fail	571
GiNaC::fderivative	
This class represents the (abstract) derivative of a symbolic function	576
GiNaC::function	
The class function is used to implement builtin functions like sin, cos... and user defined functions	589
GiNaC::function_options	609
GiNaC::G2_SERIAL	
Generalized multiple polylogarithm	650
GiNaC::G3_SERIAL	
Generalized multiple polylogarithm with explicit imaginary parts	651
GiNaC::gcd_options	
Flags to control the behavior of gcd() and friends	652
GiNaC::gcdheu_failed	
Exception thrown by heur_gcd() to signal failure	653
GiNaC::has_distance< T >	
SFINAE test for distance	653
GiNaC::has_options	
Flags to control the behavior of has()	655
std::hash< GiNaC::ex >	
Specialization of std::hash() for ex objects	655
GiNaC::idx	
This class holds one index of an indexed object	656
GiNaC::idx_is_equal_ignore_dim	668
GiNaC::indexed	
This class holds an indexed expression	669
GiNaC::info_flags	
Possible attributes an object can have	687
GiNaC::integral	
Symbolic integral	689
GiNaC::integration_kernel	
The base class for integration kernels for iterated integrals	700
GiNaC::is_not_a_clifford	
Predicate for finding non-clifford objects	708
GiNaC::is_summation_idx	708
GiNaC::iterated_integral2_SERIAL	
Complete elliptic integral of the first kind	709

GiNaC::iterated_integral3_SERIAL	
Iterated integral with explicit truncation	710
GiNaC::Kronecker_dtau_kernel	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n)}(z_j, K\tau)$ in τ (or equivalently in \bar{q})	710
GiNaC::Kronecker_dz_kernel	
The kernel corresponding to integrating the Kronecker coefficient function $g^{(n-1)}(z - z_j, K\tau)$ in z	719
GiNaC::lanczos_coeffs	727
std::less< GiNaC::ptr< T > >	
Specialization of <code>std::less</code> for <code>ptr<T></code> to enable ordering of <code>ptr<T></code> objects (e.g	729
GiNaC::library_init	
Helper class to initialize the library	729
GiNaC::make_flat_inserter	
Class to handle the renaming of dummy indices	731
GiNaC::map_function	
Function object for <code>map()</code>	733
GiNaC::matrix	
Symbolic matrices	736
GiNaC::minkmetric	
This class represents a Minkowski metric tensor	757
GiNaC::modular_form_kernel	
A kernel corresponding to a polynomial in Eisenstein series	765
GiNaC::basic_partition_generator::mpartition2	773
GiNaC::mul	
Product of expressions	775
GiNaC::multi_iterator_counter< T >	
The class <code>multi_iterator_counter</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , such that	794
GiNaC::multi_iterator_counter_indv< T >	
The class <code>multi_iterator_counter_indv</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , such that	797
GiNaC::multi_iterator_ordered< T >	
The class <code>multi_iterator_ordered</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , such that	801
GiNaC::multi_iterator_ordered_eq< T >	
The class <code>multi_iterator_ordered_eq</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , such that	805
GiNaC::multi_iterator_ordered_eq_indv< T >	
The class <code>multi_iterator_ordered_eq_indv</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , such that	808
GiNaC::multi_iterator_permutation< T >	
The class <code>multi_iterator_permutation</code> defines a <code>multi_iterator</code> (i_1, i_2, \dots, i_k) , for which	812
GiNaC::multi_iterator_shuffle< T >	
The class <code>multi_iterator_shuffle</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b	816
GiNaC::multi_iterator_shuffle_prime< T >	
The class <code>multi_iterator_shuffle_prime</code> defines a <code>multi_iterator</code> , which runs over all shuffles of a and b, excluding the first one (a,b)	820
GiNaC::multiple_polylog_kernel	
The integration kernel for multiple polylogarithms	824
GiNaC::ncmul	
Non-commutative product of expressions	831
GiNaC::normal_map_function	
Function object to be applied by <code>basic::normal()</code>	844
GiNaC::numeric	
This class is a wrapper around CLN-numbers within the <code>GiNaC</code> class hierarchy	846
GiNaC::op0_is_equal	873
GiNaC::partition_generator	
Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order	873
GiNaC::partition_with_zero_parts_generator	
Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order	876

GiNaC::pointer_to_map_function	878
GiNaC::pointer_to_map_function_1arg< T1 >	880
GiNaC::pointer_to_map_function_2args< T1, T2 >	883
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >	885
GiNaC::pointer_to_member_to_map_function< C >	888
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >	891
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >	893
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >	896
GiNaC::pole_error	
Exception class thrown when a singularity is encountered	899
GiNaC::possymbol	
Specialization of symbol to real positive domain	901
GiNaC::power	
This class holds a two-component object, a basis and and exponent representing exponentiation	907
GiNaC::print_context	
Base class for print_contexts	923
GiNaC::print_context_options	
This class stores information about a registered print_context class	925
GiNaC::print_csrc	
Base context for C source output	927
GiNaC::print_csrc_cl_N	
Context for C source output using CLN numbers	928
GiNaC::print_csrc_double	
Context for C source output using double precision	930
GiNaC::print_csrc_float	
Context for C source output using float precision	932
GiNaC::print_dflt	
Context for default (ginsh-parsable) output	934
GiNaC::print_functor	
This class represents a print method for a certain algebraic class and print_context type	935
GiNaC::print_functor_impl	
Base class for print_functor handlers	938
GiNaC::print_latex	
Context for latex-parsable output	939
GiNaC::print_memfun_handler< T, C >	
Print_functor handler for member functions of class T, context type C	941
GiNaC::print_options	
Flags to control the behavior of a print_context	943
GiNaC::print_ptrfun_handler< T, C >	
Print_functor handler for pointer-to-functions of class T, context type C	943
GiNaC::print_python	
Context for python pretty-print output	946
GiNaC::print_python_repr	
Context for python-parsable output	948
GiNaC::print_tree	
Context for tree-like output for debugging	949
GiNaC::archive_node::property	
Archived property (data type, name and associated data)	951
GiNaC::archive_node::property_info	
Information about a stored property	952
GiNaC::pseries	
This class holds a extended truncated power series (positive and negative integer powers)	954
GiNaC::psi1_SERIAL	
Polylogarithm and multiple polylogarithm	970
GiNaC::psi2_SERIAL	
Derivatives of Psi-function (aka polygamma-functions)	971
GiNaC::ptr< T >	
Class of (intrusively) reference-counted pointers that support copy-on-write semantics	972

GiNaC::realsymbol	
Specialization of symbol to real domain	977
GiNaC::refcounted	
Base class for reference-counted objects	983
GiNaC::registered_class_options	
This class stores information about a registered GiNaC class	986
GiNaC::relational	
This class holds a relation consisting of two expressions and a logical relation between them	989
GiNaC::remember_strategies	
Strategies how to clean up the function remember cache	1000
GiNaC::remember_table	
The remember table is organized like an n-fold associative cache in a microprocessor	1001
GiNaC::remember_table_entry	
A single entry in the remember table of a function	1005
GiNaC::remember_table_list	
A list of entries in the remember table having some least significant bits of the hashvalue in common	1008
GiNaC::return_type_t	
To distinguish between different kinds of non-commutative objects	1010
GiNaC::return_types	1011
GiNaC::relational::safe_bool_helper	1012
GiNaC::scalar_products	
Helper class for storing information about known scalar products which are to be automatically replaced by simplify_indexed()	1012
GiNaC::series_options	
Flags to control series expansion	1015
GiNaC::solve_algo	
Switch to control algorithm for linear system solving	1015
GiNaC::spinidx	
This class holds a spinor index that can be dotted or undotted and that also has a variance	1016
GiNaC::spinmetric	
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors	1027
GiNaC::spmapkey	1033
GiNaC::status_flags	
Flags to store information about the state of an object	1036
GiNaC::structure< T, ComparisonPolicy >	
Wrapper template for making GiNaC classes out of C++ structures	1037
GiNaC::su3d	
This class represents the tensor of symmetric su(3) structure constants	1054
GiNaC::su3f	
This class represents the tensor of antisymmetric su(3) structure constants	1060
GiNaC::su3one	
This class represents the su(3) unity element	1065
GiNaC::su3t	
This class represents an su(3) generator	1070
GiNaC::subs_options	
Flags to control the behavior of subs()	1075
GiNaC::sy_is_less	1076
GiNaC::sy_swap	1077
GiNaC::sym_desc	
This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b"	1078
GiNaC::symbol	
Basic CAS symbol	1081
GiNaC::symbolset	1093
GiNaC::symmetry	
This class describes the symmetry of a group of indices	1094

GiNaC::symminfo	
This structure stores the individual symmetrized terms obtained during the simplification of sums	1104
GiNaC::symminfo_is_less_by_orig	1106
GiNaC::symminfo_is_less_by_symmterm	1106
GiNaC::tensdelta	
This class represents the delta tensor	1107
GiNaC::tensepsilon	
This class represents the totally antisymmetric epsilon tensor	1113
GiNaC::tensmetric	
This class represents a general metric tensor which can be used to raise/lower indices	1120
GiNaC::tensor	
This class holds one of GiNaC 's predefined special tensors such as the delta and the metric tensors	1126
GiNaC::terminfo	
This structure stores the original and symmetrized versions of terms obtained during the simplification of sums	1131
GiNaC::terminfo_is_less	1133
GiNaC::class_info< OPT >::tree_node	1133
GiNaC::unarchive_table_t	1134
GiNaC::user_defined_kernel	
A user-defined integration kernel	1136
GiNaC::varidx	
This class holds an index with a variance (co- or contravariant)	1144
GiNaC::visitor	
Degenerate base class for visitors	1152
GiNaC::wildcard	
This class acts as a wildcard for subs() , match() , has() and find()	1153
GiNaC::zeta1_SERIAL	
Complex conjugate	1160
GiNaC::zeta2_SERIAL	
Alternating Euler sum or colored MZV	1160

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

add.cpp	Implementation of GiNaC 's sums of expressions	1163
add.h	Interface to GiNaC 's sums of expressions	1164
archive.cpp	Archiving of GiNaC expressions	1164
archive.h	Archiving of GiNaC expressions	1165
assertion.h	Assertion macro definition	1167
basic.cpp	Implementation of GiNaC 's ABC	1169
basic.h	Interface to GiNaC 's ABC	1170
class_info.h	Helper templates to provide per-class information for class hierarchies	1171
clifford.cpp	Implementation of GiNaC 's clifford algebra (Dirac gamma) objects	1172
clifford.h	Interface to GiNaC 's clifford algebra (Dirac gamma) objects	1174
color.cpp	Implementation of GiNaC 's color (SU(3) Lie algebra) objects	1176
color.h	Interface to GiNaC 's color (SU(3) Lie algebra) objects	1178
compiler.h	Definition of optimizing macros	1179
constant.cpp	Implementation of GiNaC 's constant types and some special constants	1180
constant.h	Interface to GiNaC 's constant types and some special constants	1181
container.h	Wrapper template for making GiNaC classes out of STL containers	1181
crc32.h	CRC32 hash function	1182
ex.cpp	Implementation of GiNaC 's light-weight expression handles	1183

ex.h	Interface to GiNaC 's light-weight expression handles	1183
excompiler.cpp	Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration	1185
excompiler.h	Functions to facilitate the conversion of a ex to a function pointer suited for fast numerical integration	1186
expair.cpp	Implementation of expression pairs (building blocks of expairseq)	1187
expair.h	Definition of expression pairs (building blocks of expairseq)	1188
expairseq.cpp	Implementation of sequences of expression pairs	1188
expairseq.h	Interface to sequences of expression pairs	1189
exprseq.cpp	Implementation of GiNaC 's exprseq	1190
exprseq.h	Definition of GiNaC 's exprseq	1190
factor.cpp	Polynomial factorization (implementation)	1191
factor.h	Polynomial factorization	1199
fail.cpp	Implementation of class signaling failure of operation	1199
fail.h	Interface to class signaling failure of operation	1200
fderivative.cpp	Implementation of abstract derivatives of functions	1200
fderivative.h	Interface to abstract derivatives of functions	1201
flags.h	Collection of all flags used through the GiNaC framework	1201
function.cpp	Implementation of class of symbolic functions	1202
function.h	Interface to class of symbolic functions	1203
ginac.h	This include file includes all other public GiNaC headers	1214
hash_map.h	Replacement for <code>map<></code> using hash tables	1215
hash_seed.h	Type-specific hash seed	1215
idx.cpp	Implementation of GiNaC 's indices	1216
idx.h	Interface to GiNaC 's indices	1217
indexed.cpp	Implementation of GiNaC 's indexed expressions	1218
indexed.h	Interface to GiNaC 's indexed expressions	1220
inifcns.cpp	Implementation of GiNaC 's initially known functions	1221
inifcns.h	Interface to GiNaC 's initially known functions	1224
inifcns_elliptic.cpp	Implementation of some special functions related to elliptic curves	1225

inifcns_gamma.cpp	Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff	1227
inifcns_nstdsums.cpp	Implementation of some special functions that have a representation as nested sums	1228
inifcns_trans.cpp	Implementation of transcendental (and trigonometric and hyperbolic) functions	1230
integral.cpp	Implementation of GiNaC 's symbolic integral	1233
integral.h	Interface to GiNaC 's symbolic integral	1234
integration_kernel.cpp	Implementation of GiNaC 's integration kernels for iterated integrals	1234
integration_kernel.h	Interface to GiNaC 's integration kernels for iterated integrals	1236
lst.cpp	Implementation of GiNaC 's <code>lst</code>	1238
lst.h	Definition of GiNaC 's <code>lst</code>	1238
matrix.cpp	Implementation of symbolic matrices	1239
matrix.h	Interface to symbolic matrices	1240
mul.cpp	Implementation of GiNaC 's products of expressions	1241
mul.h	Interface to GiNaC 's products of expressions	1242
ncmul.cpp	Implementation of GiNaC 's non-commutative products of expressions	1243
ncmul.h	Interface to GiNaC 's non-commutative products of expressions	1244
normal.cpp	This file implements several functions that work on univariate and multivariate polynomials and rational functions	1244
normal.h	This file defines several functions that work on univariate and multivariate polynomials and rational functions	1247
numeric.cpp	This file contains the interface to the underlying bignum package	1248
numeric.h	Makes the interface to the underlying bignum package available	1251
operators.cpp	Implementation of GiNaC 's overloaded operators	1254
operators.h	Interface to GiNaC 's overloaded operators	1256
power.cpp	Implementation of GiNaC 's symbolic exponentiation ($\text{basis}^{\text{exponent}}$)	1258
power.h	Interface to GiNaC 's symbolic exponentiation ($\text{basis}^{\text{exponent}}$)	1258
print.cpp	Implementation of helper classes for expression output	1259
print.h	Definition of helper classes for expression output	1260
pseries.cpp	Implementation of class for extended truncated power series and methods for series expansion	1262
pseries.h	Interface to class for extended truncated power series	1263
ptr.h	Reference-counted pointer template	1264

registrar.cpp	
GiNaC's class registrar (for class basic and all classes derived from it)	1264
registrar.h	
GiNaC's class registrar (for class basic and all classes derived from it)	1265
relational.cpp	
Implementation of relations between expressions	1268
relational.h	
Interface to relations between expressions	1268
remember.cpp	
Implementation of helper classes for using the remember option in GiNaC functions	1269
remember.h	
Interface to helper classes for using the remember option in GiNaC functions	1269
structure.h	
Wrapper template for making GiNaC classes out of C++ structures	1270
symbol.cpp	
Implementation of GiNaC's symbolic objects	1271
symbol.h	
Interface to GiNaC's symbolic objects	1271
symmetry.cpp	
Implementation of GiNaC's symmetry definitions	1272
symmetry.h	
Interface to GiNaC's symmetry definitions	1273
tensor.cpp	
Implementation of GiNaC's special tensors	1275
tensor.h	
Interface to GiNaC's special tensors	1276
utils.cpp	
Implementation of several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1277
utils.h	
Interface to several small and furry utilities needed within GiNaC but not of any interest to the user of the library	1279
utils_multi_iterator.h	
Utilities for summing over multiple indices	1281
version.h	
GiNaC library version information	1283
wildcard.cpp	
Implementation of GiNaC's wildcard objects	1285
wildcard.h	
Interface to GiNaC's wildcard objects	1285

Chapter 5

Namespace Documentation

5.1 GiNaC Namespace Reference

Namespaces

- namespace [internal](#)

Classes

- class [_numeric_digits](#)
This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.
- class [add](#)
Sum of expressions.
- class [archive](#)
This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).
- class [archive_node](#)
This class stores all properties needed to record/retrieve the state of one object of class [basic](#) (or a derived class).
- class [basic](#)
This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.
- class [basic_log_kernel](#)
The basic integration kernel with a logarithmic singularity at the origin.
- class [basic_multi_iterator](#)
[basic_multi_iterator](#) is a base class.
- class [basic_partition_generator](#)
Base class for generating all bounded combinatorial partitions of an integer n with exactly m parts in non-decreasing order.
- class [class_info](#)
- class [clifford](#)
This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).
- class [cliffordunit](#)
This class represents the Clifford algebra generators (units).
- class [color](#)
This class holds a generator T_a or the unity element of the Lie algebra of $SU(3)$, as used for calculations in quantum chromodynamics.
- class [compare_all_equal](#)
Comparison policy: all structures of one type are equal.

- class [compare_bitwise](#)
Comparison policy: use bit-wise comparison to compare structures.
- class [compare_std_less](#)
Comparison policy: use `std::equal_to/std::less` (defaults to operators `==` and `<`) to compare structures.
- class [composition_generator](#)
Generate all compositions of a partition of an integer n , starting with the compositions which has non-decreasing order.
- class [const_iterator](#)
- class [const_postorder_iterator](#)
- class [const_preorder_iterator](#)
- class [constant](#)
This class holds constants, symbols with specific numerical value.
- class [container](#)
Wrapper template for making [GiNaC](#) classes out of STL containers.
- class [container_storage](#)
Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.
- struct [derivative_map_function](#)
Function object to be applied by `basic::derivative()`.
- class [determinant_algo](#)
Switch to control algorithm for determinant computation.
- class [diracgamma](#)
This class represents the Dirac gamma Lorentz vector.
- class [diracgamma5](#)
This class represents the Dirac gamma5 object which anticommutes with all other gammas.
- class [diracgammaL](#)
This class represents the Dirac gammaL object which behaves like $1/2 (1-\text{gamma}5)$.
- class [diracgammaR](#)
This class represents the Dirac gammaL object which behaves like $1/2 (1+\text{gamma}5)$.
- class [diracone](#)
This class represents the Clifford algebra unity element.
- class [do_taylor](#)
Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.
- class [domain](#)
Domain of an object.
- class [dunno](#)
Exception class thrown by functions to signal unimplemented functionality so the expression may just be `.hold()`
- class [Ebar_kernel](#)
The Ebar-kernel.
- class [Eisenstein_h_kernel](#)
The kernel corresponding to the Eisenstein series $h_{k,N,r,s}(\tau)$.
- class [Eisenstein_kernel](#)
The kernel corresponding to the Eisenstein series $E_{k,N,a,b,K}(\tau)$.
- class [ELi_kernel](#)
The ELi-kernel.
- struct [error_and_integral](#)
- struct [error_and_integral_is_less](#)
- struct [eval_integ_map_function](#)
Function object to be applied by `basic::eval_integ()`.
- struct [evalf_map_function](#)
Function object to be applied by `basic::evalf()`.

- struct [evalm_map_function](#)
Function object to be applied by [basic::evalm\(\)](#).
- class [ex](#)
Lightweight wrapper for [GiNaC](#)'s symbolic objects.
- struct [ex_base_is_less](#)
- struct [ex_is_equal](#)
- struct [ex_is_less](#)
- struct [ex_swap](#)
- class [expair](#)
A pair of expressions.
- struct [expair_is_less](#)
Function object for insertion into third argument of STL's [sort\(\)](#) etc.
- struct [expair_rest_is_less](#)
Function object not caring about the numerical coefficients for insertion into third argument of STL's [sort\(\)](#).
- struct [expair_swap](#)
- class [expairseq](#)
A sequence of class [expair](#).
- struct [expand_map_function](#)
Function object to be applied by [basic::expand\(\)](#).
- class [expand_options](#)
Flags to control the behavior of [expand\(\)](#).
- class [factor_options](#)
Flags to control the polynomial factorization.
- class [fail](#)
- class [fderivative](#)
This class represents the (abstract) derivative of a symbolic function.
- class [function](#)
The class [function](#) is used to implement builtin functions like [sin](#), [cos](#)... and user defined functions.
- class [function_options](#)
- class [G2_SERIAL](#)
Generalized multiple polylogarithm.
- class [G3_SERIAL](#)
Generalized multiple polylogarithm with explicit imaginary parts.
- struct [gcd_options](#)
Flags to control the behavior of [gcd\(\)](#) and friends.
- class [gcdheu_failed](#)
Exception thrown by [heur_gcd\(\)](#) to signal failure.
- class [has_distance](#)
SFINAE test for distance.
- class [has_options](#)
Flags to control the behavior of [has\(\)](#).
- class [idx](#)
This class holds one index of an indexed object.
- struct [idx_is_equal_ignore_dim](#)
- class [indexed](#)
This class holds an indexed expression.
- class [info_flags](#)
Possible attributes an object can have.
- class [integral](#)
Symbolic integral.
- class [integration_kernel](#)

- The base class for integration kernels for iterated integrals.

 - struct [is_not_a_clifford](#)

Predicate for finding non-clifford objects.
 - struct [is_summation_idx](#)
 - class [iterated_integral2_SERIAL](#)

Complete elliptic integral of the first kind.
 - class [iterated_integral3_SERIAL](#)

Iterated integral with explicit truncation.
 - class [Kronecker_dtau_kernel](#)

The kernel corresponding to integrating the Kronecker coefficient function $g^{(n)}(z_j, K\tau)$ in τ (or equivalently in \bar{q}).
 - class [Kronecker_dz_kernel](#)

The kernel corresponding to integrating the Kronecker coefficient function $g^{(n-1)}(z - z_j, K\tau)$ in z .
 - class [lanczos_coeffs](#)
 - class [library_init](#)

Helper class to initialize the library.
 - class [make_flat_inserter](#)

Class to handle the renaming of dummy indices.
 - struct [map_function](#)

Function object for `map()`.
 - class [matrix](#)

Symbolic matrices.
 - class [minkmetric](#)

This class represents a Minkowski metric tensor.
 - class [modular_form_kernel](#)

A kernel corresponding to a polynomial in Eisenstein series.
 - class [mul](#)

Product of expressions.
 - class [multi_iterator_counter](#)

The class [multi_iterator_counter](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , such that.
 - class [multi_iterator_counter_indv](#)

The class [multi_iterator_counter_indv](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , such that.
 - class [multi_iterator_ordered](#)

The class [multi_iterator_ordered](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , such that.
 - class [multi_iterator_ordered_eq](#)

The class [multi_iterator_ordered_eq](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , such that.
 - class [multi_iterator_ordered_eq_indv](#)

The class [multi_iterator_ordered_eq_indv](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , such that.
 - class [multi_iterator_permutation](#)

The class [multi_iterator_permutation](#) defines a `multi_iterator` (i_1, i_2, \dots, i_k) , for which.
 - class [multi_iterator_shuffle](#)

The class [multi_iterator_shuffle](#) defines a `multi_iterator`, which runs over all shuffles of a and b .
 - class [multi_iterator_shuffle_prime](#)

The class [multi_iterator_shuffle_prime](#) defines a `multi_iterator`, which runs over all shuffles of a and b , excluding the first one (a,b) .
 - class [multiple_polylog_kernel](#)

The integration kernel for multiple polylogarithms.
 - class [ncmul](#)

Non-commutative product of expressions.
 - struct [normal_map_function](#)

Function object to be applied by `basic::normal()`.
 - class [numeric](#)

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

- struct [op0_is_equal](#)
- class [partition_generator](#)
Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order.
- class [partition_with_zero_parts_generator](#)
Generate all bounded combinatorial partitions of an integer n with exactly m parts (including zero parts) in non-decreasing order.
- class [pointer_to_map_function](#)
- class [pointer_to_map_function_1arg](#)
- class [pointer_to_map_function_2args](#)
- class [pointer_to_map_function_3args](#)
- class [pointer_to_member_to_map_function](#)
- class [pointer_to_member_to_map_function_1arg](#)
- class [pointer_to_member_to_map_function_2args](#)
- class [pointer_to_member_to_map_function_3args](#)
- class [pole_error](#)
Exception class thrown when a singularity is encountered.
- class [possymbol](#)
Specialization of symbol to real positive domain.
- class [power](#)
This class holds a two-component object, a basis and an exponent representing exponentiation.
- class [print_context](#)
Base class for `print_contexts`.
- class [print_context_options](#)
This class stores information about a registered `print_context` class.
- class [print_csrc](#)
Base context for C source output.
- class [print_csrc_cl_N](#)
Context for C source output using CLN numbers.
- class [print_csrc_double](#)
Context for C source output using double precision.
- class [print_csrc_float](#)
Context for C source output using float precision.
- class [print_dflt](#)
Context for default (ginsh-parsable) output.
- class [print_functor](#)
This class represents a print method for a certain algebraic class and `print_context` type.
- class [print_functor_impl](#)
Base class for `print_functor` handlers.
- class [print_latex](#)
Context for latex-parsable output.
- class [print_memfun_handler](#)
`print_functor` handler for member functions of class T , context type C
- class [print_options](#)
Flags to control the behavior of a `print_context`.
- class [print_ptrfun_handler](#)
`print_functor` handler for pointer-to-functions of class T , context type C
- class [print_python](#)
Context for python pretty-print output.
- class [print_python_repr](#)
Context for python-parsable output.

- class [print_tree](#)
Context for tree-like output for debugging.
- class [pseries](#)
This class holds a extended truncated power series (positive and negative integer powers).
- class [psi1_SERIAL](#)
Polylogarithm and multiple polylogarithm.
- class [psi2_SERIAL](#)
Derivatives of Psi-function (aka polygamma-functions).
- class [ptr](#)
Class of (intrusively) reference-counted pointers that support copy-on-write semantics.
- class [realsymbol](#)
Specialization of symbol to real domain.
- class [refcounted](#)
Base class for reference-counted objects.
- class [registered_class_options](#)
This class stores information about a registered [GiNaC](#) class.
- class [relational](#)
This class holds a relation consisting of two expressions and a logical relation between them.
- class [remember_strategies](#)
Strategies how to clean up the function remember cache.
- class [remember_table](#)
The remember table is organized like an n-fold associative cache in a microprocessor.
- class [remember_table_entry](#)
A single entry in the remember table of a function.
- class [remember_table_list](#)
A list of entries in the remember table having some least significant bits of the hashvalue in common.
- struct [return_type_t](#)
To distinguish between different kinds of non-commutative objects.
- class [return_types](#)
- class [scalar_products](#)
Helper class for storing information about known scalar products which are to be automatically replaced by [simplify_indexed\(\)](#).
- class [series_options](#)
Flags to control series expansion.
- class [solve_algo](#)
Switch to control algorithm for linear system solving.
- class [spinidx](#)
This class holds a spinor index that can be dotted or undotted and that also has a variance.
- class [spinmetric](#)
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.
- class [smapkey](#)
- class [status_flags](#)
Flags to store information about the state of an object.
- class [structure](#)
Wrapper template for making [GiNaC](#) classes out of C++ structures.
- class [su3d](#)
This class represents the tensor of symmetric su(3) structure constants.
- class [su3f](#)
This class represents the tensor of antisymmetric su(3) structure constants.
- class [su3one](#)

This class represents the $su(3)$ unity element.

- class **su3t**

This class represents an $su(3)$ generator.

- class `subs_options`

Flags to control the behavior of `subs()`.

- class `sy_is_less`
- class `sy_swap`
- struct `sym_desc`

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b":

- class **symbol**

Basic CAS symbol.

- class `symbolset`
- class `symmetry`

This class describes the symmetry of a group of indices.

- class `symminfo`

This structure stores the individual symmetrized terms obtained during the simplification of sums.

- class `symminfo_is_less_by_orig`
- class `symminfo_is_less_by_symmterm`
- class `tensdelta`

This class represents the delta tensor.

- class **tensepsilon**

This class represents the totally antisymmetric epsilon tensor.

- class **tensmetric**

This class represents a general metric tensor which can be used to raise/lower indices.

- class **tensor**

This class holds one of GiNaC's predefined special tensors such as the delta and the metric tensors.

- class `terminfo`

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

- class `terminfo_is_less`
- class `unarchive_table_t`
- class `user defined kernel`

A user-defined integration kernel.

- class `varidx`

This class holds an index with a variance (co- or contravariant).

- class **visitor**

Degenerate base class for visitors.

- class wildcard

This class acts as a wildcard for `subs()`, `match()`, `has()` and `find()`.

- class `zeta1_SERIAL`

Complex conjugate.

- class `zeta2_SERIAL`

Alternating Euler sum or colored MZV.

Typedefs

- typedef unsigned archive_node_id

Numerical ID value to refer to an [archive node](#).

- typedef unsigned archive atom

Numerical ID value to refer to a string.

- typedef basic *(* synthesize_func) ()

- typedef std::map< std::string, [synthesize_func](#) > [unarchive_map_t](#)
- typedef std::vector< [ex](#) > [exvector](#)
- typedef std::set< [ex](#), [ex_is_less](#) > [exset](#)
- typedef std::map< [ex](#), [ex](#), [ex_is_less](#) > [exmap](#)
- typedef [ex](#)(* [evalffunc](#)) ()
- typedef double(* [FUNCP_1P](#)) (double)
Function pointer with one function parameter.
- typedef double(* [FUNCP_2P](#)) (double, double)
Function pointer with two function parameters.
- typedef void(* [FUNCP_CUBA](#)) (const int *, const double[], const int *, double[])
Function pointer for use with the CUBA library (<http://www.feynarts.de/cuba>).
- typedef std::vector< [expair](#) > [epvector](#)
expair-vector
- typedef [epvector](#)::iterator [epp](#)
expair-vector pointer
- typedef [container](#)< std::vector > [exprseq](#)
- typedef std::multiset< unsigned > [paramset](#)
- typedef [ex](#)(* [eval_func](#)) ()
- typedef [ex](#)(* [evalf_func](#)) ()
- typedef [ex](#)(* [conjugate_func](#)) ()
- typedef [ex](#)(* [real_part_func](#)) ()
- typedef [ex](#)(* [imag_part_func](#)) ()
- typedef [ex](#)(* [expand_func](#)) ()
- typedef [ex](#)(* [derivative_func](#)) ()
- typedef [ex](#)(* [expl_derivative_func](#)) ()
- typedef [ex](#)(* [power_func](#)) ()
- typedef [ex](#)(* [series_func](#)) ()
- typedef void(* [print_func](#)) ()
- typedef bool(* [info_func](#)) ()
- typedef [ex](#)(* [eval_func_1](#)) (const [ex](#) &)
- typedef [ex](#)(* [evalf_func_1](#)) (const [ex](#) &)
- typedef [ex](#)(* [conjugate_func_1](#)) (const [ex](#) &)
- typedef [ex](#)(* [real_part_func_1](#)) (const [ex](#) &)
- typedef [ex](#)(* [imag_part_func_1](#)) (const [ex](#) &)
- typedef [ex](#)(* [expand_func_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(* [derivative_func_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(* [expl_derivative_func_1](#)) (const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(* [power_func_1](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [series_func_1](#)) (const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(* [print_func_1](#)) (const [ex](#) &, const [print_context](#) &)
- typedef bool(* [info_func_1](#)) (const [ex](#) &, unsigned)
- typedef [ex](#)(* [eval_func_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [evalf_func_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [conjugate_func_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [real_part_func_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [imag_part_func_2](#)) (const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [expand_func_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(* [derivative_func_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(* [expl_derivative_func_2](#)) (const [ex](#) &, const [ex](#) &, const [symbol](#) &)
- typedef [ex](#)(* [power_func_2](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)
- typedef [ex](#)(* [series_func_2](#)) (const [ex](#) &, const [ex](#) &, const [relational](#) &, int, unsigned)
- typedef void(* [print_func_2](#)) (const [ex](#) &, const [ex](#) &, const [print_context](#) &)
- typedef bool(* [info_func_2](#)) (const [ex](#) &, const [ex](#) &, unsigned)
- typedef [ex](#)(* [eval_func_3](#)) (const [ex](#) &, const [ex](#) &, const [ex](#) &)

- [illegible]

- [illegible]

- ```

• typedef ex(* eval_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* evalf_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* conjugate_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* real_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* imag_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* expand_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
• typedef ex(* derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
• typedef ex(* expl_derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
• typedef ex(* power_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
• typedef ex(* series_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
• typedef void(* print_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
• typedef bool(* info_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
• typedef ex(* eval_funcp_exvector) (const exvector &)
• typedef ex(* evalf_funcp_exvector) (const exvector &)
• typedef ex(* conjugate_funcp_exvector) (const exvector &)
• typedef ex(* real_part_funcp_exvector) (const exvector &)
• typedef ex(* imag_part_funcp_exvector) (const exvector &)
• typedef ex(* expand_funcp_exvector) (const exvector &, unsigned)
• typedef ex(* derivative_funcp_exvector) (const exvector &, unsigned)
• typedef ex(* expl_derivative_funcp_exvector) (const exvector &, const symbol &)
• typedef ex(* power_funcp_exvector) (const exvector &, const ex &)
• typedef ex(* series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
• typedef void(* print_funcp_exvector) (const exvector &, const print_context &)
• typedef bool(* info_funcp_exvector) (const exvector &, unsigned)
• template<typename T , class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class Allocator = std::allocator<std::pair<const ex, T>>>>
 using exhashmap = std::unordered_map<ex, T, Hash, KeyEqual, Allocator>
• typedef std::map< spmapkey, ex > spmap
• typedef map< error_and_integral, ex, error_and_integral_is_less > lookup_map
• typedef container< std::list > lst
• typedef std::vector< std::size_t > uintvector
• typedef std::vector< unsigned > unsignedvector
• typedef std::vector< exvector > exvectorvector
• typedef std::vector< sym_desc > sym_desc_vec
• typedef void(* digits_changed_callback) (long)

 Function pointer to implement callbacks in the case 'Digits' gets changed.
• typedef class_info< print_context_options > print_context_class_info
• typedef class_info< registered_class_options > registered_class_info

```

## Enumerations

- enum { `callback_registered` = 1 }

## Functions

- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`add`, `expairseq`, `print_func`< `print_context` >(&`add::do_print`), `print_func`< `print_latex` >(&`add::do_print_latex`), `print_func`< `print_csrc` >(&`add::do_print_csrc`), `print_func`< `print_tree` >(&`add::do_print_tree`), `print_func`< `print_python_repr` >(&`add::do_print_python_repr`))  
`add`
- `GINAC_BIND_UNARCHIVER` (`add`)
- `GINAC_DECLARE_UNARCHIVER` (`add`)
- static void `write_unsigned` (`std::ostream` &`os`, unsigned `val`)  
*Write unsigned integer quantity to stream.*
- static unsigned `read_unsigned` (`std::istream` &`is`)  
*Read unsigned integer quantity from stream.*
- `std::ostream` & `operator<<` (`std::ostream` &`os`, const `archive_node` &`n`)  
*Write `archive_node` to binary data stream.*
- `std::ostream` & `operator<<` (`std::ostream` &`os`, const `archive` &`ar`)  
*Write archive to binary data stream.*
- `std::istream` & `operator>>` (`std::istream` &`is`, `archive_node` &`n`)  
*Read `archive_node` from binary data stream.*
- `std::istream` & `operator>>` (`std::istream` &`is`, `archive` &`ar`)  
*Read archive from binary data stream.*
- static `synthesize_func find_factory_fcn` (const `std::string` &`name`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`basic`, void, `print_func`< `print_context` >(&`basic::do_print`), `print_func`< `print_tree` >(&`basic::do_print_tree`), `print_func`< `print_python_repr` >(&`basic::do_print_python_repr`))  
`basic`  
*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by `duplicate()`), so it can copy the `info_key` and the `hash` value.*
- template<class `T` >  
bool `is_a` (const `basic` &`obj`)  
*Check if `obj` is a `T`, including base classes.*
- template<class `T` >  
bool `is_exactly_a` (const `basic` &`obj`)  
*Check if `obj` is a `T`, not including base classes.*
- template<class `B` , typename... `Args` >  
`B` & `dynallocate` (`Args` &&... `args`)  
*Constructs a new (class `basic` or derived) `B` object on the heap.*
- template<class `B` >  
`B` & `dynallocate` (`std::initializer_list`< `ex` > `il`)  
*Constructs a new (class `basic` or derived) `B` object on the heap.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`clifford`, `indexed`, `print_func`< `print_dflt` >(&`clifford::do_print_dflt`), `print_func`< `print_latex` >(&`clifford::do_print_latex`), `print_func`< `print_tree` >(&`clifford::do_print_tree`))  
`GINAC_IMPLEMENT_REGISTERED_CLASS_OPT`(`diracone`
- `print_func`< `print_dflt` > (&`diracone::do_print`), `print_func`< `print_latex` >(&`diracone`
- `GINAC_BIND_UNARCHIVER` (`clifford`)
- `GINAC_BIND_UNARCHIVER` (`cliffordunit`)
- `GINAC_BIND_UNARCHIVER` (`diracone`)
- `GINAC_BIND_UNARCHIVER` (`diracgamma`)
- `GINAC_BIND_UNARCHIVER` (`diracgamma5`)
- `GINAC_BIND_UNARCHIVER` (`diracgammaL`)
- `GINAC_BIND_UNARCHIVER` (`diracgammaR`)

- static bool `is_dirac_slash` (const `ex` &seq0)
- static void `base_and_index` (const `ex` &c, `ex` &b, `ex` &i)  
*This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$*
- `ex` `dirac_ONE` (unsigned char rl=0)  
*Create a Clifford unity object.*
- static unsigned `get_dim_uint` (const `ex` &e)
- `ex` `clifford_unit` (const `ex` &mu, const `ex` &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
- `ex` `dirac_gamma` (const `ex` &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
- `ex` `dirac_gamma5` (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
- `ex` `dirac_gammaL` (unsigned char rl=0)  
*Create a Dirac gammaL object.*
- `ex` `dirac_gammaR` (unsigned char rl=0)  
*Create a Dirac gammaR object.*
- `ex` `dirac_slash` (const `ex` &e, const `ex` &dim, unsigned char rl=0)  
*Create a term of the form  $e_{\mu} * \gamma \sim \mu$  with a unique index  $\mu$ .*
- static unsigned char `get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- static `ex` `trace_string` (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
- `ex` `dirac_trace` (const `ex` &e, const std::set< unsigned char > &rls, const `ex` &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
- `ex` `dirac_trace` (const `ex` &e, const `lst` &rls, const `ex` &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
- `ex` `dirac_trace` (const `ex` &e, unsigned char rl=0, const `ex` &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- `ex` `canonicalize_clifford` (const `ex` &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
- `ex` `clifford_star_bar` (const `ex` &e, bool do\_bar, unsigned `options`)  
*An auxillary function performing `clifford_star()` and `clifford_bar()`.*
- `ex` `clifford_prime` (const `ex` &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- `ex` `remove_dirac_ONE` (const `ex` &e, unsigned char rl=0, unsigned `options`=0)  
*Replaces `dirac_ONE`'s (with a representation\_label no less than rl) in e with 1.*
- int `clifford_max_label` (const `ex` &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if e contains at least one, otherwise returns -1.*
- `ex` `clifford_norm` (const `ex` &e)  
*Calculation of the norm in the Clifford algebra.*
- `ex` `clifford_inverse` (const `ex` &e)  
*Calculation of the inverse in the Clifford algebra.*
- `ex` `lst_to_clifford` (const `ex` &v, const `ex` &mu, const `ex` &metr, unsigned char rl=0)  
*List or vector conversion into the Clifford vector.*
- `ex` `lst_to_clifford` (const `ex` &v, const `ex` &e)  
*List or vector conversion into the Clifford vector.*
- static `ex` `get_clifford_comp` (const `ex` &e, const `ex` &c, bool root=true)  
*Auxiliary structure to define a function for stripping one Clifford unit from vectors.*
- `lst` `clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)  
*An inverse function to `lst_to_clifford()`.*

- `ex clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*
- `ex clifford_moebius_map` (const `ex` &M, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*
- `GINAC_DECLARE_UNARCHIVER` (clifford)
- `GINAC_DECLARE_UNARCHIVER` (diracone)
- `GINAC_DECLARE_UNARCHIVER` (cliffordunit)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma)
- `GINAC_DECLARE_UNARCHIVER` (diracgamma5)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaL)
- `GINAC_DECLARE_UNARCHIVER` (diracgammaR)
- `bool is_clifford_tinfo` (const `return_type_t` &ti)  
*Check whether a given `return_type_t` object (as returned by `return_type_tinfo()`) is that of a clifford object (with an arbitrary representation label).*
- `ex clifford_bar` (const `ex` &e)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- `ex clifford_star` (const `ex` &e)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (su3one, tensor, print\_func< `print_dflt` >(&su3one::do\_print). print\_func< `print_latex` >(&su3one::do\_print\_latex)) `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT`(su3t  
`print_func< print_dflt > (&su3t::do_print). print_func< print_latex >(&su3t`  
`GINAC_BIND_UNARCHIVER` (color)  
`GINAC_BIND_UNARCHIVER` (su3one)  
`GINAC_BIND_UNARCHIVER` (su3t)  
`GINAC_BIND_UNARCHIVER` (su3f)  
`GINAC_BIND_UNARCHIVER` (su3d)
- `static ex permute_free_index_to_front` (const `exvector` &iv3, const `exvector` &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- `ex color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- `static bool is_color_tinfo` (const `return_type_t` &ti)  
*Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).*
- `static unsigned char get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- `ex color_trace` (const `ex` &e, const `std::set< unsigned char >` &rls)  
*Calculate color traces over the specified set of representation labels.*
- `ex color_trace` (const `ex` &e, const `lst` &rls)  
*Calculate color traces over the specified list of representation labels.*
- `ex color_trace` (const `ex` &e, unsigned char rl=0)

*Calculate the trace of an expression containing color objects with a specified representation label.*

- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([color](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3one](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3t](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3f](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3d](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([constant](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[constant::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[constant::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[constant::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[constant::do\\_print\\_python\\_repr](#))) [const](#) ant
- [GINAC\\_BIND\\_UNARCHIVER](#) ([constant](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([constant](#))
- [static unsigned crc32](#) ([const char \\*c](#), [const unsigned len](#), [const unsigned crcinit](#))
- [bool are\\_ex\\_trivially\\_equal](#) ([const ex &e1](#), [const ex &e2](#))

*Compare two objects of class quickly without doing a deep tree traversal.*

- [std::ostream & operator<<](#) ([std::ostream &os](#), [const exvector &e](#))
- [std::ostream & operator<<](#) ([std::ostream &os](#), [const exset &e](#))
- [std::ostream & operator<<](#) ([std::ostream &os](#), [const exmap &e](#))
- [size\\_t nops](#) ([const ex &thisex](#))
- [ex expand](#) ([const ex &thisex](#), [unsigned options=0](#))
- [ex conjugate](#) ([const ex &thisex](#))
- [ex real\\_part](#) ([const ex &thisex](#))
- [ex imag\\_part](#) ([const ex &thisex](#))
- [bool has](#) ([const ex &thisex](#), [const ex &pattern](#), [unsigned options=0](#))
- [bool find](#) ([const ex &thisex](#), [const ex &pattern](#), [exset &found](#))
- [bool is\\_polynomial](#) ([const ex &thisex](#), [const ex &vars](#))
- [int degree](#) ([const ex &thisex](#), [const ex &s](#))
- [int ldegree](#) ([const ex &thisex](#), [const ex &s](#))
- [ex coeff](#) ([const ex &thisex](#), [const ex &s](#), [int n=1](#))
- [ex numer](#) ([const ex &thisex](#))
- [ex denom](#) ([const ex &thisex](#))
- [ex numer\\_denom](#) ([const ex &thisex](#))
- [ex normal](#) ([const ex &thisex](#))
- [ex to\\_rational](#) ([const ex &thisex](#), [exmap &repl](#))
- [ex to\\_polynomial](#) ([const ex &thisex](#), [exmap &repl](#))
- [ex collect](#) ([const ex &thisex](#), [const ex &s](#), [bool distributed=false](#))
- [ex eval](#) ([const ex &thisex](#))
- [ex evalf](#) ([const ex &thisex](#))
- [ex evalm](#) ([const ex &thisex](#))
- [ex eval\\_integ](#) ([const ex &thisex](#))
- [ex diff](#) ([const ex &thisex](#), [const symbol &s](#), [unsigned nth=1](#))
- [ex series](#) ([const ex &thisex](#), [const ex &r](#), [int order](#), [unsigned options=0](#))
- [bool match](#) ([const ex &thisex](#), [const ex &pattern](#), [exmap &repl\\_lst](#))
- [ex simplify\\_indexed](#) ([const ex &thisex](#), [unsigned options=0](#))
- [ex simplify\\_indexed](#) ([const ex &thisex](#), [const scalar\\_products &sp](#), [unsigned options=0](#))
- [ex symmetrize](#) ([const ex &thisex](#))
- [ex symmetrize](#) ([const ex &thisex](#), [const lst &l](#))
- [ex antisymmetrize](#) ([const ex &thisex](#))
- [ex antisymmetrize](#) ([const ex &thisex](#), [const lst &l](#))
- [ex symmetrize\\_cyclic](#) ([const ex &thisex](#))
- [ex symmetrize\\_cyclic](#) ([const ex &thisex](#), [const lst &l](#))
- [ex op](#) ([const ex &thisex](#), [size\\_t i](#))
- [ex lhs](#) ([const ex &thisex](#))
- [ex rhs](#) ([const ex &thisex](#))
- [bool is\\_zero](#) ([const ex &thisex](#))

- `void swap (ex &e1, ex &e2)`
- `ex subs (const ex &thisex, const exmap &m, unsigned options=0)`
- `ex subs (const ex &thisex, const lst &ls, const lst &lr, unsigned options=0)`
- `ex subs (const ex &thisex, const ex &e, unsigned options=0)`
- `template<class T >`  
`bool is_a (const ex &obj)`  
*Check if ex is a handle to a T, including base classes.*
- `template<class T >`  
`bool is_exactly_a (const ex &obj)`  
*Check if ex is a handle to a T, not including base classes.*
- `template<class T >`  
`const T & ex_to (const ex &e)`  
*Return a reference to the basic-derived class T object embedded in an expression.*
- `void compile_ex (const ex &expr, const symbol &sym, FUNCPC_1P &fp, const std::string filename="")`  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex (const ex &expr, const symbol &sym1, const symbol &sym2, FUNCPC_2P &fp, const std::string filename="")`  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void compile_ex (const lst &exprs, const lst &syms, FUNCPC_CUBA &fp, const std::string filename="")`  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- `void link_ex (const std::string filename, FUNCPC_1P &fp)`  
*Opens an existing so-file and returns a function pointer of type FUNCPC\_1P to the contained function.*
- `void link_ex (const std::string filename, FUNCPC_2P &fp)`  
*Opens an existing so-file and returns a function pointer of type FUNCPC\_2P to the contained function.*
- `void link_ex (const std::string filename, FUNCPC_CUBA &fp)`  
*Opens an existing so-file and returns a function pointer of type FUNCPC\_CUBA to the contained function.*
- `void unlink_ex (const std::string filename)`  
*Closes all linked .so files that have the supplied filename.*
- `void swap (expair &e1, expair &e2)`
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (expairseq, basic, print_func< print_context >(&expairseq::do_print), print_func< print_tree >(&expairseq::do_print_tree)) class epp_is_less`
- `epvector * conjugateepvector (const epvector &)`  
*Complex conjugate every element of an epvector.*
- `ex factor (const ex &poly, unsigned options)`  
*Interface function to the outside world.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (fail, basic, print_func< print_context >(&fail::do_print), print_func< print_tree >(&fail::do_print_tree)) GINAC_BIND_UNARCHIVER(fail)`
- `GINAC_DECLARE_UNARCHIVER (fail)`
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (fderivative, function, print_func< print_context >(&fderivative::do_print), print_func< print_latex >(&fderivative::do_print_latex), print_func< print_csrc >(&fderivative::do_print_csrc), print_func< print_tree >(&fderivative::do_print_tree)) fderivative`
- `GINAC_BIND_UNARCHIVER (fderivative)`
- `GINAC_DECLARE_UNARCHIVER (fderivative)`
- `GINAC_BIND_UNARCHIVER (function)`
- `GINAC_DECLARE_UNARCHIVER (function)`
- `template<typename T >`  
`bool is_the_function (const ex &x)`
- `static unsigned make_hash_seed (const std::type_info &tinfo)`  
*We need a hash function which gives different values for objects of different types.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (idx, basic, print_func< print_context >(&idx::do_print), print_func< print_latex >(&idx::do_print_latex), print_func< print_csrc >(&idx::do_print_csrc), print_func< print_tree >(&idx::do_print_tree)) GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(varidx`
- `print_func< print_context > (&varidx::do_print), print_func< print_latex >(&varidx`

- [GINAC\\_BIND\\_UNARCHIVER](#) (idx)
- [GINAC\\_BIND\\_UNARCHIVER](#) (varidx)
- [GINAC\\_BIND\\_UNARCHIVER](#) (spinidx)
- bool [is\\_dummy\\_pair](#) (const [idx](#) &i1, const [idx](#) &i2)  
*Check whether two indices form a dummy pair.*
- bool [is\\_dummy\\_pair](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Check whether two expressions form a dummy index pair.*
- void [find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- [ex](#) [minimal\\_dim](#) (const [ex](#) &dim1, const [ex](#) &dim2)  
*Return the minimum of two index dimensions.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (idx)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (varidx)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (spinidx)
- void [find\\_free\\_and\\_dummy](#) (const [exvector](#) &v, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [find\\_dummy\\_indices](#) (const [exvector](#) &v, [exvector](#) &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [count\\_dummy\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [count\\_free\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (indexed, [exprseq](#), print\_func< [print\\_context](#) >(&indexed::do\_print), print\_func< [print\\_latex](#) >(&indexed::do\_print\_latex), print\_func< [print\\_tree](#) >(&indexed::do\_print\_tree))  
[indexed](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) (indexed)
- static bool [indices\\_consistent](#) (const [exvector](#) &v1, const [exvector](#) &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- template<class T >  
size\_t [number\\_of\\_type](#) (const [exvector](#) &v)
- template<class T >  
static [ex](#) [rename\\_dummy\\_indices](#) (const [ex](#) &e, [exvector](#) &global\_dummy\_indices, [exvector](#) &local\_dummy\_↵  
\_indices)  
*Rename dummy indices in an expression.*
- static void [find\\_variant\\_indices](#) (const [exvector](#) &v, [exvector](#) &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- bool [reposition\\_dummy\\_indices](#) ([ex](#) &e, [exvector](#) &variant\_dummy\_indices, [exvector](#) &moved\_indices)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- static void [product\\_to\\_exvector](#) (const [ex](#) &e, [exvector](#) &v, bool &non\_commutative)
- template<class T >  
[ex](#) [idx\\_symmetrization](#) (const [ex](#) &r, const [exvector](#) &local\_dummy\_indices)
- [ex](#) [simplify\\_indexed](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
*Simplify indexed expression, return list of free indices.*
- [ex](#) [simplify\\_indexed\\_product](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.*
- bool [hasindex](#) (const [ex](#) &x, const [ex](#) &sym)
- [exvector](#) [get\\_all\\_dummy\\_indices\\_safely](#) (const [ex](#) &e)

*More reliable version of the form.*

- [exvector get\\_all\\_dummy\\_indices](#) (const [ex](#) &e)  
*Returns all dummy indices from the exvector.*
- [lst rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &va, const [exvector](#) &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- [ex rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &va, const [exvector](#) &vb, const [ex](#) &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- [ex rename\\_dummy\\_indices\\_uniquely](#) (const [ex](#) &a, const [ex](#) &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- [ex rename\\_dummy\\_indices\\_uniquely](#) ([exvector](#) &va, const [ex](#) &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.*
- [ex expand\\_dummy\\_sum](#) (const [ex](#) &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (indexed)
- static [ex conjugate\\_evalf](#) (const [ex](#) &arg)
- static [ex conjugate\\_eval](#) (const [ex](#) &arg)
- static void [conjugate\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex conjugate\\_conjugate](#) (const [ex](#) &arg)
- static [ex conjugate\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- static [ex conjugate\\_real\\_part](#) (const [ex](#) &arg)
- static [ex conjugate\\_imag\\_part](#) (const [ex](#) &arg)
- static bool [func\\_arg\\_info](#) (const [ex](#) &arg, unsigned inf)
- static bool [conjugate\\_info](#) (const [ex](#) &arg, unsigned inf)
- [REGISTER\\_FUNCTION](#) (conjugate\_function, eval\_func([conjugate\\_eval](#)). evalf\_func([conjugate\\_evalf](#)).  
expl\_derivative\_func([conjugate\\_expl\\_derivative](#)). info\_func([conjugate\\_info](#)). print\_func< [print\\_latex](#)  
>([conjugate\\_print\\_latex](#)). conjugate\_func([conjugate\\_conjugate](#)). real\_part\_func([conjugate\\_real\\_part](#)).  
imag\_part\_func([conjugate\\_imag\\_part](#)). set\_name("conjugate","conjugate"))
- static [ex real\\_part\\_evalf](#) (const [ex](#) &arg)
- static [ex real\\_part\\_eval](#) (const [ex](#) &arg)
- static void [real\\_part\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex real\\_part\\_conjugate](#) (const [ex](#) &arg)
- static [ex real\\_part\\_real\\_part](#) (const [ex](#) &arg)
- static [ex real\\_part\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex real\\_part\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- [REGISTER\\_FUNCTION](#) (real\_part\_function, eval\_func([real\\_part\\_eval](#)). evalf\_func([real\\_part\\_evalf](#)). expl←  
\_expl\_derivative\_func([real\\_part\\_expl\\_derivative](#)). print\_func< [print\\_latex](#) >([real\\_part\\_print\\_latex](#)). conjugate←  
\_func([real\\_part\\_conjugate](#)). real\_part\_func([real\\_part\\_real\\_part](#)). imag\_part\_func([real\\_part\\_imag\\_part](#)).  
set\_name("real\_part","real\_part"))
- static [ex imag\\_part\\_evalf](#) (const [ex](#) &arg)
- static [ex imag\\_part\\_eval](#) (const [ex](#) &arg)
- static void [imag\\_part\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex imag\\_part\\_conjugate](#) (const [ex](#) &arg)
- static [ex imag\\_part\\_real\\_part](#) (const [ex](#) &arg)
- static [ex imag\\_part\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex imag\\_part\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- [REGISTER\\_FUNCTION](#) (imag\_part\_function, eval\_func([imag\\_part\\_eval](#)). evalf\_func([imag\\_part\\_evalf](#)).  
expl\_derivative\_func([imag\\_part\\_expl\\_derivative](#)). print\_func< [print\\_latex](#) >([imag\\_part\\_print\\_latex](#)).  
conjugate\_func([imag\\_part\\_conjugate](#)). real\_part\_func([imag\\_part\\_real\\_part](#)). imag\_part\_func([imag\\_part\\_imag\\_part](#)).  
set\_name("imag\_part","imag\_part"))
- static [ex abs\\_evalf](#) (const [ex](#) &arg)
- static [ex abs\\_eval](#) (const [ex](#) &arg)
- static [ex abs\\_expand](#) (const [ex](#) &arg, unsigned [options](#))

- static `ex abs_expl_derivative` (const `ex` &arg, const `symbol` &s)
- static void `abs_print_latex` (const `ex` &arg, const `print_context` &c)
- static void `abs_print_csrc_float` (const `ex` &arg, const `print_context` &c)
- static `ex abs_conjugate` (const `ex` &arg)
- static `ex abs_real_part` (const `ex` &arg)
- static `ex abs_imag_part` (const `ex` &arg)
- static `ex abs_power` (const `ex` &arg, const `ex` &exp)
- bool `abs_info` (const `ex` &arg, unsigned inf)
- `REGISTER_FUNCTION` (`abs`, `eval_func(abs_eval)`. `evalf_func(abs_evalf)`. `expand_func(abs_expand)`. `expl_derivative_func(abs_expl_derivative)`. `info_func(abs_info)`. `print_func< print_latex >(abs_print_latex)`. `print_func< print_csrc_float >(abs_print_csrc_float)`. `print_func< print_csrc_double >(abs_print_csrc_double)`. `conjugate_func(abs_conjugate)`. `real_part_func(abs_real_part)`. `imag_part_func(abs_imag_part)`. `power_func(abs_power)`)
- static `ex step_evalf` (const `ex` &arg)
- static `ex step_eval` (const `ex` &arg)
- static `ex step_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex step_conjugate` (const `ex` &arg)
- static `ex step_real_part` (const `ex` &arg)
- static `ex step_imag_part` (const `ex` &arg)
- `REGISTER_FUNCTION` (`step`, `eval_func(step_eval)`. `evalf_func(step_evalf)`. `series_func(step_series)`. `conjugate_func(step_conjugate)`. `real_part_func(step_real_part)`. `imag_part_func(step_imag_part)`)
- static `ex csgn_evalf` (const `ex` &arg)
- static `ex csgn_eval` (const `ex` &arg)
- static `ex csgn_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex csgn_conjugate` (const `ex` &arg)
- static `ex csgn_real_part` (const `ex` &arg)
- static `ex csgn_imag_part` (const `ex` &arg)
- static `ex csgn_power` (const `ex` &arg, const `ex` &exp)
- `REGISTER_FUNCTION` (`csgn`, `eval_func(csgn_eval)`. `evalf_func(csgn_evalf)`. `series_func(csgn_series)`. `conjugate_func(csgn_conjugate)`. `real_part_func(csgn_real_part)`. `imag_part_func(csgn_imag_part)`. `power_func(csgn_power)`)
- static `ex eta_evalf` (const `ex` &x, const `ex` &y)
- static `ex eta_eval` (const `ex` &x, const `ex` &y)
- static `ex eta_series` (const `ex` &x, const `ex` &y, const `relational` &rel, int `order`, unsigned `options`)
- static `ex eta_conjugate` (const `ex` &x, const `ex` &y)
- static `ex eta_real_part` (const `ex` &x, const `ex` &y)
- static `ex eta_imag_part` (const `ex` &x, const `ex` &y)
- `REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`. `evalf_func(eta_evalf)`. `series_func(eta_series)`. `latex_name("\\eta")`. `set_symmetry(sy_symm(0, 1))`. `conjugate_func(eta_conjugate)`. `real_part_func(eta_real_part)`. `imag_part_func(eta_imag_part)`)
- static `ex Li2_evalf` (const `ex` &x)
- static `ex Li2_eval` (const `ex` &x)
- static `ex Li2_deriv` (const `ex` &x, unsigned `deriv_param`)
- static `ex Li2_series` (const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex Li2_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`. `evalf_func(Li2_evalf)`. `derivative_func(Li2_deriv)`. `series_func(Li2_series)`. `conjugate_func(Li2_conjugate)`. `latex_name("\\mathrm{Li}_2")`)
- static `ex Li3_eval` (const `ex` &x)
- `REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`. `latex_name("\\mathrm{Li}_3")`)
- static `ex zetaderiv_eval` (const `ex` &n, const `ex` &x)
- static `ex zetaderiv_deriv` (const `ex` &n, const `ex` &x, unsigned `deriv_param`)
- `REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`. `derivative_func(zetaderiv_deriv)`. `latex_name("\\zeta^{\\prime}")`)
- static `ex factorial_evalf` (const `ex` &x)
- static `ex factorial_eval` (const `ex` &x)

- static void `factorial_print_dflt_latex` (const `ex` &`x`, const `print_context` &`c`)
- static `ex` `factorial_conjugate` (const `ex` &`x`)
- static `ex` `factorial_real_part` (const `ex` &`x`)
- static `ex` `factorial_imag_part` (const `ex` &`x`)
- `REGISTER_FUNCTION` (`factorial`, `eval_func`(`factorial_eval`). `evalf_func`(`factorial_evalf`). `print_func`<`print_dflt` >(`factorial_print_dflt_latex`). `print_func`< `print_latex` >(`factorial_print_dflt_latex`). `conjugate_↔`  
`func`(`factorial_conjugate`). `real_part_func`(`factorial_real_part`). `imag_part_func`(`factorial_imag_part`))
- static `ex` `binomial_evalf` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `binomial_sym` (const `ex` &`x`, const `numeric` &`y`)
- static `ex` `binomial_eval` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `binomial_conjugate` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `binomial_real_part` (const `ex` &`x`, const `ex` &`y`)
- static `ex` `binomial_imag_part` (const `ex` &`x`, const `ex` &`y`)
- `REGISTER_FUNCTION` (`binomial`, `eval_func`(`binomial_eval`). `evalf_func`(`binomial_evalf`). `conjugate_↔`  
`func`(`binomial_conjugate`). `real_part_func`(`binomial_real_part`). `imag_part_func`(`binomial_imag_part`))
- static `ex` `Order_eval` (const `ex` &`x`)
- static `ex` `Order_series` (const `ex` &`x`, const `relational` &`r`, int `order`, unsigned `options`)
- static `ex` `Order_conjugate` (const `ex` &`x`)
- static `ex` `Order_real_part` (const `ex` &`x`)
- static `ex` `Order_imag_part` (const `ex` &`x`)
- static `ex` `Order_power` (const `ex` &`x`, const `ex` &`e`)
- static `ex` `Order_expl_derivative` (const `ex` &`arg`, const `symbol` &`s`)
- `REGISTER_FUNCTION` (`Order`, `eval_func`(`Order_eval`). `series_func`(`Order_series`). `latex_name`("\\mathcal{O}").  
`expl_derivative_func`(`Order_expl_derivative`). `conjugate_func`(`Order_conjugate`). `real_part_func`(`Order_real_part`).  
`imag_part_func`(`Order_imag_part`). `power_func`(`Order_power`))
- `ex` `Isolve` (const `ex` &`eqns`, const `ex` &`symbols`, unsigned `options=solve_algo::automatic`)  
*Factorial function.*
- const `numeric` `fsolve` (const `ex` &`f`, const `symbol` &`x`, const `numeric` &`x1`, const `numeric` &`x2`)  
*Find a real root of real-valued function f(x) numerically within a given interval.*
- template<typename T1 >  
`function` `zeta` (const T1 &`p1`)
- template<typename T1 , typename T2 >  
`function` `zeta` (const T1 &`p1`, const T2 &`p2`)
- template<> bool `is_the_function`< `zeta_SERIAL` > (const `ex` &`x`)
- template<typename T1 , typename T2 >  
`function` `G` (const T1 &`x`, const T2 &`y`)
- template<typename T1 , typename T2 , typename T3 >  
`function` `G` (const T1 &`x`, const T2 &`s`, const T3 &`y`)
- template<> bool `is_the_function`< `G_SERIAL` > (const `ex` &`x`)
- template<typename T1 >  
`function` `psi` (const T1 &`p1`)
- template<typename T1 , typename T2 >  
`function` `psi` (const T1 &`p1`, const T2 &`p2`)
- template<> bool `is_the_function`< `psi_SERIAL` > (const `ex` &`x`)
- template<typename T1 , typename T2 >  
`function` `iterated_integral` (const T1 &`kernel_lst`, const T2 &`lambda`)
- template<typename T1 , typename T2 , typename T3 >  
`function` `iterated_integral` (const T1 &`kernel_lst`, const T2 &`lambda`, const T3 &`N_trunc`)
- template<> bool `is_the_function`< `iterated_integral_SERIAL` > (const `ex` &`x`)
- bool `is_order_function` (const `ex` &`e`)  
*Check whether a function is the Order (O(n)) function.*
- `ex` `convert_H_to_Li` (const `ex` &`parameterlst`, const `ex` &`arg`)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*
- static `ex` `EllipticK_evalf` (const `ex` &`k`)

- static `ex EllipticK_eval` (const `ex` &k)
- static `ex EllipticK_deriv` (const `ex` &k, unsigned deriv\_param)
- static `ex EllipticK_series` (const `ex` &k, const `relational` &rel, int `order`, unsigned `options`)
- static void `EllipticK_print_latex` (const `ex` &k, const `print_context` &c)
- `REGISTER_FUNCTION` (EllipticK, evalf\_func(EllipticK\_evalf). eval\_func(EllipticK\_eval). derivative\_↵  
func(EllipticK\_deriv). series\_func(EllipticK\_series). print\_func< print\_latex >(EllipticK\_print\_latex). do\_↵  
not\_evalf\_params())
- static `ex EllipticE_evalf` (const `ex` &k)
- static `ex EllipticE_eval` (const `ex` &k)
- static `ex EllipticE_deriv` (const `ex` &k, unsigned deriv\_param)
- static `ex EllipticE_series` (const `ex` &k, const `relational` &rel, int `order`, unsigned `options`)
- static void `EllipticE_print_latex` (const `ex` &k, const `print_context` &c)
- `REGISTER_FUNCTION` (EllipticE, evalf\_func(EllipticE\_evalf). eval\_func(EllipticE\_eval). derivative\_↵  
func(EllipticE\_deriv). series\_func(EllipticE\_series). print\_func< print\_latex >(EllipticE\_print\_latex). do\_↵  
not\_evalf\_params())
- static `ex iterated_integral_evalf_impl` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)
- static `ex iterated_integral2_evalf` (const `ex` &kernel\_lst, const `ex` &lambda)
- static `ex iterated_integral3_evalf` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)
- static `ex iterated_integral2_eval` (const `ex` &kernel\_lst, const `ex` &lambda)
- static `ex iterated_integral3_eval` (const `ex` &kernel\_lst, const `ex` &lambda, const `ex` &N\_trunc)
- static `ex lgamma_evalf` (const `ex` &x)
- static `ex lgamma_eval` (const `ex` &x)

*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*

- static `ex lgamma_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex lgamma_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex lgamma_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (lgamma, eval\_func(lgamma\_eval). evalf\_func(lgamma\_evalf). derivative\_↵  
func(lgamma\_deriv). series\_func(lgamma\_series). conjugate\_func(lgamma\_conjugate). latex\_name("\\log  
\\Gamma"))
- static `ex tgamma_evalf` (const `ex` &x)
- static `ex tgamma_eval` (const `ex` &x)

*Evaluation of  $tgamma(x)$ , the true Gamma function.*

- static `ex tgamma_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex tgamma_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex tgamma_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (tgamma, eval\_func(tgamma\_eval). evalf\_func(tgamma\_evalf). derivative\_↵  
\_func(tgamma\_deriv). series\_func(tgamma\_series). conjugate\_func(tgamma\_conjugate). latex\_↵  
name("\\Gamma"))
- static `ex beta_evalf` (const `ex` &x, const `ex` &y)
- static `ex beta_eval` (const `ex` &x, const `ex` &y)
- static `ex beta_deriv` (const `ex` &x, const `ex` &y, unsigned deriv\_param)
- static `ex beta_series` (const `ex` &arg1, const `ex` &arg2, const `relational` &rel, int `order`, unsigned `options`)
- `REGISTER_FUNCTION` (beta, eval\_func(beta\_eval). evalf\_func(beta\_evalf). derivative\_func(beta\_deriv).  
series\_func(beta\_series). latex\_name("\\mathrm{B}"). set\_symmetry(sy\_symm(0, 1)))
- static `ex psi1_evalf` (const `ex` &x)
- static `ex psi1_eval` (const `ex` &x)

*Evaluation of digamma-function  $\psi(x)$ .*

- static `ex psi1_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex psi1_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex psi2_evalf` (const `ex` &n, const `ex` &x)
- static `ex psi2_eval` (const `ex` &n, const `ex` &x)

*Evaluation of polygamma-function  $\psi(n,x)$ .*

- static `ex psi2_deriv` (const `ex` &n, const `ex` &x, unsigned deriv\_param)
- static `ex psi2_series` (const `ex` &n, const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)

- static `ex G2_evalf` (const `ex` &`x_`, const `ex` &`y`)
- static `ex G2_eval` (const `ex` &`x_`, const `ex` &`y`)
- static `ex G3_evalf` (const `ex` &`x_`, const `ex` &`s_`, const `ex` &`y`)
- static `ex G3_eval` (const `ex` &`x_`, const `ex` &`s_`, const `ex` &`y`)
- static `ex Li_evalf` (const `ex` &`m_`, const `ex` &`x_`)
- static `ex Li_eval` (const `ex` &`m_`, const `ex` &`x_`)
- static `ex Li_series` (const `ex` &`m`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex Li_deriv` (const `ex` &`m_`, const `ex` &`x_`, unsigned `deriv_param`)
- static void `Li_print_latex` (const `ex` &`m_`, const `ex` &`x_`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`Li`, `evalf_func`(`Li_evalf`). `eval_func`(`Li_eval`). `series_func`(`Li_series`). `derivative_`↔  
`func`(`Li_deriv`). `print_func`< `print_latex` >(`Li_print_latex`). `do_not_evalf_params`()
- static `ex S_evalf` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex S_eval` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex S_series` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex S_deriv` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, unsigned `deriv_param`)
- static void `S_print_latex` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`S`, `evalf_func`(`S_evalf`). `eval_func`(`S_eval`). `series_func`(`S_series`). `derivative_`↔  
`func`(`S_deriv`). `print_func`< `print_latex` >(`S_print_latex`). `do_not_evalf_params`()
- static `ex H_evalf` (const `ex` &`x1`, const `ex` &`x2`)
- static `ex H_eval` (const `ex` &`m_`, const `ex` &`x`)
- static `ex H_series` (const `ex` &`m`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex H_deriv` (const `ex` &`m_`, const `ex` &`x`, unsigned `deriv_param`)
- static void `H_print_latex` (const `ex` &`m_`, const `ex` &`x`, const `print_context` &`c`)
- `REGISTER_FUNCTION` (`H`, `evalf_func`(`H_evalf`). `eval_func`(`H_eval`). `series_func`(`H_series`). `derivative_`↔  
`func`(`H_deriv`). `print_func`< `print_latex` >(`H_print_latex`). `do_not_evalf_params`()
- static `ex zeta1_evalf` (const `ex` &`x`)
- static `ex zeta1_eval` (const `ex` &`m`)
- static `ex zeta1_deriv` (const `ex` &`m`, unsigned `deriv_param`)
- static void `zeta1_print_latex` (const `ex` &`m_`, const `print_context` &`c`)
- static `ex zeta2_evalf` (const `ex` &`x`, const `ex` &`s`)
- static `ex zeta2_eval` (const `ex` &`m`, const `ex` &`s_`)
- static `ex zeta2_deriv` (const `ex` &`m`, const `ex` &`s`, unsigned `deriv_param`)
- static void `zeta2_print_latex` (const `ex` &`m_`, const `ex` &`s_`, const `print_context` &`c`)
- static `ex exp_evalf` (const `ex` &`x`)
- static `ex exp_eval` (const `ex` &`x`)
- static `ex exp_expand` (const `ex` &`arg`, unsigned `options`)
- static `ex exp_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex exp_real_part` (const `ex` &`x`)
- static `ex exp_imag_part` (const `ex` &`x`)
- static `ex exp_conjugate` (const `ex` &`x`)
- static `ex exp_power` (const `ex` &`x`, const `ex` &`a`)
- static bool `exp_info` (const `ex` &`x`, unsigned `inf`)
- `REGISTER_FUNCTION` (`exp`, `eval_func`(`exp_eval`). `evalf_func`(`exp_evalf`). `info_func`(`exp_info`). `expand_`↔  
`func`(`exp_expand`). `derivative_func`(`exp_deriv`). `real_part_func`(`exp_real_part`). `imag_part_func`(`exp_imag_part`).  
`conjugate_func`(`exp_conjugate`). `power_func`(`exp_power`). `latex_name`("\\exp")
- static `ex log_evalf` (const `ex` &`x`)
- static `ex log_eval` (const `ex` &`x`)
- static `ex log_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex log_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex log_real_part` (const `ex` &`x`)
- static `ex log_imag_part` (const `ex` &`x`)
- static `ex log_expand` (const `ex` &`arg`, unsigned `options`)
- static `ex log_conjugate` (const `ex` &`x`)
- static bool `log_info` (const `ex` &`x`, unsigned `inf`)

- [REGISTER\\_FUNCTION](#) ([log](#), [eval\\_func\(log\\_eval\)](#). [evalf\\_func\(log\\_evalf\)](#). [info\\_func\(log\\_info\)](#). [expand\\_func\(log\\_expand\)](#). [derivative\\_func\(log\\_deriv\)](#). [series\\_func\(log\\_series\)](#). [real\\_part\\_func\(log\\_real\\_part\)](#). [imag\\_part\\_func\(log\\_imag\\_part\)](#). [conjugate\\_func\(log\\_conjugate\)](#). [latex\\_name\("\\ln"\)](#))
- static [ex sin\\_evalf](#) (const [ex](#) &[x](#))
- static [ex sin\\_eval](#) (const [ex](#) &[x](#))
- static [ex sin\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex sin\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex sin\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex sin\\_conjugate](#) (const [ex](#) &[x](#))
- static bool [trig\\_info](#) (const [ex](#) &[x](#), unsigned [inf](#))
- [REGISTER\\_FUNCTION](#) ([sin](#), [eval\\_func\(sin\\_eval\)](#). [evalf\\_func\(sin\\_evalf\)](#). [info\\_func\(trig\\_info\)](#). [derivative\\_func\(sin\\_deriv\)](#). [real\\_part\\_func\(sin\\_real\\_part\)](#). [imag\\_part\\_func\(sin\\_imag\\_part\)](#). [conjugate\\_func\(sin\\_conjugate\)](#). [latex\\_name\("\\sin"\)](#))
- static [ex cos\\_evalf](#) (const [ex](#) &[x](#))
- static [ex cos\\_eval](#) (const [ex](#) &[x](#))
- static [ex cos\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex cos\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex cos\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex cos\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([cos](#), [eval\\_func\(cos\\_eval\)](#). [info\\_func\(trig\\_info\)](#). [evalf\\_func\(cos\\_evalf\)](#). [derivative\\_func\(cos\\_deriv\)](#). [real\\_part\\_func\(cos\\_real\\_part\)](#). [imag\\_part\\_func\(cos\\_imag\\_part\)](#). [conjugate\\_func\(cos\\_conjugate\)](#). [latex\\_name\("\\cos"\)](#))
- static [ex tan\\_evalf](#) (const [ex](#) &[x](#))
- static [ex tan\\_eval](#) (const [ex](#) &[x](#))
- static [ex tan\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex tan\\_real\\_part](#) (const [ex](#) &[x](#))
- static [ex tan\\_imag\\_part](#) (const [ex](#) &[x](#))
- static [ex tan\\_series](#) (const [ex](#) &[x](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- static [ex tan\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([tan](#), [eval\\_func\(tan\\_eval\)](#). [evalf\\_func\(tan\\_evalf\)](#). [info\\_func\(trig\\_info\)](#). [derivative\\_func\(tan\\_deriv\)](#). [series\\_func\(tan\\_series\)](#). [real\\_part\\_func\(tan\\_real\\_part\)](#). [imag\\_part\\_func\(tan\\_imag\\_part\)](#). [conjugate\\_func\(tan\\_conjugate\)](#). [latex\\_name\("\\tan"\)](#))
- static [ex asin\\_evalf](#) (const [ex](#) &[x](#))
- static [ex asin\\_eval](#) (const [ex](#) &[x](#))
- static [ex asin\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex asin\\_conjugate](#) (const [ex](#) &[x](#))
- static bool [asin\\_info](#) (const [ex](#) &[x](#), unsigned [inf](#))
- [REGISTER\\_FUNCTION](#) ([asin](#), [eval\\_func\(asin\\_eval\)](#). [evalf\\_func\(asin\\_evalf\)](#). [info\\_func\(asin\\_info\)](#). [derivative\\_func\(asin\\_deriv\)](#). [conjugate\\_func\(asin\\_conjugate\)](#). [latex\\_name\("\\arcsin"\)](#))
- static [ex acos\\_evalf](#) (const [ex](#) &[x](#))
- static [ex acos\\_eval](#) (const [ex](#) &[x](#))
- static [ex acos\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex acos\\_conjugate](#) (const [ex](#) &[x](#))
- [REGISTER\\_FUNCTION](#) ([acos](#), [eval\\_func\(acos\\_eval\)](#). [evalf\\_func\(acos\\_evalf\)](#). [info\\_func\(asin\\_info\)](#). [derivative\\_func\(acos\\_deriv\)](#). [conjugate\\_func\(acos\\_conjugate\)](#). [latex\\_name\("\\arccos"\)](#))
- static [ex atan\\_evalf](#) (const [ex](#) &[x](#))
- static [ex atan\\_eval](#) (const [ex](#) &[x](#))
- static [ex atan\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex atan\\_series](#) (const [ex](#) &[arg](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- static [ex atan\\_conjugate](#) (const [ex](#) &[x](#))
- static bool [atan\\_info](#) (const [ex](#) &[x](#), unsigned [inf](#))
- [REGISTER\\_FUNCTION](#) ([atan](#), [eval\\_func\(atan\\_eval\)](#). [evalf\\_func\(atan\\_evalf\)](#). [info\\_func\(atan\\_info\)](#). [derivative\\_func\(atan\\_deriv\)](#). [series\\_func\(atan\\_series\)](#). [conjugate\\_func\(atan\\_conjugate\)](#). [latex\\_name\("\\arctan"\)](#))
- static [ex atan2\\_evalf](#) (const [ex](#) &[y](#), const [ex](#) &[x](#))

- static `ex atan2_eval` (const `ex` &y, const `ex` &x)
- static `ex atan2_deriv` (const `ex` &y, const `ex` &x, unsigned deriv\_param)
- static bool `atan2_info` (const `ex` &y, const `ex` &x, unsigned inf)
- `REGISTER_FUNCTION` (`atan2`, `eval_func(atan2_eval)`. `evalf_func(atan2_evalf)`. `info_func(atan2_info)`. `evalf_func(atan2_evalf)`. `derivative_func(atan2_deriv)`)
- static `ex sinh_evalf` (const `ex` &x)
- static `ex sinh_eval` (const `ex` &x)
- static `ex sinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex sinh_real_part` (const `ex` &x)
- static `ex sinh_imag_part` (const `ex` &x)
- static `ex sinh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`sinh`, `eval_func(sinh_eval)`. `evalf_func(sinh_evalf)`. `info_func(atan_info)`. `derivative_func(sinh_deriv)`. `real_part_func(sinh_real_part)`. `imag_part_func(sinh_imag_part)`. `conjugate_←func(sinh_conjugate)`. `latex_name("\\sinh")`)
- static `ex cosh_evalf` (const `ex` &x)
- static `ex cosh_eval` (const `ex` &x)
- static `ex cosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex cosh_real_part` (const `ex` &x)
- static `ex cosh_imag_part` (const `ex` &x)
- static `ex cosh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`cosh`, `eval_func(cosh_eval)`. `evalf_func(cosh_evalf)`. `info_func(exp_info)`. `derivative_func(cosh_deriv)`. `real_part_func(cosh_real_part)`. `imag_part_func(cosh_imag_part)`. `conjugate_←func(cosh_conjugate)`. `latex_name("\\cosh")`)
- static `ex tanh_evalf` (const `ex` &x)
- static `ex tanh_eval` (const `ex` &x)
- static `ex tanh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex tanh_series` (const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex tanh_real_part` (const `ex` &x)
- static `ex tanh_imag_part` (const `ex` &x)
- static `ex tanh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`tanh`, `eval_func(tanh_eval)`. `evalf_func(tanh_evalf)`. `info_func(atan_info)`. `derivative_func(tanh_deriv)`. `series_func(tanh_series)`. `real_part_func(tanh_real_part)`. `imag_part_←func(tanh_imag_part)`. `conjugate_func(tanh_conjugate)`. `latex_name("\\tanh")`)
- static `ex asinh_evalf` (const `ex` &x)
- static `ex asinh_eval` (const `ex` &x)
- static `ex asinh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex asinh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`asinh`, `eval_func(asinh_eval)`. `evalf_func(asinh_evalf)`. `info_func(atan_info)`. `derivative_func(asinh_deriv)`. `conjugate_func(asinh_conjugate)`)
- static `ex acosh_evalf` (const `ex` &x)
- static `ex acosh_eval` (const `ex` &x)
- static `ex acosh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex acosh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`acosh`, `eval_func(acosh_eval)`. `evalf_func(acosh_evalf)`. `info_func(asin_info)`. `derivative_func(acosh_deriv)`. `conjugate_func(acosh_conjugate)`)
- static `ex atanh_evalf` (const `ex` &x)
- static `ex atanh_eval` (const `ex` &x)
- static `ex atanh_deriv` (const `ex` &x, unsigned deriv\_param)
- static `ex atanh_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex atanh_conjugate` (const `ex` &x)
- `REGISTER_FUNCTION` (`atanh`, `eval_func(atanh_eval)`. `evalf_func(atanh_evalf)`. `info_func(asin_info)`. `derivative_func(atanh_deriv)`. `series_func(atanh_series)`. `conjugate_func(atanh_conjugate)`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`integral`, `basic`, `print_func< print_dflt >(&integral::do_print)`. `print_func< print_python >(&integral::do_print)`. `print_func< print_latex >(&integral::do_print_latex)`) `integral`
- `ex subsvalue` (const `ex` &var, const `ex` &value, const `ex` &fun)

- [ex adaptivesimpson](#) (const [ex](#) &x, const [ex](#) &a\_in, const [ex](#) &b\_in, const [ex](#) &f, const [ex](#) &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- [GINAC\\_BIND\\_UNARCHIVER](#) (integral)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (integral)
- [ex ifactor](#) (const [numeric](#) &n)  
*Returns the decomposition of the positive integer n into prime numbers in the form  $lst(p_1, \dots, pr), lst(a_1, \dots, ar)$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .*
- [bool is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const [numeric](#) &n)  
*Returns true if the integer n is either one or the discriminant of a quadratic number field.*
- [numeric kronecker\\_symbol](#) (const [numeric](#) &a, const [numeric](#) &n)  
*Returns the Kronecker symbol a: integer n: integer.*
- [numeric primitive\\_dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- [numeric dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a, const [numeric](#) &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- [numeric generalised\\_Bernoulli\\_number](#) (const [numeric](#) &k, const [numeric](#) &b)  
*The generalised Bernoulli number.*
- [ex Bernoulli\\_polynomial](#) (const [numeric](#) &k, const [ex](#) &x)  
*The Bernoulli polynomials.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< [print\\_context](#) >(&integration\_kernel::do\_print)) integration\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< [print\\_context](#) >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (integration\_kernel)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) (basic\_log\_kernel)

- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([multiple\\_polylog\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([ELi\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Ebar\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([lst](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([matrix](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[matrix::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[matrix::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[matrix::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[matrix::do\\_print\\_python\\_repr](#))) [matrix](#)
- Default ctor.*
- [GINAC\\_BIND\\_UNARCHIVER](#) ([matrix](#))
- [ex lst\\_to\\_matrix](#) (const [lst](#) &l)
- Convert list of lists to matrix.*
- [ex diag\\_matrix](#) (const [lst](#) &l)
- Convert list of diagonal elements to matrix.*
- [ex diag\\_matrix](#) (std::initializer\_list< [ex](#) > l)
- [ex unit\\_matrix](#) (unsigned [r](#), unsigned [c](#))
- Create an r times c unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &base\_name, const std::string &tex\_base\_name)
- Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex reduced\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [c](#))
- Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex sub\\_matrix](#) (const [matrix](#) &m, unsigned [r](#), unsigned [nr](#), unsigned [c](#), unsigned [nc](#))
- Return the nr times nc submatrix starting at position r, c of matrix m.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t nops](#) (const [matrix](#) &m)
- [ex expand](#) (const [matrix](#) &m, unsigned [options](#)=0)
- [ex evalf](#) (const [matrix](#) &m)
- unsigned [rows](#) (const [matrix](#) &m)
- unsigned [cols](#) (const [matrix](#) &m)
- [matrix transpose](#) (const [matrix](#) &m)
- [ex determinant](#) (const [matrix](#) &m, unsigned [options](#)=[determinant\\_algo::automatic](#))
- [ex trace](#) (const [matrix](#) &m)
- [ex charpoly](#) (const [matrix](#) &m, const [ex](#) &lambda)
- [matrix inverse](#) (const [matrix](#) &m)
- [matrix inverse](#) (const [matrix](#) &m, unsigned [algo](#))
- unsigned [rank](#) (const [matrix](#) &m)
- unsigned [rank](#) (const [matrix](#) &m, unsigned [solve\\_algo](#))
- [ex unit\\_matrix](#) (unsigned [x](#))
- Create a x times x unit matrix.*
- [ex symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &base\_name)
- Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([mul](#), [expairseq](#), [print\\_func](#)< [print\\_context](#) >(&[mul::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[mul::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_csrc](#) >(&[mul::do\\_print\\_csrc](#)), [print\\_func](#)< [print\\_tree](#) >(&[mul::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[mul::do\\_print\\_python\\_repr](#))) [mul](#)
- bool [tryfactsubs](#) (const [ex](#) &origfactor, const [ex](#) &patternfactor, int &nummatches, [exmap](#) &repls)

- bool `algebraic_match_mul_with_mul` (const `mul` &*e*, const `ex` &*pat*, `exmap` &*repls*, int `factor`, int &*num\_matches*, const std::vector< bool > &*subsed*, std::vector< bool > &*matched*)  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- `GINAC_BIND_UNARCHIVER` (`mul`)
- `GINAC_DECLARE_UNARCHIVER` (`mul`)
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`ncmul`, `exprseq`, print\_func< `print_context` >(&`ncmul::do_print`), print\_func< `print_tree` >(&`ncmul::do_print_tree`), print\_func< `print_csrc` >(&`ncmul::do_print_csrc`), print\_func< `print_python_repr` >(&`ncmul::do_print_csrc`)) `ncmul`
- `ex reeval_ncmul` (const `exvector` &*v*)
- `ex hold_ncmul` (const `exvector` &*v*)
- `GINAC_BIND_UNARCHIVER` (`ncmul`)
- `GINAC_DECLARE_UNARCHIVER` (`ncmul`)
- static bool `get_first_symbol` (const `ex` &*e*, `ex` &*x*)  
*Return pointer to first symbol found in expression.*
- static void `add_symbol` (const `ex` &*s*, `sym_desc_vec` &*v*)
- static void `collect_symbols` (const `ex` &*e*, `sym_desc_vec` &*v*)
- static void `get_symbol_stats` (const `ex` &*a*, const `ex` &*b*, `sym_desc_vec` &*v*)  
*Collect statistical information about symbols in polynomials.*
- static `numeric lcmcoeff` (const `ex` &*e*, const `numeric` &*l*)
- static `numeric lcm_of_coefficients_denominators` (const `ex` &*e*)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static `ex multiply_lcm` (const `ex` &*e*, const `numeric` &*lcm*)  
*Bring polynomial from Q[X] to Z[X] by multiplying in the previously determined LCM of the coefficient's denominators.*
- `ex quo` (const `ex` &*a*, const `ex` &*b*, const `ex` &*x*, bool *check\_args*)  
*Quotient q(x) of polynomials a(x) and b(x) in Q[x].*
- `ex rem` (const `ex` &*a*, const `ex` &*b*, const `ex` &*x*, bool *check\_args*)  
*Remainder r(x) of polynomials a(x) and b(x) in Q[x].*
- `ex decomp_rational` (const `ex` &*a*, const `ex` &*x*)  
*Decompose rational function a(x)=N(x)/D(x) into P(x)+n(x)/D(x) with degree(n, x) < degree(D, x).*
- `ex prem` (const `ex` &*a*, const `ex` &*b*, const `ex` &*x*, bool *check\_args*)  
*Pseudo-remainder of polynomials a(x) and b(x) in Q[x].*
- `ex sprem` (const `ex` &*a*, const `ex` &*b*, const `ex` &*x*, bool *check\_args*)  
*Sparse pseudo-remainder of polynomials a(x) and b(x) in Q[x].*
- bool `divide` (const `ex` &*a*, const `ex` &*b*, `ex` &*q*, bool *check\_args*)  
*Exact polynomial division of a(X) by b(X) in Q[X].*
- static bool `divide_in_z` (const `ex` &*a*, const `ex` &*b*, `ex` &*q*, `sym_desc_vec::const_iterator` *var*)  
*Exact polynomial division of a(X) by b(X) in Z[X].*
- static `ex sr_gcd` (const `ex` &*a*, const `ex` &*b*, `sym_desc_vec::const_iterator` *var*)  
*Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
- static `ex interpolate` (const `ex` &*gamma*, const `numeric` &*xi*, const `ex` &*x*, int *degree\_hint*=1)  
*xi-adic polynomial interpolation*
- static bool `heur_gcd_z` (`ex` &*res*, const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*, `sym_desc_vec::const_iterator` *var*)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static bool `heur_gcd` (`ex` &*res*, const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*, `sym_desc_vec::const_iterator` *var*)  
*Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
- static `ex gcd_pf_pow` (const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*)
- static `ex gcd_pf_mul` (const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*)
- `ex gcd` (const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*, bool *check\_args*, unsigned *options*)  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials a(X) and b(X) in Z[X].*
- static `ex gcd_pf_pow_pow` (const `ex` &*a*, const `ex` &*b*, `ex` \**ca*, `ex` \**cb*)
- `ex lcm` (const `ex` &*a*, const `ex` &*b*, bool *check\_args*)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in Z[X].*

- static `epvector` `sqrfree_yun` (const `ex` &a, const `symbol` &x)  
*Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
- `ex` `sqrfree` (const `ex` &a, const `lst` &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- `ex` `sqrfree_parfrac` (const `ex` &a, const `symbol` &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- static `ex` `replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex` `replace_with_symbol` (const `ex` &e, `exmap` &repl)  
*Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
- static `ex` `frac_cancel` (const `ex` &n, const `ex` &d)  
*Fraction cancellation.*
- static `ex` `find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)  
*Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
- `ex` `collect_common_factors` (const `ex` &e)  
*Collect common factors in sums.*
- `ex` `resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)  
*Resultant of two expressions  $e_1, e_2$  with respect to symbol  $s$ .*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`numeric`, `basic`, `print_func`< `print_context` >(&`numeric::do_print`).  
`print_func`< `print_latex` >(&`numeric::do_print_latex`). `print_func`< `print_csrc` >(&`numeric::do_print_csrc`).  
`print_func`< `print_csrc_cl_N` >(&`numeric::do_print_csrc_cl_N`). `print_func`< `print_tree` >(&`numeric::do_print_tree`).  
`print_func`< `print_python_repr` >(&`numeric::do_print_python_repr`)) `numeric`  
*default ctor.*
- static const `cln::cl_F` `make_real_float` (const `cln::cl_idcoded_float` &dec)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const `cln::cl_F` `read_real_float` (`std::istream` &s)  
*Read serialized floating point number.*
- `GINAC_BIND_UNARCHIVER` (`numeric`)
- static void `write_real_float` (`std::ostream` &s, const `cln::cl_R` &n)
- static void `print_real_number` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void `print_integer_csrc` (const `print_context` &c, const `cln::cl_I` &x)  
*Helper function to print integer number in C++ source format.*
- static void `print_real_csrc` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print real number in C++ source format.*
- `template`<typename T1 , typename T2 >  
static bool `coerce` (T1 &dst, const T2 &arg)
- `template`<> bool `coerce`< `int`, `cln::cl_I` > (`int` &dst, const `cln::cl_I` &arg)  
*Check if CLN integer can be converted into int.*
- `template`<> bool `coerce`< `unsigned int`, `cln::cl_I` > (`unsigned int` &dst, const `cln::cl_I` &arg)
- static void `print_real_cl_N` (const `print_context` &c, const `cln::cl_R` &x)  
*Helper function to print real number in C++ source format using  $cl\_N$  types.*
- const `numeric` `exp` (const `numeric` &x)  
*Exponential function.*
- const `numeric` `log` (const `numeric` &x)  
*Natural logarithm.*
- const `numeric` `sin` (const `numeric` &x)  
*Numeric sine (trigonometric function).*
- const `numeric` `cos` (const `numeric` &x)  
*Numeric cosine (trigonometric function).*
- const `numeric` `tan` (const `numeric` &x)

- Numeric tangent (trigonometric function).*
- const [numeric asin](#) (const [numeric &x](#))
- Numeric inverse sine (trigonometric function).*
- const [numeric acos](#) (const [numeric &x](#))
- Numeric inverse cosine (trigonometric function).*
- const [numeric atan](#) (const [numeric &x](#))
- Numeric arcustangent.*
- const [numeric atan](#) (const [numeric &y](#), const [numeric &x](#))
- Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const [numeric sinh](#) (const [numeric &x](#))
- Numeric hyperbolic sine (trigonometric function).*
- const [numeric cosh](#) (const [numeric &x](#))
- Numeric hyperbolic cosine (trigonometric function).*
- const [numeric tanh](#) (const [numeric &x](#))
- Numeric hyperbolic tangent (trigonometric function).*
- const [numeric asinh](#) (const [numeric &x](#))
- Numeric inverse hyperbolic sine (trigonometric function).*
- const [numeric acosh](#) (const [numeric &x](#))
- Numeric inverse hyperbolic cosine (trigonometric function).*
- const [numeric atanh](#) (const [numeric &x](#))
- Numeric inverse hyperbolic tangent (trigonometric function).*
- static [cln::cl\\_N Li2\\_series](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))
- Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static [cln::cl\\_N Li2\\_projection](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))
- Folds Li2's argument inside a small rectangle to enhance convergence.*
- const [cln::cl\\_N Li2\\_](#) (const [cln::cl\\_N &value](#))
- Numeric evaluation of Dilogarithm.*
- const [numeric Li2](#) (const [numeric &x](#))
- const [numeric zeta](#) (const [numeric &x](#))
- Numeric evaluation of Riemann's Zeta function.*
- static [cln::float\\_format\\_t guess\\_precision](#) (const [cln::cl\\_N &x](#))
- const [cln::cl\\_N lgamma](#) (const [cln::cl\\_N &x](#))
- The Gamma function.*
- const [numeric lgamma](#) (const [numeric &x](#))
- const [cln::cl\\_N tgamma](#) (const [cln::cl\\_N &x](#))
- const [numeric tgamma](#) (const [numeric &x](#))
- const [numeric psi](#) (const [numeric &x](#))
- The psi function (aka polygamma function).*
- const [numeric psi](#) (const [numeric &n](#), const [numeric &x](#))
- The psi functions (aka polygamma functions).*
- const [numeric factorial](#) (const [numeric &n](#))
- Factorial combinatorial function.*
- const [numeric doublefactorial](#) (const [numeric &n](#))
- The double factorial combinatorial function.*
- const [numeric binomial](#) (const [numeric &n](#), const [numeric &k](#))
- The Binomial coefficients.*
- const [numeric bernoulli](#) (const [numeric &nn](#))
- Bernoulli number.*
- const [numeric fibonacci](#) (const [numeric &n](#))
- Fibonacci number.*
- const [numeric abs](#) (const [numeric &x](#))

- Absolute value.*
- `const numeric mod` (`const numeric &a`, `const numeric &b`)
- Modulus (in positive representation).*
- `const numeric smod` (`const numeric &a_`, `const numeric &b_`)
- Modulus (in symmetric representation).*
- `const numeric irem` (`const numeric &a`, `const numeric &b`)
- Numeric integer remainder.*
- `const numeric irem` (`const numeric &a`, `const numeric &b`, `numeric &q`)
- Numeric integer remainder.*
- `const numeric iquo` (`const numeric &a`, `const numeric &b`)
- Numeric integer quotient.*
- `const numeric iquo` (`const numeric &a`, `const numeric &b`, `numeric &r`)
- Numeric integer quotient.*
- `const numeric gcd` (`const numeric &a`, `const numeric &b`)
- Greatest Common Divisor.*
- `const numeric lcm` (`const numeric &a`, `const numeric &b`)
- Least Common Multiple.*
- `const numeric sqrt` (`const numeric &x`)
- Numeric square root.*
- `const numeric isqrt` (`const numeric &x`)
- Integer numeric square root.*
- `ex PiEvalf` ()
- Floating point evaluation of Archimedes' constant Pi.*
- `ex EulerEvalf` ()
- Floating point evaluation of Euler's constant gamma.*
- `ex CatalanEvalf` ()
- Floating point evaluation of Catalan's constant.*
- `std::ostream & operator<<` (`std::ostream &os`, `const _numeric_digits &e`)
- `GINAC_DECLARE_UNARCHIVER` (`numeric`)
- `const numeric pow` (`const numeric &x`, `const numeric &y`)
- `const numeric inverse` (`const numeric &x`)
- `numeric step` (`const numeric &x`)
- `int csgn` (`const numeric &x`)
- `bool is_zero` (`const numeric &x`)
- `bool is_positive` (`const numeric &x`)
- `bool is_negative` (`const numeric &x`)
- `bool is_integer` (`const numeric &x`)
- `bool is_pos_integer` (`const numeric &x`)
- `bool is_nonneg_integer` (`const numeric &x`)
- `bool is_even` (`const numeric &x`)
- `bool is_odd` (`const numeric &x`)
- `bool is_prime` (`const numeric &x`)
- `bool is_rational` (`const numeric &x`)
- `bool is_real` (`const numeric &x`)
- `bool is_cinteger` (`const numeric &x`)
- `bool is_crational` (`const numeric &x`)
- `int to_int` (`const numeric &x`)
- `long to_long` (`const numeric &x`)
- `double to_double` (`const numeric &x`)
- `const numeric real` (`const numeric &x`)
- `const numeric imag` (`const numeric &x`)
- `const numeric numer` (`const numeric &x`)

- const [numeric denom](#) (const [numeric](#) &x)
- static const [ex exadd](#) (const [ex](#) &lh, const [ex](#) &rh)
  - Used internally by [operator+\(\)](#) to add two ex objects.*
- static const [ex exmul](#) (const [ex](#) &lh, const [ex](#) &rh)
  - Used internally by [operator\\*\(\)](#) to multiply two ex objects.*
- static const [ex exminus](#) (const [ex](#) &lh)
  - Used internally by [operator-\(\)](#) and friends to change the sign of an argument.*
- const [ex operator+](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex operator-](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex operator\\*](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex operator/](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [numeric operator+](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric operator-](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric operator\\*](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric operator/](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- [ex & operator+=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex & operator-=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex & operator\\*=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex & operator/=](#) ([ex](#) &lh, const [ex](#) &rh)
- [numeric & operator+=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric & operator-=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric & operator\\*=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric & operator/=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- const [ex operator+](#) (const [ex](#) &lh)
- const [ex operator-](#) (const [ex](#) &lh)
- const [numeric operator+](#) (const [numeric](#) &lh)
- const [numeric operator-](#) (const [numeric](#) &lh)
- [ex & operator++](#) ([ex](#) &rh)
  - Expression prefix increment.*
- [ex & operator--](#) ([ex](#) &rh)
  - Expression prefix decrement.*
- const [ex operator++](#) ([ex](#) &lh, int)
  - Expression postfix increment.*
- const [ex operator--](#) ([ex](#) &lh, int)
  - Expression postfix decrement.*
- [numeric & operator++](#) ([numeric](#) &rh)
  - Numeric prefix increment.*
- [numeric & operator--](#) ([numeric](#) &rh)
  - Numeric prefix decrement.*
- const [numeric operator++](#) ([numeric](#) &lh, int)
  - Numeric postfix increment.*
- const [numeric operator--](#) ([numeric](#) &lh, int)
  - Numeric postfix decrement.*
- const [relational operator==](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator!=](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator<](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator<=](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator>](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational operator>=](#) (const [ex](#) &lh, const [ex](#) &rh)
- static int [my\\_ios\\_index](#) ()
- static void [my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static [print\\_context](#) \* [get\\_print\\_context](#) (std::ios\_base &s)
- static void [set\\_print\\_context](#) (std::ios\_base &s, const [print\\_context](#) &c)

- static unsigned [get\\_print\\_options](#) (std::ios\_base &s)
- static void [set\\_print\\_options](#) (std::ostream &s, unsigned [options](#))
- std::ostream & [operator<<](#) (std::ostream &os, const [ex](#) &e)
- std::istream & [operator>>](#) (std::istream &is, [ex](#) &e)
- std::ostream & [dflt](#) (std::ostream &os)
- std::ostream & [latex](#) (std::ostream &os)
- std::ostream & [python](#) (std::ostream &os)
- std::ostream & [python\\_repr](#) (std::ostream &os)
- std::ostream & [tree](#) (std::ostream &os)
- std::ostream & [csrc](#) (std::ostream &os)
- std::ostream & [csrc\\_float](#) (std::ostream &os)
- std::ostream & [csrc\\_double](#) (std::ostream &os)
- std::ostream & [csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [index\\_dimensions](#) (std::ostream &os)
- std::ostream & [no\\_index\\_dimensions](#) (std::ostream &os)
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([power](#), [basic](#), print\_func< [print\\_dflt](#) >(&[power::do\\_print\\_dflt](#)), print\_func< [print\\_latex](#) >(&[power::do\\_print\\_latex](#)), print\_func< [print\\_csrc](#) >(&[power::do\\_print\\_csrc](#)), print\_func< [print\\_python](#) >(&[power::do\\_print\\_python](#)), print\_func< [print\\_python\\_repr](#) >(&[power::do\\_print\\_python\\_repr](#)), print\_func< [print\\_csrc\\_cl\\_N](#) >(&[power::do\\_print\\_csrc\\_cl\\_N](#))) [power](#)
- static void [print\\_sym\\_pow](#) (const [print\\_context](#) &c, const [symbol](#) &x, int [exp](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([power](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([power](#))
- [ex pow](#) (const [ex](#) &b, const [ex](#) &e)  
*Symbolic exponentiation.*
- template<typename T1 , typename T2 >  
[ex pow](#) (const T1 &b, const T2 &e)
- [ex sqrt](#) (const [ex](#) &a)  
*Square root expression.*
- template<class T >  
bool [is\\_a](#) (const [print\\_context](#) &obj)  
*Check if obj is a T, including base classes.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([pseries](#), [basic](#), print\_func< [print\\_context](#) >(&[pseries::do\\_print](#)), print\_func< [print\\_latex](#) >(&[pseries::do\\_print\\_latex](#)), print\_func< [print\\_tree](#) >(&[pseries::do\\_print\\_tree](#)), print\_func< [print\\_python](#) >(&[pseries::do\\_print\\_python](#)), print\_func< [print\\_python\\_repr](#) >(&[pseries::do\\_print\\_python\\_repr](#))) [pseries](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex series\\_to\\_poly](#) (const [ex](#) &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [is\\_terminating](#) (const [pseries](#) &s)
- template<typename T >  
[return\\_type\\_t](#) [make\\_return\\_type\\_t](#) (const unsigned [rl](#)=0)
- template<class Alg , class Ctx , class T , class C >  
void [set\\_print\\_func](#) (void f(const T &, const C &c, unsigned))  
*Add or replace a print method.*
- template<class Alg , class Ctx , class T , class C >  
void [set\\_print\\_func](#) (void (T::\*f)(const C &, unsigned))  
*Add or replace a print method.*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([relational](#), [basic](#), print\_func< [print\\_context](#) >(&[relational::do\\_print](#)), print\_func< [print\\_tree](#) >(&[relational::do\\_print\\_tree](#)), print\_func< [print\\_python\\_repr](#) >(&[relational::do\\_print\\_python\\_repr](#))) [relational](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([relational](#))
- static void [print\\_operator](#) (const [print\\_context](#) &c, [relational::operators](#) o)
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([relational](#))

- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symbol](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symbol::do\\_print](#)), [print\\_func](#)< [print\\_latex](#) >(&[symbol::do\\_print\\_latex](#)), [print\\_func](#)< [print\\_tree](#) >(&[symbol::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#) >(&[symbol::do\\_print\\_python\\_repr](#))) [symbol](#)
- static const std::string & [get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GINAC\\_BIND\\_UNARCHIVER](#) ([symbol](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([realsymbol](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([possymbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([symbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([realsymbol](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([possymbol](#))
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([symmetry](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[symmetry::do\\_print](#)), [print\\_func](#)< [print\\_tree](#) >(&[symmetry::do\\_print\\_tree](#))) [symmetry](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([symmetry](#))
- static const [symmetry](#) & [index0](#) ()
- static const [symmetry](#) & [index1](#) ()
- static const [symmetry](#) & [index2](#) ()
- static const [symmetry](#) & [index3](#) ()
- const [symmetry](#) & [not\\_symmetric](#) ()
- const [symmetry](#) & [symmetric2](#) ()
- const [symmetry](#) & [symmetric3](#) ()
- const [symmetry](#) & [symmetric4](#) ()
- const [symmetry](#) & [antisymmetric2](#) ()
- const [symmetry](#) & [antisymmetric3](#) ()
- const [symmetry](#) & [antisymmetric4](#) ()
- int [canonicalize](#) (exvector::iterator v, const [symmetry](#) &[symm](#))  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static [ex symm](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#), bool asymmetric)
- [ex symmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex antisymmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator [last](#))  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([symmetry](#))
- [symmetry sy\\_none](#) ()
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_symm](#) ()
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_anti](#) ()
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry sy\\_cycl](#) ()
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [ex symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression over a set of objects (symbols, indices).*

- [ex antisymmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex symmetrize\\_cyclic](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([tensdelta](#), [tensor](#), print\_func< [print\\_dflt](#) >(&[tensdelta::do\\_print](#)), print\_func< [print\\_latex](#) >(&[tensdelta::do\\_print\\_latex](#))) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([tensmetric](#)
- [print\\_func< print\\_dflt >](#) (&[tensmetric::do\\_print](#)). [print\\_func< print\\_latex >](#)(&[tensmetric](#)
- [GINAC\\_BIND\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensepsilon](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_BIND\\_UNARCHIVER](#) ([spinmetric](#))
- [ex delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensdelta](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([minkmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinmetric](#))
- [GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensepsilon](#))
- unsigned [log2](#) (unsigned n)  
*Integer binary logarithm.*
- const [numeric multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1!p_2!...p_k!)$  where  $n = p_1+p_2+...+p_k$ , i.e.*
- unsigned [rotate\\_left](#) (unsigned n)  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T >  
int [compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [golden\\_ratio\\_hash](#) (uintptr\_t n)  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It >  
int [permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap >  
int [permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Cmp, class Swap >  
void [shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Swap >  
void [cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)

- `template<typename T >`  
`std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T >`  
`std::enable_if<!has\_distance< T >::value, T >::type format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const basic\_multi\_iterator< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter\_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_permutation< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle\_prime< T > &v)`  
*Output operator.*
- `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (wildcard, basic, print_func< print\_context >(&wildcard::do_print).`  
`print_func< print\_tree >(&wildcard::do_print_tree). print_func< print\_python\_repr >(&wildcard::do_print_python_repr))`  
`wildcard`
- `GINAC_BIND_UNARCHIVER (wildcard)`
- `bool haswild (const ex &x)`  
*Check whether x has a wildcard anywhere as a subexpression.*
- `GINAC_DECLARE_UNARCHIVER (wildcard)`
- `ex wild (unsigned label=0)`  
*Create a wildcard object with the specified label.*

## Variables

- static `unarchive\_table\_t unarch\_table\_instance`
- `GiNaC::evalm_map_function map\_evalm`
- `GiNaC::eval_integ_map_function map\_eval\_integ`
- `tensor`
- `const constant Pi ("Pi", PiEvalf, "\\pi", domain::positive)`  
*Pi.*

- const [constant Euler](#) ("Euler", [EulerEvalf](#), "\\gamma\_E", [domain::positive](#))  
*Euler's constant.*
- const [constant Catalan](#) ("Catalan", [CatalanEvalf](#), "G", [domain::positive](#))  
*Catalan's constant.*
- static unsigned const [crtab](#) [256]
- static [library\\_init](#) [library\\_initializer](#)  
*For construction of flyweights, etc.*
- const [basic](#) \* [\\_num0\\_bp](#)
- [idx](#)
- unsigned [force\\_include\\_tgamma](#) = [tgamma\\_SERIAL::serial](#)
- unsigned [force\\_include\\_zeta1](#) = [zeta1\\_SERIAL::serial](#)
- template<> [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\\_T](#)([lst](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[lst::do\\_print](#))). [print\\_func](#)< [print\\_tree](#)>(&[lst::do\\_print\\_tree](#))) template<> bool [Is GINAC\\_BIND\\_UNARCHIVER](#))([lst](#))  
*Specialization of [container::info\(\)](#) for [lst](#).*
- const [numeric](#) [I](#) = [numeric](#)([cln::complex](#)([cln::cl\\_I](#)(0),[cln::cl\\_I](#)(1)))  
*Imaginary unit.*
- [\\_numeric\\_digits](#) [Digits](#)  
*Accuracy in decimal digits.*
- unsigned [next\\_print\\_context\\_id](#) = 0  
*Next free ID for [print\\_context](#) types.*
- const int [version\\_major](#) = [GINACLIB\\_MAJOR\\_VERSION](#)
- const int [version\\_minor](#) = [GINACLIB\\_MINOR\\_VERSION](#)
- const int [version\\_micro](#) = [GINACLIB\\_MICRO\\_VERSION](#)
- const [numeric](#) \* [\\_num\\_120\\_p](#)
- const [ex\\_ex\\_120](#) = [ex](#)(\* [\\_num\\_120\\_p](#))
- const [numeric](#) \* [\\_num\\_60\\_p](#)
- const [ex\\_ex\\_60](#) = [ex](#)(\* [\\_num\\_60\\_p](#))
- const [numeric](#) \* [\\_num\\_48\\_p](#)
- const [ex\\_ex\\_48](#) = [ex](#)(\* [\\_num\\_48\\_p](#))
- const [numeric](#) \* [\\_num\\_30\\_p](#)
- const [ex\\_ex\\_30](#) = [ex](#)(\* [\\_num\\_30\\_p](#))
- const [numeric](#) \* [\\_num\\_25\\_p](#)
- const [ex\\_ex\\_25](#) = [ex](#)(\* [\\_num\\_25\\_p](#))
- const [numeric](#) \* [\\_num\\_24\\_p](#)
- const [ex\\_ex\\_24](#) = [ex](#)(\* [\\_num\\_24\\_p](#))
- const [numeric](#) \* [\\_num\\_20\\_p](#)
- const [ex\\_ex\\_20](#) = [ex](#)(\* [\\_num\\_20\\_p](#))
- const [numeric](#) \* [\\_num\\_18\\_p](#)
- const [ex\\_ex\\_18](#) = [ex](#)(\* [\\_num\\_18\\_p](#))
- const [numeric](#) \* [\\_num\\_15\\_p](#)
- const [ex\\_ex\\_15](#) = [ex](#)(\* [\\_num\\_15\\_p](#))
- const [numeric](#) \* [\\_num\\_12\\_p](#)
- const [ex\\_ex\\_12](#) = [ex](#)(\* [\\_num\\_12\\_p](#))
- const [numeric](#) \* [\\_num\\_11\\_p](#)
- const [ex\\_ex\\_11](#) = [ex](#)(\* [\\_num\\_11\\_p](#))
- const [numeric](#) \* [\\_num\\_10\\_p](#)
- const [ex\\_ex\\_10](#) = [ex](#)(\* [\\_num\\_10\\_p](#))
- const [numeric](#) \* [\\_num\\_9\\_p](#)
- const [ex\\_ex\\_9](#) = [ex](#)(\* [\\_num\\_9\\_p](#))
- const [numeric](#) \* [\\_num\\_8\\_p](#)
- const [ex\\_ex\\_8](#) = [ex](#)(\* [\\_num\\_8\\_p](#))
- const [numeric](#) \* [\\_num\\_7\\_p](#)
- const [ex\\_ex\\_7](#) = [ex](#)(\* [\\_num\\_7\\_p](#))

- `const numeric * _num_6_p`
- `const ex _ex_6 = ex(*_num_6_p)`
- `const numeric * _num_5_p`
- `const ex _ex_5 = ex(*_num_5_p)`
- `const numeric * _num_4_p`
- `const ex _ex_4 = ex(*_num_4_p)`
- `const numeric * _num_3_p`
- `const ex _ex_3 = ex(*_num_3_p)`
- `const numeric * _num_2_p`
- `const ex _ex_2 = ex(*_num_2_p)`
- `const numeric * _num_1_p`
- `const ex _ex_1 = ex(*_num_1_p)`
- `const numeric * _num_1_2_p`
- `const ex _ex_1_2 = ex(*_num_1_2_p)`
- `const numeric * _num_1_3_p`
- `const ex _ex_1_3 = ex(*_num_1_3_p)`
- `const numeric * _num_1_4_p`
- `const ex _ex_1_4 = ex(*_num_1_4_p)`
- `const numeric * _num0_p`
- `const ex _ex0 = ex(*_num0_p)`
- `const numeric * _num1_4_p`
- `const ex _ex1_4 = ex(*_num1_4_p)`
- `const numeric * _num1_3_p`
- `const ex _ex1_3 = ex(*_num1_3_p)`
- `const numeric * _num1_2_p`
- `const ex _ex1_2 = ex(*_num1_2_p)`
- `const numeric * _num1_p`
- `const ex _ex1 = ex(*_num1_p)`
- `const numeric * _num2_p`
- `const ex _ex2 = ex(*_num2_p)`
- `const numeric * _num3_p`
- `const ex _ex3 = ex(*_num3_p)`
- `const numeric * _num4_p`
- `const ex _ex4 = ex(*_num4_p)`
- `const numeric * _num5_p`
- `const ex _ex5 = ex(*_num5_p)`
- `const numeric * _num6_p`
- `const ex _ex6 = ex(*_num6_p)`
- `const numeric * _num7_p`
- `const ex _ex7 = ex(*_num7_p)`
- `const numeric * _num8_p`
- `const ex _ex8 = ex(*_num8_p)`
- `const numeric * _num9_p`
- `const ex _ex9 = ex(*_num9_p)`
- `const numeric * _num10_p`
- `const ex _ex10 = ex(*_num10_p)`
- `const numeric * _num11_p`
- `const ex _ex11 = ex(*_num11_p)`
- `const numeric * _num12_p`
- `const ex _ex12 = ex(*_num12_p)`
- `const numeric * _num15_p`
- `const ex _ex15 = ex(*_num15_p)`
- `const numeric * _num18_p`
- `const ex _ex18 = ex(*_num18_p)`
- `const numeric * _num20_p`

- `const ex _ex20 = ex(*_num20_p)`
- `const numeric * _num24_p`
- `const ex _ex24 = ex(*_num24_p)`
- `const numeric * _num25_p`
- `const ex _ex25 = ex(*_num25_p)`
- `const numeric * _num30_p`
- `const ex _ex30 = ex(*_num30_p)`
- `const numeric * _num48_p`
- `const ex _ex48 = ex(*_num48_p)`
- `const numeric * _num60_p`
- `const ex _ex60 = ex(*_num60_p)`
- `const numeric * _num120_p`
- `const ex _ex120 = ex(*_num120_p)`

### 5.1.1 Typedef Documentation

#### 5.1.1.1 `archive_node_id`

```
typedef unsigned GiNaC::archive_node_id
```

Numerical ID value to refer to an [archive\\_node](#).

#### 5.1.1.2 `archive_atom`

```
typedef unsigned GiNaC::archive_atom
```

Numerical ID value to refer to a string.

#### 5.1.1.3 `synthesize_func`

```
typedef basic *(* GiNaC::synthesize_func) ()
```

#### 5.1.1.4 `unarchive_map_t`

```
typedef std::map<std::string, synthesize_func> GiNaC::unarchive_map_t
```

#### 5.1.1.5 `exvector`

```
typedef std::vector<ex> GiNaC::exvector
```

#### 5.1.1.6 `exset`

```
typedef std::set<ex, ex_is_less> GiNaC::exset
```

### 5.1.1.7 exmap

```
typedef std::map<ex, ex, ex_is_less> GiNaC::exmap
```

### 5.1.1.8 evalffunctype

```
typedef ex(* GiNaC::evalffunctype) ()
```

### 5.1.1.9 FUNCP\_1P

```
typedef double(* GiNaC::FUNCP_1P) (double)
```

Function pointer with one function parameter.

### 5.1.1.10 FUNCP\_2P

```
typedef double(* GiNaC::FUNCP_2P) (double, double)
```

Function pointer with two function parameters.

### 5.1.1.11 FUNCP\_CUBA

```
typedef void(* GiNaC::FUNCP_CUBA) (const int *, const double[], const int *, double[])
```

Function pointer for use with the CUBA library ( <http://www.feynarts.de/cuba>).

### 5.1.1.12 epvector

```
typedef std::vector<expair> GiNaC::epvector
```

expair-vector

### 5.1.1.13 epp

```
typedef epvector::iterator GiNaC::epp
```

expair-vector pointer

### 5.1.1.14 exprseq

```
typedef container<std::vector> GiNaC::exprseq
```

#### 5.1.1.15 paramset

```
typedef std::multiset<unsigned> GiNaC::paramset
```

#### 5.1.1.16 eval\_funcp

```
typedef ex(* GiNaC::eval_funcp) ()
```

#### 5.1.1.17 evalf\_funcp

```
typedef ex(* GiNaC::evalf_funcp) ()
```

#### 5.1.1.18 conjugate\_funcp

```
typedef ex(* GiNaC::conjugate_funcp) ()
```

#### 5.1.1.19 real\_part\_funcp

```
typedef ex(* GiNaC::real_part_funcp) ()
```

#### 5.1.1.20 imag\_part\_funcp

```
typedef ex(* GiNaC::imag_part_funcp) ()
```

#### 5.1.1.21 expand\_funcp

```
typedef ex(* GiNaC::expand_funcp) ()
```

#### 5.1.1.22 derivative\_funcp

```
typedef ex(* GiNaC::derivative_funcp) ()
```

#### 5.1.1.23 expl\_derivative\_funcp

```
typedef ex(* GiNaC::expl_derivative_funcp) ()
```

#### 5.1.1.24 power\_funcp

```
typedef ex(* GiNaC::power_funcp) ()
```

**5.1.1.25 series\_funcp**

```
typedef ex(* GiNaC::series_funcp) ()
```

**5.1.1.26 print\_funcp**

```
typedef void(* GiNaC::print_funcp) ()
```

**5.1.1.27 info\_funcp**

```
typedef bool(* GiNaC::info_funcp) ()
```

**5.1.1.28 eval\_funcp\_1**

```
typedef ex(* GiNaC::eval_funcp_1) (const ex &)
```

**5.1.1.29 evalf\_funcp\_1**

```
typedef ex(* GiNaC::evalf_funcp_1) (const ex &)
```

**5.1.1.30 conjugate\_funcp\_1**

```
typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)
```

**5.1.1.31 real\_part\_funcp\_1**

```
typedef ex(* GiNaC::real_part_funcp_1) (const ex &)
```

**5.1.1.32 imag\_part\_funcp\_1**

```
typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)
```

**5.1.1.33 expand\_funcp\_1**

```
typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)
```

**5.1.1.34 derivative\_funcp\_1**

```
typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.35 expl\_derivative\_funcp\_1

```
typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)
```

#### 5.1.1.36 power\_funcp\_1

```
typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)
```

#### 5.1.1.37 series\_funcp\_1

```
typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)
```

#### 5.1.1.38 print\_funcp\_1

```
typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)
```

#### 5.1.1.39 info\_funcp\_1

```
typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)
```

#### 5.1.1.40 eval\_funcp\_2

```
typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.41 evalf\_funcp\_2

```
typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.42 conjugate\_funcp\_2

```
typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.43 real\_part\_funcp\_2

```
typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)
```

#### 5.1.1.44 imag\_part\_funcp\_2

```
typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)
```

**5.1.1.45 expand\_funcp\_2**

```
typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)
```

**5.1.1.46 derivative\_funcp\_2**

```
typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)
```

**5.1.1.47 expl\_derivative\_funcp\_2**

```
typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)
```

**5.1.1.48 power\_funcp\_2**

```
typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)
```

**5.1.1.49 series\_funcp\_2**

```
typedef ex(* GiNaC::series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
```

**5.1.1.50 print\_funcp\_2**

```
typedef void(* GiNaC::print_funcp_2) (const ex &, const ex &, const print_context &)
```

**5.1.1.51 info\_funcp\_2**

```
typedef bool(* GiNaC::info_funcp_2) (const ex &, const ex &, unsigned)
```

**5.1.1.52 eval\_funcp\_3**

```
typedef ex(* GiNaC::eval_funcp_3) (const ex &, const ex &, const ex &)
```

**5.1.1.53 evalf\_funcp\_3**

```
typedef ex(* GiNaC::evalf_funcp_3) (const ex &, const ex &, const ex &)
```

**5.1.1.54 conjugate\_funcp\_3**

```
typedef ex(* GiNaC::conjugate_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.55 real\_part\_funcp\_3

```
typedef ex(* GiNaC::real_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.56 imag\_part\_funcp\_3

```
typedef ex(* GiNaC::imag_part_funcp_3) (const ex &, const ex &, const ex &)
```

#### 5.1.1.57 expand\_funcp\_3

```
typedef ex(* GiNaC::expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.58 derivative\_funcp\_3

```
typedef ex(* GiNaC::derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.59 expl\_derivative\_funcp\_3

```
typedef ex(* GiNaC::expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol
&)
```

#### 5.1.1.60 power\_funcp\_3

```
typedef ex(* GiNaC::power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.61 series\_funcp\_3

```
typedef ex(* GiNaC::series_funcp_3) (const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.62 print\_funcp\_3

```
typedef void(* GiNaC::print_funcp_3) (const ex &, const ex &, const ex &, const print_context
&)
```

#### 5.1.1.63 info\_funcp\_3

```
typedef bool(* GiNaC::info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.64 eval\_funcp\_4

```
typedef ex(* GiNaC::eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.65 evalf\_funcp\_4

```
typedef ex(* GiNaC::evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.66 conjugate\_funcp\_4

```
typedef ex(* GiNaC::conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.67 real\_part\_funcp\_4

```
typedef ex(* GiNaC::real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.68 imag\_part\_funcp\_4

```
typedef ex(* GiNaC::imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.69 expand\_funcp\_4

```
typedef ex(* GiNaC::expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.70 derivative\_funcp\_4

```
typedef ex(* GiNaC::derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.71 expl\_derivative\_funcp\_4

```
typedef ex(* GiNaC::expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &,
const symbol &)
```

#### 5.1.1.72 power\_funcp\_4

```
typedef ex(* GiNaC::power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex
&)
```

#### 5.1.1.73 series\_funcp\_4

```
typedef ex(* GiNaC::series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const
relational &, int, unsigned)
```

#### 5.1.1.74 print\_funcp\_4

```
typedef void(* GiNaC::print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const
print_context &)
```

#### 5.1.1.75 info\_funcp\_4

```
typedef bool(* GiNaC::info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.76 eval\_funcp\_5

```
typedef ex(* GiNaC::eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.77 evalf\_funcp\_5

```
typedef ex(* GiNaC::evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.78 conjugate\_funcp\_5

```
typedef ex(* GiNaC::conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.79 real\_part\_funcp\_5

```
typedef ex(* GiNaC::real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.80 imag\_part\_funcp\_5

```
typedef ex(* GiNaC::imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.81 expand\_funcp\_5

```
typedef ex(* GiNaC::expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.82 derivative\_funcp\_5

```
typedef ex(* GiNaC::derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.83 expl\_derivative\_funcp\_5

```
typedef ex(* GiNaC::expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.84 power\_funcp\_5

```
typedef ex(* GiNaC::power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &)
```

#### 5.1.1.85 series\_funcp\_5

```
typedef ex(* GiNaC::series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex
&, const relational &, int, unsigned)
```

#### 5.1.1.86 print\_funcp\_5

```
typedef void(* GiNaC::print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex
&, const print_context &)
```

#### 5.1.1.87 info\_funcp\_5

```
typedef bool(* GiNaC::info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex
&, unsigned)
```

#### 5.1.1.88 eval\_funcp\_6

```
typedef ex(* GiNaC::eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &)
```

#### 5.1.1.89 evalf\_funcp\_6

```
typedef ex(* GiNaC::evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &)
```

#### 5.1.1.90 conjugate\_funcp\_6

```
typedef ex(* GiNaC::conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

#### 5.1.1.91 real\_part\_funcp\_6

```
typedef ex(* GiNaC::real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

#### 5.1.1.92 imag\_part\_funcp\_6

```
typedef ex(* GiNaC::imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.93 expand\_funcp\_6**

```
typedef ex(* GiNaC::expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, unsigned)
```

**5.1.1.94 derivative\_funcp\_6**

```
typedef ex(* GiNaC::derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

**5.1.1.95 expl\_derivative\_funcp\_6**

```
typedef ex(* GiNaC::expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const symbol &)
```

**5.1.1.96 power\_funcp\_6**

```
typedef ex(* GiNaC::power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &)
```

**5.1.1.97 series\_funcp\_6**

```
typedef ex(* GiNaC::series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const relational &, int, unsigned)
```

**5.1.1.98 print\_funcp\_6**

```
typedef void(* GiNaC::print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const print_context &)
```

**5.1.1.99 info\_funcp\_6**

```
typedef bool(* GiNaC::info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, unsigned)
```

**5.1.1.100 eval\_funcp\_7**

```
typedef ex(* GiNaC::eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

**5.1.1.101 evalf\_funcp\_7**

```
typedef ex(* GiNaC::evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &)
```

**5.1.1.102 conjugate\_funcp\_7**

```
typedef ex(* GiNaC::conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

**5.1.1.103 real\_part\_funcp\_7**

```
typedef ex(* GiNaC::real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

**5.1.1.104 imag\_part\_funcp\_7**

```
typedef ex(* GiNaC::imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

**5.1.1.105 expand\_funcp\_7**

```
typedef ex(* GiNaC::expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

**5.1.1.106 derivative\_funcp\_7**

```
typedef ex(* GiNaC::derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, unsigned)
```

**5.1.1.107 expl\_derivative\_funcp\_7**

```
typedef ex(* GiNaC::expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.108 power\_funcp\_7**

```
typedef ex(* GiNaC::power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

**5.1.1.109 series\_funcp\_7**

```
typedef ex(* GiNaC::series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const relational &, int, unsigned)
```

**5.1.1.110 print\_funcp\_7**

```
typedef void(* GiNaC::print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const print_context &)
```

#### 5.1.1.111 info\_funcp\_7

```
typedef bool(* GiNaC::info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.112 eval\_funcp\_8

```
typedef ex(* GiNaC::eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &)
```

#### 5.1.1.113 evalf\_funcp\_8

```
typedef ex(* GiNaC::evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &)
```

#### 5.1.1.114 conjugate\_funcp\_8

```
typedef ex(* GiNaC::conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.115 real\_part\_funcp\_8

```
typedef ex(* GiNaC::real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.116 imag\_part\_funcp\_8

```
typedef ex(* GiNaC::imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.117 expand\_funcp\_8

```
typedef ex(* GiNaC::expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.118 derivative\_funcp\_8

```
typedef ex(* GiNaC::derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.119 expl\_derivative\_funcp\_8

```
typedef ex(* GiNaC::expl_derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const symbol &)
```

#### 5.1.1.120 power\_funcp\_8

```
typedef ex(* GiNaC::power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.121 series\_funcp\_8

```
typedef ex(* GiNaC::series_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.122 print\_funcp\_8

```
typedef void(* GiNaC::print_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const print_context &)
```

#### 5.1.1.123 info\_funcp\_8

```
typedef bool(* GiNaC::info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.124 eval\_funcp\_9

```
typedef ex(* GiNaC::eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.125 evalf\_funcp\_9

```
typedef ex(* GiNaC::evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.126 conjugate\_funcp\_9

```
typedef ex(* GiNaC::conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.127 real\_part\_funcp\_9

```
typedef ex(* GiNaC::real_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.128 imag\_part\_funcp\_9

```
typedef ex(* GiNaC::imag_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.129 expand\_funcp\_9**

```
typedef ex(* GiNaC::expand_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.130 derivative\_funcp\_9**

```
typedef ex(* GiNaC::derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.131 expl\_derivative\_funcp\_9**

```
typedef ex(* GiNaC::expl_derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.132 power\_funcp\_9**

```
typedef ex(* GiNaC::power_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.133 series\_funcp\_9**

```
typedef ex(* GiNaC::series_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
```

**5.1.1.134 print\_funcp\_9**

```
typedef void(* GiNaC::print_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**5.1.1.135 info\_funcp\_9**

```
typedef bool(* GiNaC::info_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.136 eval\_funcp\_10**

```
typedef ex(* GiNaC::eval_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.137 evalf\_funcp\_10**

```
typedef ex(* GiNaC::evalf_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.138 conjugate\_funcp\_10**

```
typedef ex(* GiNaC::conjugate_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.139 real\_part\_funcp\_10**

```
typedef ex(* GiNaC::real_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.140 imag\_part\_funcp\_10**

```
typedef ex(* GiNaC::imag_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.141 expand\_funcp\_10**

```
typedef ex(* GiNaC::expand_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.142 derivative\_funcp\_10**

```
typedef ex(* GiNaC::derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.143 expl\_derivative\_funcp\_10**

```
typedef ex(* GiNaC::expl_derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
```

**5.1.1.144 power\_funcp\_10**

```
typedef ex(* GiNaC::power_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.145 series\_funcp\_10**

```
typedef ex(* GiNaC::series_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int,
unsigned)
```

**5.1.1.146 print\_funcp\_10**

```
typedef void(* GiNaC::print_funcp_10) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
```

**5.1.1.147 info\_funcp\_10**

```
typedef bool(* GiNaC::info_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.148 eval\_funcp\_11**

```
typedef ex(* GiNaC::eval_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.149 evalf\_funcp\_11**

```
typedef ex(* GiNaC::evalf_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.150 conjugate\_funcp\_11**

```
typedef ex(* GiNaC::conjugate_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.151 real\_part\_funcp\_11**

```
typedef ex(* GiNaC::real_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.152 imag\_part\_funcp\_11**

```
typedef ex(* GiNaC::imag_part_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.153 expand\_funcp\_11**

```
typedef ex(* GiNaC::expand_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.154 derivative\_funcp\_11**

```
typedef ex(* GiNaC::derivative_funcp_11) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

**5.1.1.155 expl\_derivative\_funcp\_11**

```
typedef ex(* GiNaC::expl_derivative_funcp_11) (const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
symbol &)
```

#### 5.1.1.156 power\_funcp\_11

```
typedef ex(* GiNaC::power_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.157 series\_funcp\_11

```
typedef ex(* GiNaC::series_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &
int, unsigned)
```

#### 5.1.1.158 print\_funcp\_11

```
typedef void(* GiNaC::print_funcp_11) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context
&)
```

#### 5.1.1.159 info\_funcp\_11

```
typedef bool(* GiNaC::info_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
```

#### 5.1.1.160 eval\_funcp\_12

```
typedef ex(* GiNaC::eval_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.161 evalf\_funcp\_12

```
typedef ex(* GiNaC::evalf_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.162 conjugate\_funcp\_12

```
typedef ex(* GiNaC::conjugate_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

#### 5.1.1.163 real\_part\_funcp\_12

```
typedef ex(* GiNaC::real_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.164 imag\_part\_funcp\_12**

```
typedef ex(* GiNaC::imag_part_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
```

**5.1.1.165 expand\_funcp\_12**

```
typedef ex(* GiNaC::expand_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
unsigned)
```

**5.1.1.166 derivative\_funcp\_12**

```
typedef ex(* GiNaC::derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, unsigned)
```

**5.1.1.167 expl\_derivative\_funcp\_12**

```
typedef ex(* GiNaC::expl_derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const symbol &)
```

**5.1.1.168 power\_funcp\_12**

```
typedef ex(* GiNaC::power_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

**5.1.1.169 series\_funcp\_12**

```
typedef ex(* GiNaC::series_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
relational &, int, unsigned)
```

**5.1.1.170 print\_funcp\_12**

```
typedef void(* GiNaC::print_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const print_context &)
```

**5.1.1.171 info\_funcp\_12**

```
typedef bool(* GiNaC::info_funcp_12) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
unsigned)
```

#### 5.1.1.172 eval\_funcp\_13

```
typedef ex(* GiNaC::eval_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

#### 5.1.1.173 evalf\_funcp\_13

```
typedef ex(* GiNaC::evalf_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

#### 5.1.1.174 conjugate\_funcp\_13

```
typedef ex(* GiNaC::conjugate_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

#### 5.1.1.175 real\_part\_funcp\_13

```
typedef ex(* GiNaC::real_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

#### 5.1.1.176 imag\_part\_funcp\_13

```
typedef ex(* GiNaC::imag_part_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &)
```

#### 5.1.1.177 expand\_funcp\_13

```
typedef ex(* GiNaC::expand_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, unsigned)
```

#### 5.1.1.178 derivative\_funcp\_13

```
typedef ex(* GiNaC::derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, unsigned)
```

#### 5.1.1.179 expl\_derivative\_funcp\_13

```
typedef ex(* GiNaC::expl_derivative_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const symbol &)
```

**5.1.1.180 power\_funcp\_13**

```
typedef ex(* GiNaC::power_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.181 series\_funcp\_13**

```
typedef ex(* GiNaC::series_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const relational &, int, unsigned)
```

**5.1.1.182 print\_funcp\_13**

```
typedef void(* GiNaC::print_funcp_13) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const print_context &)
```

**5.1.1.183 info\_funcp\_13**

```
typedef bool(* GiNaC::info_funcp_13) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, unsigned)
```

**5.1.1.184 eval\_funcp\_14**

```
typedef ex(* GiNaC::eval_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.185 evalf\_funcp\_14**

```
typedef ex(* GiNaC::evalf_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.186 conjugate\_funcp\_14**

```
typedef ex(* GiNaC::conjugate_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

**5.1.1.187 real\_part\_funcp\_14**

```
typedef ex(* GiNaC::real_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &)
```

#### 5.1.1.188 imag\_part\_funcp\_14

```
typedef ex(* GiNaC::imag_part_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &)
```

#### 5.1.1.189 expand\_funcp\_14

```
typedef ex(* GiNaC::expand_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

#### 5.1.1.190 derivative\_funcp\_14

```
typedef ex(* GiNaC::derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, unsigned)
```

#### 5.1.1.191 expl\_derivative\_funcp\_14

```
typedef ex(* GiNaC::expl_derivative_funcp_14) (const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const symbol &)
```

#### 5.1.1.192 power\_funcp\_14

```
typedef ex(* GiNaC::power_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &)
```

#### 5.1.1.193 series\_funcp\_14

```
typedef ex(* GiNaC::series_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const relational &, int, unsigned)
```

#### 5.1.1.194 print\_funcp\_14

```
typedef void(* GiNaC::print_funcp_14) (const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &,
const ex &, const ex &, const print_context &)
```

#### 5.1.1.195 info\_funcp\_14

```
typedef bool(* GiNaC::info_funcp_14) (const ex &, const ex &, const ex &, const ex &, const ex
&, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const
ex &, const ex &, unsigned)
```

**5.1.1.196 eval\_funcp\_exvector**

```
typedef ex(* GiNaC::eval_funcp_exvector) (const exvector &)
```

**5.1.1.197 evalf\_funcp\_exvector**

```
typedef ex(* GiNaC::evalf_funcp_exvector) (const exvector &)
```

**5.1.1.198 conjugate\_funcp\_exvector**

```
typedef ex(* GiNaC::conjugate_funcp_exvector) (const exvector &)
```

**5.1.1.199 real\_part\_funcp\_exvector**

```
typedef ex(* GiNaC::real_part_funcp_exvector) (const exvector &)
```

**5.1.1.200 imag\_part\_funcp\_exvector**

```
typedef ex(* GiNaC::imag_part_funcp_exvector) (const exvector &)
```

**5.1.1.201 expand\_funcp\_exvector**

```
typedef ex(* GiNaC::expand_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.202 derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::derivative_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.203 expl\_derivative\_funcp\_exvector**

```
typedef ex(* GiNaC::expl_derivative_funcp_exvector) (const exvector &, const symbol &)
```

**5.1.1.204 power\_funcp\_exvector**

```
typedef ex(* GiNaC::power_funcp_exvector) (const exvector &, const ex &)
```

**5.1.1.205 series\_funcp\_exvector**

```
typedef ex(* GiNaC::series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
```

**5.1.1.206 print\_funcp\_exvector**

```
typedef void(* GiNaC::print_funcp_exvector) (const exvector &, const print_context &)
```

**5.1.1.207 info\_funcp\_exvector**

```
typedef bool(* GiNaC::info_funcp_exvector) (const exvector &, unsigned)
```

**5.1.1.208 exhashmap**

```
template<typename T , class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class
Allocator = std::allocator<std::pair<const ex, T>>>
using GiNaC::exhashmap = std::unordered_map<ex, T, Hash, KeyEqual, Allocator>
```

**5.1.1.209 spmap**

```
typedef std::map<spmapkey, ex> GiNaC::spmap
```

**5.1.1.210 lookup\_map**

```
typedef map<error_and_integral, ex, error_and_integral_is_less> GiNaC::lookup_map
```

**5.1.1.211 lst**

```
typedef container< std::list > GiNaC::lst
```

**5.1.1.212 uintvector**

```
typedef std::vector<std::size_t> GiNaC::uintvector
```

**5.1.1.213 unsignedvector**

```
typedef std::vector<unsigned> GiNaC::unsignedvector
```

**5.1.1.214 exvectorvector**

```
typedef std::vector<exvector> GiNaC::exvectorvector
```

**5.1.1.215 sym\_desc\_vec**

```
typedef std::vector<sym_desc> GiNaC::sym_desc_vec
```

#### 5.1.1.216 digits\_changed\_callback

```
typedef void(* GiNaC::digits_changed_callback) (long)
```

Function pointer to implement callbacks in the case 'Digits' gets changed.

Main purpose of such callbacks is to adjust look-up tables of certain functions to the new precision. Parameter contains the signed difference between new Digits and old Digits.

#### 5.1.1.217 print\_context\_class\_info

```
typedef class_info<print_context_options> GiNaC::print_context_class_info
```

#### 5.1.1.218 registered\_class\_info

```
typedef class_info<registered_class_options> GiNaC::registered_class_info
```

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 anonymous enum

anonymous enum

Enumerator

|                     |
|---------------------|
| callback_registered |
|---------------------|

### 5.1.3 Function Documentation

#### 5.1.3.1 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [1/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 add ,
 expairseq ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree. print_func< print_python_repr > &::do_print_python_repr)
```

#### 5.1.3.2 GINAC\_BIND\_UNARCHIVER() [1/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 add)
```

#### 5.1.3.3 GINAC\_DECLARE\_UNARCHIVER() [1/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 add)
```

#### 5.1.3.4 write\_unsigned()

```
static void GiNaC::write_unsigned (
 std::ostream & os,
 unsigned val) [static]
```

Write unsigned integer quantity to stream.

#### 5.1.3.5 read\_unsigned()

```
static unsigned GiNaC::read_unsigned (
 std::istream & is) [static]
```

Read unsigned integer quantity from stream.

#### 5.1.3.6 operator<<() [1/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const archive_node & n)
```

Write [archive\\_node](#) to binary data stream.

#### 5.1.3.7 operator<<() [2/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const archive & ar)
```

Write archive to binary data stream.

#### 5.1.3.8 operator>>() [1/3]

```
std::istream & GiNaC::operator>> (
 std::istream & is,
 archive_node & n)
```

Read [archive\\_node](#) from binary data stream.

#### 5.1.3.9 operator>>() [2/3]

```
std::istream & GiNaC::operator>> (
 std::istream & is,
 archive & ar)
```

Read archive from binary data stream.

#### 5.1.3.10 find\_factory\_fcn()

```
static synthesize_func GiNaC::find_factory_fcn (
 const std::string & name) [static]
```

References [GiNaC::unarchive\\_table\\_t::find\(\)](#).

Referenced by [GiNaC::archive\\_node::unarchive\(\)](#).

#### 5.1.3.11 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [2/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 basic ,
 void ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print↔
 _tree. print_func< print_python_repr > &::do_print_python_repr)
```

basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by duplicate()), so it can copy the tinfo\_key and the hash value.

#### 5.1.3.12 is\_a() [1/3]

```
template<class T >
bool GiNaC::is_a (
 const basic & obj) [inline]
```

Check if obj is a T, including base classes.

Referenced by [GiNaC::container< C >::subs\(\)](#).

#### 5.1.3.13 is\_exactly\_a() [1/2]

```
template<class T >
bool GiNaC::is_exactly_a (
 const basic & obj) [inline]
```

Check if obj is a T, not including base classes.

**5.1.3.14 dynallocate()** [1/2]

```
template<class B , typename... Args>
B & GiNaC::dynallocate (
 Args &&... args) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function picks the object's ctor based on the given argument types.

This helps the constructor of ex from basic (or a derived class B) because then the constructor doesn't have to duplicate the object onto the heap. See [ex::construct\\_from\\_basic\(const basic &\)](#) for more information.

References [GiNaC::status\\_flags::dynallocated](#).

**5.1.3.15 dynallocate()** [2/2]

```
template<class B >
B & GiNaC::dynallocate (
 std::initializer_list< ex > il) [inline]
```

Constructs a new (class basic or derived) B object on the heap.

This function is needed for [GiNaC](#) classes which have public ctors from initializer lists of expressions (which are not a type and not captured by the variadic template version).

References [GiNaC::status\\_flags::dynallocated](#).

**5.1.3.16 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [3/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 clifford ,
 indexed ,
 print_func< print_dflt > &::do_print_dflt, print_func< print_latex > &::do_↵
print_latex, print_func< print_tree > &::do_print_tree)
```

**5.1.3.17 print\_func< print\_dflt >()** [1/3]

```
GiNaC::print_func< print_dflt > (
 &diracone::do_print) &
```

**5.1.3.18 GINAC\_BIND\_UNARCHIVER()** [2/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 clifford)
```

**5.1.3.19 GINAC\_BIND\_UNARCHIVER()** [3/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 cliffordunit)
```

**5.1.3.20 GINAC\_BIND\_UNARCHIVER()** [4/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 diracone)
```

**5.1.3.21 GINAC\_BIND\_UNARCHIVER()** [5/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 diracgamma)
```

**5.1.3.22 GINAC\_BIND\_UNARCHIVER()** [6/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 diracgamma5)
```

**5.1.3.23 GINAC\_BIND\_UNARCHIVER()** [7/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 diracgammaL)
```

**5.1.3.24 GINAC\_BIND\_UNARCHIVER()** [8/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 diracgammaR)
```

**5.1.3.25 is\_dirac\_slash()**

```
static bool GiNaC::is_dirac_slash (
 const ex & seq0) [static]
```

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), and [GiNaC::clifford::do\\_print\\_latex\(\)](#).

**5.1.3.26 base\_and\_index()**

```
static void GiNaC::base_and_index (
 const ex & c,
 ex & b,
 ex & i) [static]
```

This function decomposes  $\gamma_\mu \rightarrow (1, \mu)$  and  $a_\lambda \rightarrow (a_{i\lambda}, i\lambda)$

**5.1.3.27 dirac\_ONE()**

```
ex GiNaC::dirac_ONE (
 unsigned char r1 = 0)
```

Create a Clifford unity object.

## Parameters

|           |                      |
|-----------|----------------------|
| <i>rl</i> | Representation label |
|-----------|----------------------|

## Returns

newly constructed object

Referenced by [GiNaC::add::coeff\(\)](#).

**5.1.3.28 get\_dim\_uint()**

```
static unsigned GiNaC::get_dim_uint (
 const ex & e) [static]
```

**5.1.3.29 clifford\_unit()**

```
ex GiNaC::clifford_unit (
 const ex & mu,
 const ex & metr,
 unsigned char rl = 0)
```

Create a Clifford unit object.

## Parameters

|             |                                                                        |
|-------------|------------------------------------------------------------------------|
| <i>mu</i>   | Index (must be of class varidx or a derived class)                     |
| <i>metr</i> | Metric (should be indexed, tensmetric or a derived class, or a matrix) |
| <i>rl</i>   | Representation label                                                   |

## Returns

newly constructed Clifford unit object

**5.1.3.30 dirac\_gamma()**

```
ex GiNaC::dirac_gamma (
 const ex & mu,
 unsigned char rl = 0)
```

Create a Dirac gamma object.

## Parameters

|           |                                                    |
|-----------|----------------------------------------------------|
| <i>mu</i> | Index (must be of class varidx or a derived class) |
| <i>rl</i> | Representation label                               |

**Returns**

newly constructed gamma object

**5.1.3.31 `dirac_gamma5()`**

```
ex GiNaC::dirac_gamma5 (
 unsigned char rl = 0)
```

Create a Dirac gamma5 object.

**Parameters**

|           |                      |
|-----------|----------------------|
| <i>rl</i> | Representation label |
|-----------|----------------------|

**Returns**

newly constructed object

**5.1.3.32 `dirac_gammaL()`**

```
ex GiNaC::dirac_gammaL (
 unsigned char rl = 0)
```

Create a Dirac gammaL object.

**Parameters**

|           |                      |
|-----------|----------------------|
| <i>rl</i> | Representation label |
|-----------|----------------------|

**Returns**

newly constructed object

**5.1.3.33 `dirac_gammaR()`**

```
ex GiNaC::dirac_gammaR (
 unsigned char rl = 0)
```

Create a Dirac gammaR object.

**Parameters**

|           |                      |
|-----------|----------------------|
| <i>rl</i> | Representation label |
|-----------|----------------------|

**Returns**

newly constructed object

**5.1.3.34 `dirac_slash()`**

```
ex GiNaC::dirac_slash (
 const ex & e,
 const ex & dim,
 unsigned char rl = 0)
```

Create a term of the form  $e_\mu * \gamma_\mu$  with a unique index  $\mu$ .

**Parameters**

|            |                      |
|------------|----------------------|
| <i>e</i>   | Original expression  |
| <i>dim</i> | Dimension of index   |
| <i>rl</i>  | Representation label |

**5.1.3.35 `get_representation_label()` [1/2]**

```
static unsigned char GiNaC::get_representation_label (
 const return_type_t & ti) [static]
```

Extract representation label from tinfo key (as returned by `return_type_tinfo()`).

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

**5.1.3.36 `trace_string()`**

```
static ex GiNaC::trace_string (
 exvector::const_iterator ix,
 size_t num) [static]
```

Take trace of a string of an even number of Dirac gammas given a vector of indices.

**5.1.3.37 `dirac_trace()` [1/3]**

```
ex GiNaC::dirac_trace (
 const ex & e,
 const std::set< unsigned char > & rls,
 const ex & trONE = 4)
```

Calculate dirac traces over the specified set of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>e</i>     | Expression to take the trace of                           |
| <i>rls</i>   | Set of representation labels                              |
| <i>trONE</i> | Expression to be returned as the trace of the unit matrix |

**5.1.3.38** `dirac_trace()` [2/3]

```
ex GiNaC::dirac_trace (
 const ex & e,
 const lst & rll,
 const ex & trONE = 4)
```

Calculate dirac traces over the specified list of representation labels.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>e</i>     | Expression to take the trace of                           |
| <i>rll</i>   | List of representation labels                             |
| <i>trONE</i> | Expression to be returned as the trace of the unit matrix |

**5.1.3.39** `dirac_trace()` [3/3]

```
ex GiNaC::dirac_trace (
 const ex & e,
 unsigned char rl = 0,
 const ex & trONE = 4)
```

Calculate the trace of an expression containing gamma objects with a specified representation label.

The computed trace is a linear functional that is equal to the usual trace only in  $D = 4$  dimensions. In particular, the functional is not always cyclic in  $D \neq 4$  dimensions when  $\gamma_5$  is involved.

## Parameters

|              |                                                           |
|--------------|-----------------------------------------------------------|
| <i>e</i>     | Expression to take the trace of                           |
| <i>rl</i>    | Representation label                                      |
| <i>trONE</i> | Expression to be returned as the trace of the unit matrix |

**5.1.3.40** `canonicalize_clifford()`

```
ex GiNaC::canonicalize_clifford (
 const ex & e)
```

Bring all products of clifford objects in an expression into a canonical order.

This is not necessarily the most simple form but it will allow to check two expressions for equality.

**5.1.3.41 clifford\_star\_bar()**

```
ex GiNaC::clifford_star_bar (
 const ex & e,
 bool do_bar,
 unsigned options)
```

An auxillary function performing [clifford\\_star\(\)](#) and [clifford\\_bar\(\)](#).

Referenced by [clifford\\_bar\(\)](#), and [clifford\\_star\(\)](#).

**5.1.3.42 clifford\_prime()**

```
ex GiNaC::clifford_prime (
 const ex & e)
```

Automorphism of the Clifford algebra, simply changes signs of all clifford units.

**5.1.3.43 remove\_dirac\_ONE()**

```
ex GiNaC::remove_dirac_ONE (
 const ex & e,
 unsigned char rl = 0,
 unsigned options = 0)
```

Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.

For the default value `rl = 0` remove all of them. Aborts if `e` contains any `clifford_unit` with `representation_label` to be removed.

**Parameters**

|                |                               |
|----------------|-------------------------------|
| <i>e</i>       | Expression to be processed    |
| <i>rl</i>      | Value of representation label |
| <i>options</i> | Defines some internal use     |

**5.1.3.44 clifford\_max\_label()**

```
int GiNaC::clifford_max_label (
 const ex & e,
 bool ignore_ONE = false)
```

Returns the maximal representation label of a clifford object if `e` contains at least one, otherwise returns -1.

**Parameters**

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <i>e</i>          | Expressed to be processed                                            |
| <i>ignore_ONE</i> | defines if <code>clifford_ONE</code> should be ignored in the search |

Referenced by [GiNaC::add::coeff\(\)](#).

#### 5.1.3.45 clifford\_norm()

```
ex GiNaC::clifford_norm (
 const ex & e)
```

Calculation of the norm in the Clifford algebra.

#### 5.1.3.46 clifford\_inverse()

```
ex GiNaC::clifford_inverse (
 const ex & e)
```

Calculation of the inverse in the Clifford algebra.

#### 5.1.3.47 lst\_to\_clifford() [1/2]

```
ex GiNaC::lst_to_clifford (
 const ex & v,
 const ex & mu,
 const ex & metr,
 unsigned char rl = 0)
```

List or vector conversion into the Clifford vector.

##### Parameters

|             |                                                                        |
|-------------|------------------------------------------------------------------------|
| <i>v</i>    | List or vector of coordinates                                          |
| <i>mu</i>   | Index (must be of class varidx or a derived class)                     |
| <i>metr</i> | Metric (should be indexed, tensmetric or a derived class, or a matrix) |
| <i>rl</i>   | Representation label                                                   |

##### Returns

Clifford vector with given components

#### 5.1.3.48 lst\_to\_clifford() [2/2]

```
ex GiNaC::lst_to_clifford (
 const ex & v,
 const ex & e)
```

List or vector conversion into the Clifford vector.

##### Parameters

|          |                               |
|----------|-------------------------------|
| <i>v</i> | List or vector of coordinates |
| <i>e</i> | Clifford unit object          |

**Returns**

Clifford vector with given components

**5.1.3.49 get\_clifford\_comp()**

```
static ex GiNaC::get_clifford_comp (
 const ex & e,
 const ex & c,
 bool root = true) [static]
```

Auxiliary structure to define a function for stripping one Clifford unit from vectors.

Used in [clifford\\_to\\_lst\(\)](#).

**5.1.3.50 clifford\_to\_lst()**

```
lst GiNaC::clifford_to_lst (
 const ex & e,
 const ex & c,
 bool algebraic = true)
```

An inverse function to [lst\\_to\\_clifford\(\)](#).

For given Clifford vector extracts its components with respect to given Clifford unit. Obtained components may contain Clifford units with a different metric. Extraction is based on the algebraic formula  $(e * c.i + c.i * e) / \text{pow}(e.i, 2)$  for non-degenerate cases (i.e. neither  $\text{pow}(e.i, 2) = 0$ ).

**Parameters**

|                  |                                                                                            |
|------------------|--------------------------------------------------------------------------------------------|
| <i>e</i>         | Clifford expression to be decomposed into components                                       |
| <i>c</i>         | Clifford unit defining the metric for splitting (should have numeric dimension of indices) |
| <i>algebraic</i> | Use algebraic or symbolic algorithm for extractions                                        |

**Returns**

List of components of a Clifford vector

**5.1.3.51 clifford\_moebius\_map()** [1/2]

```
ex GiNaC::clifford_moebius_map (
 const ex & a,
 const ex & b,
 const ex & c,
 const ex & d,
 const ex & v,
 const ex & G,
 unsigned char rl = 0)
```

Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b/c d) in linear spaces with arbitrary signature.

The expression is  $(a * x + b)/(c * x + d)$ , where  $x$  is a vector build from list  $v$  with metric  $G$ . (see Jan Cnops. An introduction to {D}irac operators on manifolds, v.24 of Progress in Mathematical Physics. Birkhauser Boston Inc., Boston, MA, 2002.)

#### Parameters

|      |                                                                                            |
|------|--------------------------------------------------------------------------------------------|
| $a$  | (1,1) entry of the defining matrix                                                         |
| $b$  | (1,2) entry of the defining matrix                                                         |
| $c$  | (2,1) entry of the defining matrix                                                         |
| $d$  | (2,2) entry of the defining matrix                                                         |
| $v$  | Vector to be transformed                                                                   |
| $G$  | Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored |
| $rl$ | Representation label                                                                       |

#### Returns

List of components of the transformed vector

#### 5.1.3.52 clifford\_moebius\_map() [2/2]

```
ex GiNaC::clifford_moebius_map (
 const ex & M,
 const ex & v,
 const ex & G,
 unsigned char rl = 0)
```

The second form of Moebius transformations defined by a 2x2 Clifford matrix  $M$  This function takes the transformation matrix  $M$  as a single entity.

#### Parameters

|      |                                                                                            |
|------|--------------------------------------------------------------------------------------------|
| $M$  | the defining matrix                                                                        |
| $v$  | Vector to be transformed                                                                   |
| $G$  | Metric of the surrounding space, may be a Clifford unit then the next parameter is ignored |
| $rl$ | Representation label                                                                       |

#### Returns

List of components of the transformed vector

#### 5.1.3.53 GINAC\_DECLARE\_UNARCHIVER() [2/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 clifford)
```

#### 5.1.3.54 GINAC\_DECLARE\_UNARCHIVER() [3/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 diracone)
```

**5.1.3.55 GINAC\_DECLARE\_UNARCHIVER()** [4/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 cliffordunit)
```

**5.1.3.56 GINAC\_DECLARE\_UNARCHIVER()** [5/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 diracgamma)
```

**5.1.3.57 GINAC\_DECLARE\_UNARCHIVER()** [6/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 diracgamma5)
```

**5.1.3.58 GINAC\_DECLARE\_UNARCHIVER()** [7/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 diracgammaL)
```

**5.1.3.59 GINAC\_DECLARE\_UNARCHIVER()** [8/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 diracgammaR)
```

**5.1.3.60 is\_clifford\_tinfo()**

```
bool GiNaC::is_clifford_tinfo (
 const return_type_t & ti) [inline]
```

Check whether a given [return\\_type\\_t](#) object (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a clifford object (with an arbitrary representation label).

**Parameters**

|           |           |
|-----------|-----------|
| <i>ti</i> | tinfo key |
|-----------|-----------|

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [GiNaC::ncmul::conjugate\(\)](#).

**5.1.3.61 clifford\_bar()**

```
ex GiNaC::clifford_bar (
 const ex & e) [inline]
```

Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.

References [clifford\\_star\\_bar\(\)](#).

#### 5.1.3.62 clifford\_star()

```
ex GiNaC::clifford_star (
 const ex & e) [inline]
```

Reversion of the Clifford algebra, reverse the order of all clifford units in ncmul.

References [clifford\\_star\\_bar\(\)](#).

#### 5.1.3.63 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [4/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 su3one ,
 tensor ,
 print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
 latex)
```

#### 5.1.3.64 print\_func< print\_dflt >() [2/3]

```
GiNaC::print_func< print_dflt > (
 &su3t::do_print) &
```

#### 5.1.3.65 GINAC\_BIND\_UNARCHIVER() [9/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 color)
```

#### 5.1.3.66 GINAC\_BIND\_UNARCHIVER() [10/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 su3one)
```

#### 5.1.3.67 GINAC\_BIND\_UNARCHIVER() [11/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 su3t)
```

#### 5.1.3.68 GINAC\_BIND\_UNARCHIVER() [12/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 su3f)
```

**5.1.3.69 GINAC\_BIND\_UNARCHIVER()** [13/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 su3d)
```

**5.1.3.70 permute\_free\_index\_to\_front()**

```
static ex GiNaC::permute_free_index_to_front (
 const exvector & iv3,
 const exvector & iv2,
 int & sig) [static]
```

Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.

**Parameters**

|            |                                              |
|------------|----------------------------------------------|
| <i>iv3</i> | Vector of 3 indices                          |
| <i>iv2</i> | Vector of 2 indices, must be a subset of iv3 |
| <i>sig</i> | Returns sign introduced by index permutation |

**Returns**

the free index (the one that is in iv3 but not in iv2)

References [GINAC\\_ASSERT](#), and [TEST\\_PERMUTATION](#).

Referenced by [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

**5.1.3.71 color\_ONE()**

```
ex GiNaC::color_ONE (
 unsigned char rl = 0)
```

Create the su(3) unity element.

This is an indexed object, although it has no indices.

**Parameters**

|           |                      |
|-----------|----------------------|
| <i>rl</i> | Representation label |
|-----------|----------------------|

**Returns**

newly constructed unity element

Referenced by [GiNaC::su3t::contract\\_with\(\)](#).

### 5.1.3.72 `color_T()`

```
ex GiNaC::color_T (
 const ex & a,
 unsigned char rl = 0)
```

Create an  $su(3)$  generator.

#### Parameters

|           |                      |
|-----------|----------------------|
| <i>a</i>  | Index                |
| <i>rl</i> | Representation label |

#### Returns

newly constructed unity generator

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), and [GiNaC::su3d::contract\\_with\(\)](#).

### 5.1.3.73 `color_f()`

```
ex GiNaC::color_f (
 const ex & a,
 const ex & b,
 const ex & c)
```

Create an  $su(3)$  antisymmetric structure constant.

#### Parameters

|          |              |
|----------|--------------|
| <i>a</i> | First index  |
| <i>b</i> | Second index |
| <i>c</i> | Third index  |

#### Returns

newly constructed structure constant

References [antisymmetric3\(\)](#), and [c](#).

Referenced by [color\\_h\(\)](#).

### 5.1.3.74 `color_d()`

```
ex GiNaC::color_d (
 const ex & a,
 const ex & b,
 const ex & c)
```

Create an  $su(3)$  symmetric structure constant.

## Parameters

|          |              |
|----------|--------------|
| <i>a</i> | First index  |
| <i>b</i> | Second index |
| <i>c</i> | Third index  |

## Returns

newly constructed structure constant

References [c](#), and [symmetric3\(\)](#).

Referenced by [color\\_h\(\)](#).

**5.1.3.75 color\_h()**

```
ex GiNaC::color_h (
 const ex & a,
 const ex & b,
 const ex & c)
```

This returns the linear combination d.a.b.c+l\*f.a.b.c.

References [c](#), [color\\_d\(\)](#), [color\\_f\(\)](#), and [l](#).

Referenced by [color\\_trace\(\)](#).

**5.1.3.76 is\_color\_tinfo()**

```
static bool GiNaC::is_color_tinfo (
 const return_type_t & ti) [static]
```

Check whether a given tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)) is that of a color object (with an arbitrary representation label).

References [GiNaC::return\\_type\\_t::tinfo](#).

Referenced by [color\\_trace\(\)](#).

**5.1.3.77 get\_representation\_label()** [2/2]

```
static unsigned char GiNaC::get_representation_label (
 const return_type_t & ti) [static]
```

Extract representation label from tinfo key (as returned by [return\\_type\\_tinfo\(\)](#)).

References [GiNaC::return\\_type\\_t::rl](#).

**5.1.3.78 color\_trace()** [1/3]

```
ex GiNaC::color_trace (
 const ex & e,
 const std::set< unsigned char > & rls)
```

Calculate color traces over the specified set of representation labels.

## Parameters

|            |                                 |
|------------|---------------------------------|
| <i>e</i>   | Expression to take the trace of |
| <i>rls</i> | Set of representation labels    |

References [\\_ex0](#), [\\_ex1](#), [\\_ex3](#), [color\\_h\(\)](#), [color\\_T\(\)](#), [color\\_trace\(\)](#), [delta\\_tensor\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_representation\\_label\(\)](#), [is\\_color\\_tinfo\(\)](#), [GiNaC::ex::map\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#)

Referenced by [color\\_trace\(\)](#), [color\\_trace\(\)](#), [color\\_trace\(\)](#), and [GiNaC::su3t::contract\\_with\(\)](#).

**5.1.3.79 color\_trace()** [2/3]

```
ex GiNaC::color_trace (
 const ex & e,
 const lst & rll)
```

Calculate color traces over the specified list of representation labels.

## Parameters

|            |                                 |
|------------|---------------------------------|
| <i>e</i>   | Expression to take the trace of |
| <i>rll</i> | List of representation labels   |

References [color\\_trace\(\)](#), and [GiNaC::info\\_flags::nonnegint](#).

**5.1.3.80 color\_trace()** [3/3]

```
ex GiNaC::color_trace (
 const ex & e,
 unsigned char rl = 0)
```

Calculate the trace of an expression containing color objects with a specified representation label.

## Parameters

|           |                                 |
|-----------|---------------------------------|
| <i>e</i>  | Expression to take the trace of |
| <i>rl</i> | Representation label            |

References [color\\_trace\(\)](#).

**5.1.3.81 GINAC\_DECLARE\_UNARCHIVER()** [9/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 color)
```

**5.1.3.82 GINAC\_DECLARE\_UNARCHIVER()** [10/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 su3one)
```

**5.1.3.83 GINAC\_DECLARE\_UNARCHIVER()** [11/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 su3t)
```

**5.1.3.84 GINAC\_DECLARE\_UNARCHIVER()** [12/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 su3f)
```

**5.1.3.85 GINAC\_DECLARE\_UNARCHIVER()** [13/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 su3d)
```

**5.1.3.86 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [5/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 constant ,
 basic ,
 print_func< print_context > &::do_print, print_func< print_latex > &::do_↵
print_latex, print_func< print_tree > &::do_print_tree, print_func< print_python_repr > &↵
::do_print_python_repr) const
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.87 GINAC\_BIND\_UNARCHIVER()** [14/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 constant)
```

**5.1.3.88 GINAC\_DECLARE\_UNARCHIVER()** [14/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 constant)
```

**5.1.3.89 crc32()**

```
static unsigned GiNaC::crc32 (
 const char * c,
 const unsigned len,
 const unsigned crcinit) [static]
```

References [c](#), [crctab](#), and [len](#).

### 5.1.3.90 `are_ex_trivially_equal()`

```
bool GiNaC::are_ex_trivially_equal (
 const ex & e1,
 const ex & e2) [inline]
```

Compare two objects of class quickly without doing a deep tree traversal.

#### Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

Referenced by [GiNaC::expair::conjugate\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::make\\_flat\\_inserter::make\\_flat\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::power::map\(\)](#), [GiNaC::relational::map\(\)](#), [GiNaC::internal::\\_iter\\_rep::operator==\(\)](#), [GiNaC::const\\_iterator::operator==\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), and [GiNaC::expairseq::subschildren\(\)](#).

### 5.1.3.91 `operator<<()` [3/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const exvector & e)
```

References [get\\_print\\_context\(\)](#).

### 5.1.3.92 `operator<<()` [4/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const exset & e)
```

References [get\\_print\\_context\(\)](#).

### 5.1.3.93 `operator<<()` [5/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const exmap & e)
```

References [get\\_print\\_context\(\)](#).

**5.1.3.94 nops()** [1/2]

```
size_t GiNaC::nops (
 const ex & thisex) [inline]
```

References [GiNaC::ex::nops\(\)](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::basic::derivative\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\(\)](#), [GiNaC::basic::eval\\_integ\(\)](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::basic::evalm\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::basic::subs\(\)](#).

**5.1.3.95 expand()** [1/2]

```
ex GiNaC::expand (
 const ex & thisex,
 unsigned options = 0) [inline]
```

References [GiNaC::ex::expand\(\)](#), and [options](#).

Referenced by [GiNaC::basic::collect\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [log\\_expand\(\)](#), [prem\(\)](#), [quo\(\)](#), [rem\(\)](#), and [sprem\(\)](#).

**5.1.3.96 conjugate()**

```
ex GiNaC::conjugate (
 const ex & thisex) [inline]
```

References [GiNaC::ex::conjugate\(\)](#).

Referenced by [conjugate\\_expl\\_derivative\(\)](#).

**5.1.3.97 real\_part()**

```
ex GiNaC::real_part (
 const ex & thisex) [inline]
```

References [GiNaC::ex::real\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [log\\_imag\\_part\(\)](#), [real\\_part\\_expl\\_derivative\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

**5.1.3.98 imag\_part()**

```
ex GiNaC::imag_part (
 const ex & thisex) [inline]
```

References [GiNaC::ex::imag\\_part\(\)](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [imag\\_part\\_expl\\_derivative\(\)](#), [log\\_imag\\_part\(\)](#), [sin\\_imag\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

#### 5.1.3.99 has()

```
bool GiNaC::has (
 const ex & thisex,
 const ex & pattern,
 unsigned options = 0) [inline]
```

References [GiNaC::ex::has\(\)](#), and [options](#).

Referenced by [GiNaC::basic::is\\_polynomial\(\)](#).

#### 5.1.3.100 find()

```
bool GiNaC::find (
 const ex & thisex,
 const ex & pattern,
 exset & found) [inline]
```

References [GiNaC::ex::find\(\)](#).

#### 5.1.3.101 is\_polynomial()

```
bool GiNaC::is_polynomial (
 const ex & thisex,
 const ex & vars) [inline]
```

References [GiNaC::ex::is\\_polynomial\(\)](#).

#### 5.1.3.102 degree()

```
int GiNaC::degree (
 const ex & thisex,
 const ex & s) [inline]
```

References [GiNaC::ex::degree\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.103 ldegree()

```
int GiNaC::ldegree (
 const ex & thisex,
 const ex & s) [inline]
```

References [GiNaC::ex::ldegree\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#).

#### 5.1.3.104 `coeff()`

```
ex GiNaC::coeff (
 const ex & thisex,
 const ex & s,
 int n = 1) [inline]
```

References [GiNaC::ex::coeff\(\)](#), and [n](#).

Referenced by [GiNaC::basic::collect\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [log\\_series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.105 `numer()` [1/2]

```
ex GiNaC::numer (
 const ex & thisex) [inline]
```

References [GiNaC::ex::numer\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [print\\_real\\_csrc\(\)](#), [GiNaC::power::real\\_part\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.106 `denom()` [1/2]

```
ex GiNaC::denom (
 const ex & thisex) [inline]
```

References [GiNaC::ex::denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcmcoeff\(\)](#), [print\\_real\\_csrc\(\)](#), [replace\\_with\\_symbol\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.107 `numer_denom()`

```
ex GiNaC::numer_denom (
 const ex & thisex) [inline]
```

References [GiNaC::ex::numer\\_denom\(\)](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.108 `normal()`

```
ex GiNaC::normal (
 const ex & thisex) [inline]
```

References [GiNaC::ex::normal\(\)](#).

Referenced by [fsolve\(\)](#), [GiNaC::normal\\_map\\_function::operator\(\)\(\)](#), and [replace\\_with\\_symbol\(\)](#).

**5.1.3.109 to\_rational()**

```
ex GiNaC::to_rational (
 const ex & thisex,
 exmap & repl) [inline]
```

References [GiNaC::ex::to\\_rational\(\)](#).

**5.1.3.110 to\_polynomial()**

```
ex GiNaC::to_polynomial (
 const ex & thisex,
 exmap & repl) [inline]
```

References [GiNaC::ex::to\\_polynomial\(\)](#).

**5.1.3.111 collect()**

```
ex GiNaC::collect (
 const ex & thisex,
 const ex & s,
 bool distributed = false) [inline]
```

References [GiNaC::ex::collect\(\)](#).

Referenced by [GiNaC::basic::collect\(\)](#).

**5.1.3.112 eval()**

```
ex GiNaC::eval (
 const ex & thisex) [inline]
```

References [GiNaC::ex::eval\(\)](#).

**5.1.3.113 evalf() [1/2]**

```
ex GiNaC::evalf (
 const ex & thisex) [inline]
```

References [GiNaC::ex::evalf\(\)](#).

Referenced by [beta\\_evalf\(\)](#), [eta\\_evalf\(\)](#), [GiNaC::evalf\\_map\\_function::operator\(\)\(\)](#), and [zeta2\\_evalf\(\)](#).

**5.1.3.114 evalm()**

```
ex GiNaC::evalm (
 const ex & thisex) [inline]
```

References [GiNaC::ex::evalm\(\)](#).

Referenced by [GiNaC::evalm\\_map\\_function::operator\(\)\(\)](#).

#### 5.1.3.115 eval\_integ()

```
ex GiNaC::eval_integ (
 const ex & thisex) [inline]
```

References [GiNaC::ex::eval\\_integ\(\)](#).

Referenced by [GiNaC::eval\\_integ\\_map\\_function::operator\(\)\(\)](#).

#### 5.1.3.116 diff()

```
ex GiNaC::diff (
 const ex & thisex,
 const symbol & s,
 unsigned nth = 1) [inline]
```

References [GiNaC::ex::diff\(\)](#).

Referenced by [GiNaC::derivative\\_map\\_function::operator\(\)\(\)](#).

#### 5.1.3.117 series()

```
ex GiNaC::series (
 const ex & thisex,
 const ex & r,
 int order,
 unsigned options = 0) [inline]
```

References [options](#), [order](#), [r](#), and [GiNaC::ex::series\(\)](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [Li2\\_series\(\)](#), and [log\\_series\(\)](#).

#### 5.1.3.118 match()

```
bool GiNaC::match (
 const ex & thisex,
 const ex & pattern,
 exmap & repl_lst) [inline]
```

References [GiNaC::ex::match\(\)](#).

Referenced by [GiNaC::basic::has\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

#### 5.1.3.119 simplify\_indexed() [1/3]

```
ex GiNaC::simplify_indexed (
 const ex & thisex,
 unsigned options = 0) [inline]
```

References [options](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

Referenced by [GiNaC::ex::simplify\\_indexed\(\)](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

**5.1.3.120 simplify\_indexed()** [2/3]

```
ex GiNaC::simplify_indexed (
 const ex & thisex,
 const scalar_products & sp,
 unsigned options = 0) [inline]
```

References [options](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

**5.1.3.121 symmetrize()** [1/4]

```
ex GiNaC::symmetrize (
 const ex & thisex) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

Referenced by [idx\\_symmetrization\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**5.1.3.122 symmetrize()** [2/4]

```
ex GiNaC::symmetrize (
 const ex & thisex,
 const lst & l) [inline]
```

References [GiNaC::ex::symmetrize\(\)](#).

**5.1.3.123 antisymmetrize()** [1/4]

```
ex GiNaC::antisymmetrize (
 const ex & thisex) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), and [antisymmetrize\(\)](#).

**5.1.3.124 antisymmetrize()** [2/4]

```
ex GiNaC::antisymmetrize (
 const ex & thisex,
 const lst & l) [inline]
```

References [GiNaC::ex::antisymmetrize\(\)](#).

**5.1.3.125 symmetrize\_cyclic()** [1/4]

```
ex GiNaC::symmetrize_cyclic (
 const ex & thisex) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::ex::symmetrize\\_cyclic\(\)](#), and [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**5.1.3.126 symmetrize\_cyclic()** [2/4]

```
ex GiNaC::symmetrize_cyclic (
 const ex & thisex,
 const lst & l) [inline]
```

References [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**5.1.3.127 op()**

```
ex GiNaC::op (
 const ex & thisex,
 size_t i) [inline]
```

References [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::basic::has\(\)](#), [Li2\\_series\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [rename\\_dummy\\_indices\(\)](#), and [GiNaC::basic::subs\(\)](#).

**5.1.3.128 lhs()**

```
ex GiNaC::lhs (
 const ex & thisex) [inline]
```

References [GiNaC::ex::lhs\(\)](#).

**5.1.3.129 rhs()**

```
ex GiNaC::rhs (
 const ex & thisex) [inline]
```

References [GiNaC::ex::rhs\(\)](#).

Referenced by [lsolve\(\)](#), [GiNaC::ptr< T >::operator!=\(\)](#), [GiNaC::ptr< T >::operator==\(\)](#), [GiNaC::matrix::solve\(\)](#), and [sqrtfree\\_parfrac\(\)](#).

**5.1.3.130 is\_zero()** [1/2]

```
bool GiNaC::is_zero (
 const ex & thisex) [inline]
```

References [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [cosh\\_eval\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::relational::operator safe\\_b](#), [GiNaC::matrix::pivot\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

**5.1.3.131 swap()** [1/2]

```
void GiNaC::swap (
 ex & e1,
 ex & e2) [inline]
```

References [GiNaC::ex::swap\(\)](#).

Referenced by [permutation\\_sign\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

**5.1.3.132 subs()** [1/3]

```
ex GiNaC::subs (
 const ex & thisex,
 const exmap & m,
 unsigned options = 0) [inline]
```

References [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [Li2\\_series\(\)](#).

**5.1.3.133 subs()** [2/3]

```
ex GiNaC::subs (
 const ex & thisex,
 const lst & ls,
 const lst & lr,
 unsigned options = 0) [inline]
```

References [lr](#), [options](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.134 subs()** [3/3]

```
ex GiNaC::subs (
 const ex & thisex,
 const ex & e,
 unsigned options = 0) [inline]
```

References [options](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.135 is\_a()** [2/3]

```
template<class T >
bool GiNaC::is_a (
 const ex & obj) [inline]
```

Check if ex is a handle to a T, including base classes.

**5.1.3.136 is\_exactly\_a()** [2/2]

```
template<class T >
bool GiNaC::is_exactly_a (
 const ex & obj) [inline]
```

Check if *ex* is a handle to a T, not including base classes.

**5.1.3.137 ex\_to()**

```
template<class T >
const T & GiNaC::ex_to (
 const ex & e) [inline]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of *e* first. Also, you shouldn't cache the returned reference because [GiNaC](#)'s garbage collector may destroy the referenced object any time it's used in another expression.

**Parameters**

|          |            |
|----------|------------|
| <i>e</i> | expression |
|----------|------------|

**Returns**

reference to object of class T

**See also**

[is\\_exactly\\_a<class T>\(\)](#)

**5.1.3.138 compile\_ex()** [1/3]

```
void GiNaC::compile_ex (
 const ex & expr,
 const symbol & sym,
 FUNCP_1P & fp,
 const std::string filename = "")
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type `FUNCP_1P`.

**Parameters**

|                 |                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------|
| <i>expr</i>     | Expression to be compiled                                                                                   |
| <i>sym</i>      | Symbol from the expression to become the function parameter                                                 |
| <i>fp</i>       | Returned function pointer                                                                                   |
| <i>filename</i> | Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted |

**5.1.3.139 compile\_ex()** [2/3]

```
void GiNaC::compile_ex (
 const ex & expr,
 const symbol & sym1,
 const symbol & sym2,
 FUNCP_2P & fp,
 const std::string filename = "")
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_2P.

**Parameters**

|                 |                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------|
| <i>expr</i>     | Expression to be compiled                                                                                   |
| <i>sym1</i>     | Symbol from the expression to become the first function parameter                                           |
| <i>sym2</i>     | Symbol from the expression to become the second function parameter                                          |
| <i>fp</i>       | Returned function pointer                                                                                   |
| <i>filename</i> | Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted |

**5.1.3.140 compile\_ex()** [3/3]

```
void GiNaC::compile_ex (
 const lst & exprs,
 const lst & syms,
 FUNCP_CUBA & fp,
 const std::string filename = "")
```

Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.

The function pointer has type FUNCP\_CUBA.

**Parameters**

|                 |                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------|
| <i>exprs</i>    | List of expression to be compiled                                                                           |
| <i>syms</i>     | Symbols from the expression to become the function parameters                                               |
| <i>fp</i>       | Returned function pointer                                                                                   |
| <i>filename</i> | Name of the intermediate source code and so-file. If supplied, these intermediate files will not be deleted |

**5.1.3.141 link\_ex()** [1/3]

```
void GiNaC::link_ex (
 const std::string filename,
 FUNCP_1P & fp)
```

Opens an existing so-file and returns a function pointer of type FUNC\_P\_1P to the contained function.

The so-file has to be generated by compile\_ex in advance.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>filename</i> | Name of the so-file to open and link |
| <i>fp</i>       | Returned function pointer            |

#### 5.1.3.142 link\_ex() [2/3]

```
void GiNaC::link_ex (
 const std::string filename,
 FUNC_P_2P & fp)
```

Opens an existing so-file and returns a function pointer of type FUNC\_P\_2P to the contained function.

The so-file has to be generated by compile\_ex in advance.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>filename</i> | Name of the so-file to open and link |
| <i>fp</i>       | Returned function pointer            |

#### 5.1.3.143 link\_ex() [3/3]

```
void GiNaC::link_ex (
 const std::string filename,
 FUNC_P_CUBA & fp)
```

Opens an existing so-file and returns a function pointer of type FUNC\_P\_CUBA to the contained function.

The so-file has to be generated by compile\_ex in advance.

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>filename</i> | Name of the so-file to open and link |
| <i>fp</i>       | Returned function pointer            |

#### 5.1.3.144 unlink\_ex()

```
void GiNaC::unlink_ex (
 const std::string filename)
```

Closes all linked .so files that have the supplied filename.

## Parameters

|                 |                              |
|-----------------|------------------------------|
| <i>filename</i> | Name of the so-file to close |
|-----------------|------------------------------|

**5.1.3.145 swap()** [2/2]

```
void GiNaC::swap (
 expair & e1,
 expair & e2) [inline]
```

References [GiNaC::expair::swap\(\)](#).

**5.1.3.146 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [6/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 expairseq ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
 _tree)
```

**5.1.3.147 conjugateepvector()**

```
epvector * GiNaC::conjugateepvector (
 const epvector &)
```

Complex conjugate every element of an epvector.

Returns zero if this does not change anything.

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [x](#).

Referenced by [GiNaC::expairseq::conjugate\(\)](#), and [GiNaC::pseries::conjugate\(\)](#).

**5.1.3.148 factor()**

```
ex GiNaC::factor (
 const ex & poly,
 unsigned options)
```

Interface function to the outside world.

Factorizes univariate and multivariate polynomials.

It uses [factor1\(\)](#) on each of the explicitly present factors of poly.

The default option is [factor\\_options::polynomial](#), which means that [factor\(\)](#) will only factorize an expression if it is a proper polynomial (i.e. the flag [info\\_flags::polynomial](#) is set). Given the option [factor\\_options::all](#), [factor\(\)](#) will factorize all subexpressions, e.g. polynomials containing functions or polynomials inside function arguments.

## Parameters

|    |                |                                           |
|----|----------------|-------------------------------------------|
| in | <i>poly</i>    | expression to factorize                   |
| in | <i>options</i> | see <a href="#">GiNaC::factor_options</a> |

## Returns

factorized expression

References [options](#), [poly](#), and [pow\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [collect\\_common\\_factors\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::mul::series\(\)](#), and [sqrtfree\\_parfrac\(\)](#).

## 5.1.3.149 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [7/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 fail ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
 _tree)
```

## 5.1.3.150 GINAC\_DECLARE\_UNARCHIVER() [15/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 fail)
```

## 5.1.3.151 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [8/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 fderivative ,
 function ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
 print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
 print_tree)
```

## 5.1.3.152 GINAC\_BIND\_UNARCHIVER() [15/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 fderivative)
```

## 5.1.3.153 GINAC\_DECLARE\_UNARCHIVER() [16/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 fderivative)
```

**5.1.3.154 GINAC\_BIND\_UNARCHIVER()** [16/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 function)
```

**5.1.3.155 GINAC\_DECLARE\_UNARCHIVER()** [17/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 function)
```

**5.1.3.156 is\_the\_function()**

```
template<typename T >
bool GiNaC::is_the_function (
 const ex & x) [inline]
```

References [x](#).

**5.1.3.157 make\_hash\_seed()**

```
static unsigned GiNaC::make_hash_seed (
 const std::type_info & tinfo) [inline], [static]
```

We need a hash function which gives different values for objects of different types.

Hence we need some unique integer for each type. Fortunately, standard C++ RTTI class 'type\_info' stores a pointer to mangled type name. Normally this pointer is the same for all objects of the same type (although it changes from run to run), so it can be used for computing hashes. However, on some platforms (such as woe32) the pointer returned by type\_info::name() might be different even for objects of the same type! Hence we need to resort to comparing string representation of the (mangled) type names. This is quite expensive, so we compare crc32 hashes of those strings. We might get more hash collisions (and slower evaluation as a result), but being a bit slower is much better than being wrong.

References [golden\\_ratio\\_hash\(\)](#).

Referenced by [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), and [GiNaC::wildcard::calchash\(\)](#).

**5.1.3.158 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [9/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 idx ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
print_tree)
```

**5.1.3.159 print\_func< print\_context >()**

```
GiNaC::print_func< print_context > (
 &varidx::do_print) &
```

**5.1.3.160 GINAC\_BIND\_UNARCHIVER()** [17/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 idx)
```

**5.1.3.161 GINAC\_BIND\_UNARCHIVER()** [18/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 varidx)
```

**5.1.3.162 GINAC\_BIND\_UNARCHIVER()** [19/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 spinidx)
```

**5.1.3.163 is\_dummy\_pair()** [1/2]

```
bool GiNaC::is_dummy_pair (
 const idx & i1,
 const idx & i2)
```

Check whether two indices form a dummy pair.

References [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#).

Referenced by [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [GiNaC::indexed::has\\_dummy\\_index\\_for\(\)](#), [is\\_dummy\\_pair\(\)](#), [GiNaC::is\\_summation\\_idx::operator\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**5.1.3.164 is\_dummy\_pair()** [2/2]

```
bool GiNaC::is_dummy_pair (
 const ex & e1,
 const ex & e2)
```

Check whether two expressions form a dummy index pair.

References [is\\_dummy\\_pair\(\)](#).

**5.1.3.165 find\_free\_and\_dummy()** [1/2]

```
void GiNaC::find_free_and_dummy (
 exvector::const_iterator it,
 exvector::const_iterator itend,
 exvector & out_free,
 exvector & out_dummy)
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <i>it</i>        | Pointer to start of index vector           |
| <i>itend</i>     | Pointer to end of index vector             |
| <i>out_free</i>  | Vector of free indices (returned, sorted)  |
| <i>out_dummy</i> | Vector of dummy indices (returned, sorted) |

References [is\\_dummy\\_pair\(\)](#), [last](#), and [shaker\\_sort\(\)](#).

Referenced by [GiNaC::su3d::contract\\_with\(\)](#), [count\\_dummy\\_indices\(\)](#), [count\\_free\\_indices\(\)](#), [find\\_dummy\\_indices\(\)](#), [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::indexed::get\\_dummy\\_indices\(\)](#), [GiNaC::indexed::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), and [GiNaC::ncmul::get\\_free\\_indices\(\)](#).

**5.1.3.166 minimal\_dim()**

```
ex GiNaC::minimal_dim (
 const ex & dim1,
 const ex & dim2)
```

Return the minimum of two index dimensions.

If this is undecidable, throw an exception. Numeric dimensions are always considered "smaller" than symbolic dimensions.

References [GiNaC::ex::is\\_equal\(\)](#).

Referenced by [GiNaC::idx::minimal\\_dim\(\)](#).

**5.1.3.167 GINAC\_DECLARE\_UNARCHIVER() [18/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 idx)
```

**5.1.3.168 GINAC\_DECLARE\_UNARCHIVER() [19/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 varidx)
```

**5.1.3.169 GINAC\_DECLARE\_UNARCHIVER() [20/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 spinidx)
```

**5.1.3.170 find\_free\_and\_dummy() [2/2]**

```
void GiNaC::find_free_and_dummy (
 const exvector & v,
 exvector & out_free,
 exvector & out_dummy) [inline]
```

Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <i>v</i>         | Index vector                               |
| <i>out_free</i>  | Vector of free indices (returned, sorted)  |
| <i>out_dummy</i> | Vector of dummy indices (returned, sorted) |

References [find\\_free\\_and\\_dummy\(\)](#).

**5.1.3.171 find\_dummy\_indices()**

```
void GiNaC::find_dummy_indices (
 const exvector & v,
 exvector & out_dummy) [inline]
```

Given a vector of indices, find the dummy indices.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <i>v</i>         | Index vector                               |
| <i>out_dummy</i> | Vector of dummy indices (returned, sorted) |

References [find\\_free\\_and\\_dummy\(\)](#).

Referenced by [GiNaC::indexed::get\\_dummy\\_indices\(\)](#).

**5.1.3.172 count\_dummy\_indices()**

```
size_t GiNaC::count_dummy_indices (
 const exvector & v) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

**5.1.3.173 count\_free\_indices()**

```
size_t GiNaC::count_free_indices (
 const exvector & v) [inline]
```

Count the number of dummy index pairs in an index vector.

References [find\\_free\\_and\\_dummy\(\)](#).

**5.1.3.174 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [10/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 indexed ,
 exprseq ,
 print_func< print_context > &::do_print, print_func< print_latex > &::do_↔
 print_latex, print_func< print_tree > &::do_print_tree)
```

**5.1.3.175 GINAC\_BIND\_UNARCHIVER()** [20/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 indexed)
```

**5.1.3.176 indices\_consistent()**

```
static bool GiNaC::indices_consistent (
 const exvector & v1,
 const exvector & v2) [static]
```

Check whether two sorted index vectors are consistent (i.e. equal).

Referenced by [GiNaC::add::get\\_free\\_indices\(\)](#).

**5.1.3.177 number\_of\_type()**

```
template<class T >
size_t GiNaC::number_of_type (
 const exvector & v)
```

**5.1.3.178 rename\_dummy\_indices()**

```
template<class T >
static ex GiNaC::rename_dummy_indices (
 const ex & e,
 exvector & global_dummy_indices,
 exvector & local_dummy_indices) [static]
```

Rename dummy indices in an expression.

**Parameters**

|                             |                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <i>e</i>                    | Expression to work on                                                                                                            |
| <i>local_dummy_indices</i>  | The set of dummy indices that appear in the expression "e"                                                                       |
| <i>global_dummy_indices</i> | The set of dummy indices that have appeared before and which we would like to use in "e", too. This gets updated by the function |

References [GiNaC::ex::begin\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [op\(\)](#), [shaker\\_sort\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.179 find\_variant\_indices()**

```
static void GiNaC::find_variant_indices (
 const exvector & v,
 exvector & variant_indices) [static]
```

Given a set of indices, extract those of class varidx.

**5.1.3.180 reposition\_dummy\_indices()**

```
bool GiNaC::reposition_dummy_indices (
 ex & e,
 exvector & variant_dummy_indices,
 exvector & moved_indices)
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

**Parameters**

|                              |                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------|
| <i>e</i>                     | Object to work on                                                                   |
| <i>variant_dummy_indices</i> | The set of indices that might need repositioning (will be changed by this function) |
| <i>moved_indices</i>         | The set of indices that have been repositioned (will be changed by this function)   |

**Returns**

true if 'e' was changed

**5.1.3.181 product\_to\_exvector()**

```
static void GiNaC::product_to_exvector (
 const ex & e,
 exvector & v,
 bool & non_commutative) [static]
```

References [\\_ex2](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [get\\_all\\_dummy\\_indices\(\)](#).

**5.1.3.182 idx\_symmetrization()**

```
template<class T >
ex GiNaC::idx_symmetrization (
 const ex & r,
 const exvector & local_dummy_indices)
```

References [r](#), and [symmetrize\(\)](#).

**5.1.3.183 simplify\_indexed() [3/3]**

```
ex GiNaC::simplify_indexed (
 const ex & e,
 exvector & free_indices,
 exvector & dummy_indices,
 const scalar_products & sp)
```

Simplify indexed expression, return list of free indices.

**5.1.3.184 `simplify_indexed_product()`**

```
ex GiNaC::simplify_indexed_product (
 const ex & e,
 exvector & free_indices,
 exvector & dummy_indices,
 const scalar_products & sp)
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

**5.1.3.185 `hasindex()`**

```
bool GiNaC::hasindex (
 const ex & x,
 const ex & sym)
```

References [hasindex\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [hasindex\(\)](#).

**5.1.3.186 `get_all_dummy_indices_safely()`**

```
exvector GiNaC::get_all_dummy_indices_safely (
 const ex & e)
```

More reliable version of the form.

The former assumes that e is an expanded expression.

References [find\\_free\\_and\\_dummy\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [GiNaC::mul::expand\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

**5.1.3.187 `get_all_dummy_indices()`**

```
exvector GiNaC::get_all_dummy_indices (
 const ex & e)
```

Returns all dummy indices from the exvector.

Returns all dummy indices from the expression.

References [product\\_to\\_exvector\(\)](#).

Referenced by [expand\\_dummy\\_sum\(\)](#), and [GiNaC::power::expand\\_mul\(\)](#).

**5.1.3.188 rename\_dummy\_indices\_uniquely()** [1/4]

```
lst GiNaC::rename_dummy_indices_uniquely (
 const exvector & va,
 const exvector & vb)
```

Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.

References [GiNaC::container\\_storage< C >::reserve\(\)](#).

Referenced by [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [rename\\_dummy\\_indices\\_uniquely\(\)](#).

**5.1.3.189 rename\_dummy\_indices\_uniquely()** [2/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
 const exvector & va,
 const exvector & vb,
 const ex & b)
```

Same as above, where va and vb contain the indices of a and b and are sorted.

References [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.190 rename\_dummy\_indices\_uniquely()** [3/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
 const ex & a,
 const ex & b)
```

Returns b with all dummy indices, which are common with a, renamed.

References [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.191 rename\_dummy\_indices\_uniquely()** [4/4]

```
ex GiNaC::rename_dummy_indices_uniquely (
 exvector & va,
 const ex & b,
 bool modify_va)
```

Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b will be appended to va.

References [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.192 expand\_dummy\_sum()**

```
ex GiNaC::expand_dummy_sum (
 const ex & e,
 bool subs_idx = false)
```

This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.

Optionally all indices with a variance will be substituted by indices with the corresponding numeric values without variance.

## Parameters

|                 |                                                            |
|-----------------|------------------------------------------------------------|
| <i>e</i>        | the given expression                                       |
| <i>subs_idx</i> | indicates if variance of dummy indices should be neglected |

References [GiNaC::ex::expand\(\)](#), [expand\\_dummy\\_sum\(\)](#), [get\\_all\\_dummy\\_indices\(\)](#), [idx](#), [GiNaC::ex::map\(\)](#), [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [expand\\_dummy\\_sum\(\)](#).

### 5.1.3.193 GINAC\_DECLARE\_UNARCHIVER() [21/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 indexed)
```

### 5.1.3.194 conjugate\_evalf()

```
static ex GiNaC::conjugate_evalf (
 const ex & arg) [static]
```

References [GiNaC::ex::conjugate\(\)](#).

### 5.1.3.195 conjugate\_eval()

```
static ex GiNaC::conjugate_eval (
 const ex & arg) [static]
```

References [GiNaC::ex::conjugate\(\)](#).

### 5.1.3.196 conjugate\_print\_latex()

```
static void GiNaC::conjugate_print_latex (
 const ex & arg,
 const print_context & c) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

### 5.1.3.197 conjugate\_conjugate()

```
static ex GiNaC::conjugate_conjugate (
 const ex & arg) [static]
```

### 5.1.3.198 conjugate\_expl\_derivative()

```
static ex GiNaC::conjugate_expl_derivative (
 const ex & arg,
 const symbol & s) [static]
```

References [conjugate\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

**5.1.3.199 conjugate\_real\_part()**

```
static ex GiNaC::conjugate_real_part (
 const ex & arg) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

**5.1.3.200 conjugate\_imag\_part()**

```
static ex GiNaC::conjugate_imag_part (
 const ex & arg) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

**5.1.3.201 func\_arg\_info()**

```
static bool GiNaC::func_arg_info (
 const ex & arg,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [conjugate\\_info\(\)](#).

**5.1.3.202 conjugate\_info()**

```
static bool GiNaC::conjugate_info (
 const ex & arg,
 unsigned inf) [static]
```

References [func\\_arg\\_info\(\)](#).

**5.1.3.203 REGISTER\_FUNCTION() [1/36]**

```
GiNaC::REGISTER_FUNCTION (
 conjugate_function ,
 eval_func(conjugate_eval). evalf_func(conjugate_evalf). expl_derivative_←
func(conjugate_expl_derivative). info_func(conjugate_info). print_func< print_latex >(conjugate_print_latex
conjugate_func(conjugate_conjugate). real_part_func(conjugate_real_part). imag_part_func(conjugate_imag_part
set_name("conjugate","conjugate"))
```

**5.1.3.204 real\_part\_evalf()**

```
static ex GiNaC::real_part_evalf (
 const ex & arg) [static]
```

**5.1.3.205 real\_part\_eval()**

```
static ex GiNaC::real_part_eval (
 const ex & arg) [static]
```

References [GiNaC::ex::real\\_part\(\)](#).

**5.1.3.206 real\_part\_print\_latex()**

```
static void GiNaC::real_part_print_latex (
 const ex & arg,
 const print_context & c) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.207 real\_part\_conjugate()**

```
static ex GiNaC::real_part_conjugate (
 const ex & arg) [static]
```

**5.1.3.208 real\_part\_real\_part()**

```
static ex GiNaC::real_part_real_part (
 const ex & arg) [static]
```

**5.1.3.209 real\_part\_imag\_part()**

```
static ex GiNaC::real_part_imag_part (
 const ex & arg) [static]
```

**5.1.3.210 real\_part\_expl\_derivative()**

```
static ex GiNaC::real_part_expl_derivative (
 const ex & arg,
 const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::symbol::info\(\)](#), [GiNaC::info\\_flags::real](#), and [real\\_part\(\)](#).

**5.1.3.211 REGISTER\_FUNCTION() [2/36]**

```
GiNaC::REGISTER_FUNCTION (
 real_part_function ,
 eval_func(real_part_eval). evalf_func(real_part_evalf). expl_derivative_↔
func(real_part_expl_derivative). print_func< print_latex >(real_part_print_latex). conjugate↔
_func(real_part_conjugate). real_part_func(real_part_real_part). imag_part_func(real_part_imag_part).
set_name("real_part","real_part"))
```

**5.1.3.212 `imag_part_evalf()`**

```
static ex GiNaC::imag_part_evalf (
 const ex & arg) [static]
```

**5.1.3.213 `imag_part_eval()`**

```
static ex GiNaC::imag_part_eval (
 const ex & arg) [static]
```

References [GiNaC::ex::imag\\_part\(\)](#).

**5.1.3.214 `imag_part_print_latex()`**

```
static void GiNaC::imag_part_print_latex (
 const ex & arg,
 const print_context & c) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.215 `imag_part_conjugate()`**

```
static ex GiNaC::imag_part_conjugate (
 const ex & arg) [static]
```

**5.1.3.216 `imag_part_real_part()`**

```
static ex GiNaC::imag_part_real_part (
 const ex & arg) [static]
```

**5.1.3.217 `imag_part_imag_part()`**

```
static ex GiNaC::imag_part_imag_part (
 const ex & arg) [static]
```

**5.1.3.218 `imag_part_expl_derivative()`**

```
static ex GiNaC::imag_part_expl_derivative (
 const ex & arg,
 const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\\_part\(\)](#), [GiNaC::symbol::info\(\)](#), and [GiNaC::info\\_flags::real](#).

### 5.1.3.219 REGISTER\_FUNCTION() [3/36]

```
GiNaC::REGISTER_FUNCTION (
 imag_part_function ,
 eval_func(imag_part_eval). evalf_func(imag_part_evalf). expl_derivative_↵
func(imag_part_expl_derivative). print_func< print_latex >(imag_part_print_latex). conjugate_↵
_func(imag_part_conjugate). real_part_func(imag_part_real_part). imag_part_func(imag_part_imag_part).
set_name("imag_part","imag_part"))
```

### 5.1.3.220 abs\_evalf()

```
static ex GiNaC::abs_evalf (
 const ex & arg) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

### 5.1.3.221 abs\_eval()

```
static ex GiNaC::abs_eval (
 const ex & arg) [static]
```

References [abs\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [step\(\)](#).

### 5.1.3.222 abs\_expand()

```
static ex GiNaC::abs_expand (
 const ex & arg,
 unsigned options) [static]
```

References [abs\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), and [options](#).

### 5.1.3.223 abs\_expl\_derivative()

```
static ex GiNaC::abs_expl_derivative (
 const ex & arg,
 const symbol & s) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), and [GiNaC::ex::diff\(\)](#).

### 5.1.3.224 abs\_print\_latex()

```
static void GiNaC::abs_print_latex (
 const ex & arg,
 const print_context & c) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.225 `abs_print_csrc_float()`

```
static void GiNaC::abs_print_csrc_float (
 const ex & arg,
 const print_context & c) [static]
```

References [c](#), and [GiNaC::ex::print\(\)](#).

#### 5.1.3.226 `abs_conjugate()`

```
static ex GiNaC::abs_conjugate (
 const ex & arg) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

#### 5.1.3.227 `abs_real_part()`

```
static ex GiNaC::abs_real_part (
 const ex & arg) [static]
```

References [abs\(\)](#), and [GiNaC::basic::hold\(\)](#).

#### 5.1.3.228 `abs_imag_part()`

```
static ex GiNaC::abs_imag_part (
 const ex & arg) [static]
```

#### 5.1.3.229 `abs_power()`

```
static ex GiNaC::abs_power (
 const ex & arg,
 const ex & exp) [static]
```

References [abs\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::info\\_flags::even](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_even\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::real](#).

#### 5.1.3.230 `abs_info()`

```
bool GiNaC::abs_info (
 const ex & arg,
 unsigned inf)
```

References [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), and [GiNaC::info\\_flags::real](#).

**5.1.3.231 REGISTER\_FUNCTION() [4/36]**

```
GiNaC::REGISTER_FUNCTION (
 abs ,
 eval_func(abs_eval). evalf_func(abs_evalf). expand_func(abs_expand). expl_↵
 derivative_func(abs_expl_derivative). info_func(abs_info). print_func< print_latex >(abs_print_latex).
 print_func< print_csrc_float >(abs_print_csrc_float). print_func< print_csrc_double >(abs_print_csrc_float)
 conjugate_func(abs_conjugate). real_part_func(abs_real_part). imag_part_func(abs_imag_part).
 power_func(abs_power))
```

**5.1.3.232 step\_evalf()**

```
static ex GiNaC::step_evalf (
 const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

**5.1.3.233 step\_eval()**

```
static ex GiNaC::step_eval (
 const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), [l](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::numeric::real\(\)](#), and [step\(\)](#).

**5.1.3.234 step\_series()**

```
static ex GiNaC::step_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [step\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.235 step\_conjugate()**

```
static ex GiNaC::step_conjugate (
 const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

**5.1.3.236 step\_real\_part()**

```
static ex GiNaC::step_real_part (
 const ex & arg) [static]
```

References [GiNaC::basic::hold\(\)](#), and [step\(\)](#).

**5.1.3.237 step\_imag\_part()**

```
static ex GiNaC::step_imag_part (
 const ex & arg) [static]
```

**5.1.3.238 REGISTER\_FUNCTION() [5/36]**

```
GiNaC::REGISTER_FUNCTION (
 step ,
 eval_func(step_eval). evalf_func(step_evalf). series_func(step_series). conjugate←
_func(step_conjugate). real_part_func(step_real_part). imag_part_func(step_imag_part))
```

**5.1.3.239 csgn\_evalf()**

```
static ex GiNaC::csgn_evalf (
 const ex & arg) [static]
```

References [csgn\(\)](#).

**5.1.3.240 csgn\_eval()**

```
static ex GiNaC::csgn_eval (
 const ex & arg) [static]
```

References [csgn\(\)](#), [I](#), [GiNaC::numeric::imag\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::numeric::real\(\)](#).

**5.1.3.241 csgn\_series()**

```
static ex GiNaC::csgn_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.242 csgn\_conjugate()**

```
static ex GiNaC::csgn_conjugate (
 const ex & arg) [static]
```

References [csgn\(\)](#).

**5.1.3.243 csgn\_real\_part()**

```
static ex GiNaC::csgn_real_part (
 const ex & arg) [static]
```

References [csgn\(\)](#).

**5.1.3.244 csgn\_imag\_part()**

```
static ex GiNaC::csgn_imag_part (
 const ex & arg) [static]
```

**5.1.3.245 csgn\_power()**

```
static ex GiNaC::csgn_power (
 const ex & arg,
 const ex & exp) [static]
```

References [\\_ex2](#), [csgn\(\)](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::info\(\)](#), [is\\_odd\(\)](#), and [GiNaC::info\\_flags::positive](#).

**5.1.3.246 REGISTER\_FUNCTION() [6/36]**

```
GiNaC::REGISTER_FUNCTION (
 csgn ,
 eval_func(csgn_eval). evalf_func(csgn_evalf). series_func(csgn_series). conjugate↔
_func(csgn_conjugate). real_part_func(csgn_real_part). imag_part_func(csgn_imag_part). power↔
_func(csgn_power))
```

**5.1.3.247 eta\_evalf()**

```
static ex GiNaC::eta_evalf (
 const ex & x,
 const ex & y) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [evalf\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.248 eta\_eval()**

```
static ex GiNaC::eta_eval (
 const ex & x,
 const ex & y) [static]
```

References [\\_ex0](#), [csgn\(\)](#), [I](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.249 eta\_series()**

```
static ex GiNaC::eta_series (
 const ex & x,
 const ex & y,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.250 eta\_conjugate()**

```
static ex GiNaC::eta_conjugate (
 const ex & x,
 const ex & y) [static]
```

References [x](#).

**5.1.3.251 eta\_real\_part()**

```
static ex GiNaC::eta_real_part (
 const ex & x,
 const ex & y) [static]
```

**5.1.3.252 eta\_imag\_part()**

```
static ex GiNaC::eta_imag_part (
 const ex & x,
 const ex & y) [static]
```

References [GiNaC::basic::hold\(\)](#), [I](#), and [x](#).

**5.1.3.253 REGISTER\_FUNCTION() [7/36]**

```
GiNaC::REGISTER_FUNCTION (
 eta ,
 eval_func(eta_eval). evalf_func(eta_evalf). series_func(eta_series). latex↵
_name("\\eta"). set_symmetry(sy_symm(0, 1)). conjugate_func(eta_conjugate). real_part_↵
func(eta_real_part). imag_part_func(eta_imag_part))
```

**5.1.3.254 Li2\_evalf()**

```
static ex GiNaC::Li2_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [Li2\(\)](#), and [x](#).

**5.1.3.255 Li2\_eval()**

```
static ex GiNaC::Li2_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex12](#), [\\_ex1\\_2](#), [\\_ex2](#), [\\_ex6](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_48](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [l](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.256 Li2\_deriv()**

```
static ex GiNaC::Li2_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [log\(\)](#), and [x](#).

**5.1.3.257 Li2\_series()** [1/2]

```
static ex GiNaC::Li2_series (
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_num2\\_p](#), [l](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::relational::lhs\(\)](#), [Li2\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [op\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::series\\_options::suppress\\_branchcut](#), [x](#), and [zeta\(\)](#).

Referenced by [Li2\\_projection\(\)](#).

**5.1.3.258 Li2\_conjugate()**

```
static ex GiNaC::Li2_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\(\)](#), [GiNaC::info\\_flags::negative](#), and [x](#).

**5.1.3.259 REGISTER\_FUNCTION()** [8/36]

```
GiNaC::REGISTER_FUNCTION (
 Li2 ,
 eval_func(Li2_eval). evalf_func(Li2_evalf). derivative_func(Li2_deriv). series↔
_func(Li2_series). conjugate_func(Li2_conjugate). latex_name("\\mathrm{Li}_2"))
```

**5.1.3.260 Li3\_eval()**

```
static ex GiNaC::Li3_eval (
 const ex & x) [static]
```

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.261 REGISTER\_FUNCTION() [9/36]**

```
GiNaC::REGISTER_FUNCTION (
 Li3 ,
 eval_func(Li3_eval). latex_name("\\mathrm{Li}_3"))
```

**5.1.3.262 zetaderiv\_eval()**

```
static ex GiNaC::zetaderiv_eval (
 const ex & n,
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [zeta\(\)](#).

**5.1.3.263 zetaderiv\_deriv()**

```
static ex GiNaC::zetaderiv_deriv (
 const ex & n,
 const ex & x,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [n](#), and [x](#).

**5.1.3.264 REGISTER\_FUNCTION() [10/36]**

```
GiNaC::REGISTER_FUNCTION (
 zetaderiv ,
 eval_func(zetaderiv_eval). derivative_func(zetaderiv_deriv). latex_name("\\zeta^\\prime")
)
```

**5.1.3.265 factorial\_evalf()**

```
static ex GiNaC::factorial_evalf (
 const ex & x) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.266 factorial\_eval()**

```
static ex GiNaC::factorial_eval (
 const ex & x) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.267 factorial\_print\_dflt\_latex()**

```
static void GiNaC::factorial_print_dflt_latex (
 const ex & x,
 const print_context & c) [static]
```

References [c](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.268 factorial\_conjugate()**

```
static ex GiNaC::factorial_conjugate (
 const ex & x) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.269 factorial\_real\_part()**

```
static ex GiNaC::factorial_real_part (
 const ex & x) [static]
```

References [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.270 factorial\_imag\_part()**

```
static ex GiNaC::factorial_imag_part (
 const ex & x) [static]
```

**5.1.3.271 REGISTER\_FUNCTION() [11/36]**

```
GiNaC::REGISTER_FUNCTION (
 factorial ,
 eval_func(factorial_eval). evalf_func(factorial_evalf). print_func< print_dflt
>(factorial_print_dflt_latex). print_func< print_latex >(factorial_print_dflt_latex). conjugate←
_func(factorial_conjugate). real_part_func(factorial_real_part). imag_part_func(factorial_imag_part)
)
```

**5.1.3.272 binomial\_evalf()**

```
static ex GiNaC::binomial_evalf (
 const ex & x,
 const ex & y) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.273 binomial\_sym()**

```
static ex GiNaC::binomial_sym (
 const ex & x,
 const numeric & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [binomial\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_nonneg\\_integer\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [x](#).

Referenced by [binomial\\_eval\(\)](#).

**5.1.3.274 binomial\_eval()**

```
static ex GiNaC::binomial_eval (
 const ex & x,
 const ex & y) [static]
```

References [binomial\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::basic::hold\(\)](#), [is\\_integer\(\)](#), and [x](#).

**5.1.3.275 binomial\_conjugate()**

```
static ex GiNaC::binomial_conjugate (
 const ex & x,
 const ex & y) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.276 binomial\_real\_part()**

```
static ex GiNaC::binomial_real_part (
 const ex & x,
 const ex & y) [static]
```

References [binomial\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.277 binomial\_imag\_part()**

```
static ex GiNaC::binomial_imag_part (
 const ex & x,
 const ex & y) [static]
```

**5.1.3.278 REGISTER\_FUNCTION() [12/36]**

```
GiNaC::REGISTER_FUNCTION (
 binomial ,
 eval_func(binomial_eval) . evalf_func(binomial_evalf) . conjugate_func(binomial_conjugate) .
 real_part_func(binomial_real_part) . imag_part_func(binomial_imag_part))
```

**5.1.3.279 Order\_eval()**

```
static ex GiNaC::Order_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), [GiNaC::ex::op\(\)](#), and [x](#).

**5.1.3.280 Order\_series()**

```
static ex GiNaC::Order_series (
 const ex & x,
 const relational & r,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::ldegree\(\)](#), [order](#), [r](#), and [x](#).

**5.1.3.281 Order\_conjugate()**

```
static ex GiNaC::Order_conjugate (
 const ex & x) [static]
```

References [x](#).

**5.1.3.282 Order\_real\_part()**

```
static ex GiNaC::Order_real_part (
 const ex & x) [static]
```

References [x](#).

**5.1.3.283 Order\_imag\_part()**

```
static ex GiNaC::Order_imag_part (
 const ex & x) [static]
```

References [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.284 Order\_power()**

```
static ex GiNaC::Order_power (
 const ex & x,
 const ex & e) [static]
```

References [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), and [x](#).

**5.1.3.285 Order\_expl\_derivative()**

```
static ex GiNaC::Order_expl_derivative (
 const ex & arg,
 const symbol & s) [static]
```

References [GiNaC::ex::diff\(\)](#).

**5.1.3.286 REGISTER\_FUNCTION()** [13/36]

```
GiNaC::REGISTER_FUNCTION (
 Order ,
 eval_func(Order_eval). series_func(Order_series). latex_name("\\mathcal{O}").
 expl_derivative_func(Order_expl_derivative). conjugate_func(Order_conjugate). real_part_func(Order_real_part). imag_part_func(Order_imag_part). power_func(Order_power))
```

**5.1.3.287 Isolve()**

```
ex GiNaC::lsolve (
 const ex & eqns,
 const ex & symbols,
 unsigned options = solve_algo::automatic)
```

Factorial function.

Binomial function. Order term function (for truncated power series).

References [GiNaC::container< C >::append\(\)](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::matrix::cols\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::info\\_flags::exprseq](#), [GINAC\\_ASSERT](#), [GiNaC::symbolset::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::list](#), [Isolve\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [r](#), [GiNaC::info\\_flags::relation\\_equal](#), [rhs\(\)](#), [GiNaC::matrix::rows\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::info\\_flags::symbol](#), and [syms](#).

Referenced by [Isolve\(\)](#).

**5.1.3.288 fsolve()**

```
const numeric GiNaC::fsolve (
 const ex & f,
 const symbol & x,
 const numeric & x1,
 const numeric & x2)
```

Find a real root of real-valued function f(x) numerically within a given interval.

The function must change sign across interval. Uses Newton- Raphson method combined with bisection in order to guarantee convergence.

**Parameters**

|           |                      |
|-----------|----------------------|
| <i>f</i>  | Function f(x)        |
| <i>x</i>  | Symbol f(x)          |
| <i>x1</i> | lower interval limit |
| <i>x2</i> | upper interval limit |

## Exceptions

|                            |                           |
|----------------------------|---------------------------|
| <code>runtime_error</code> | (if interval is invalid). |
|----------------------------|---------------------------|

References [GiNaC::ex::diff\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [is\\_real\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::ex::lhs\(\)](#), [normal\(\)](#), [GiNaC::ex::rhs\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.289 zeta()** [1/3]

```
template<typename T1 >
function GiNaC::zeta (
 const T1 & p1) [inline]
```

References [GiNaC::zeta1\\_SERIAL::serial](#).

Referenced by [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [psi2\\_eval\(\)](#), [S\\_eval\(\)](#), [zeta1\\_eval\(\)](#), [zeta1\\_evalf\(\)](#), [zeta2\\_eval\(\)](#), [zeta2\\_evalf\(\)](#), and [zetaderiv\\_eval\(\)](#).

**5.1.3.290 zeta()** [2/3]

```
template<typename T1 , typename T2 >
function GiNaC::zeta (
 const T1 & p1,
 const T2 & p2) [inline]
```

References [GiNaC::zeta2\\_SERIAL::serial](#).

**5.1.3.291 is\_the\_function< zeta\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< zeta_SERIAL > (
 const ex & x) [inline]
```

References [x](#).

**5.1.3.292 G()** [1/2]

```
template<typename T1 , typename T2 >
function GiNaC::G (
 const T1 & x,
 const T2 & y) [inline]
```

References [GiNaC::G2\\_SERIAL::serial](#), and [x](#).

Referenced by [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), and [G3\\_evalf\(\)](#).

**5.1.3.293 G()** [2/2]

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::G (
 const T1 & x,
 const T2 & s,
 const T3 & y) [inline]
```

References [GiNaC::G3\\_SERIAL::serial](#), and [x](#).

**5.1.3.294 is\_the\_function< G\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< G_SERIAL > (
 const ex & x) [inline]
```

References [x](#).

**5.1.3.295 psi()** [1/4]

```
template<typename T1 >
function GiNaC::psi (
 const T1 & p1) [inline]
```

References [GiNaC::psi1\\_SERIAL::serial](#).

Referenced by [beta\\_deriv\(\)](#), [lgamma\\_deriv\(\)](#), [psi1\\_deriv\(\)](#), [psi1\\_eval\(\)](#), [psi1\\_evalf\(\)](#), [psi1\\_series\(\)](#), [psi2\\_deriv\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_evalf\(\)](#), [psi2\\_series\(\)](#), [sr\\_gcd\(\)](#), and [tgamma\\_deriv\(\)](#).

**5.1.3.296 psi()** [2/4]

```
template<typename T1 , typename T2 >
function GiNaC::psi (
 const T1 & p1,
 const T2 & p2) [inline]
```

References [GiNaC::psi2\\_SERIAL::serial](#).

**5.1.3.297 is\_the\_function< psi\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< psi_SERIAL > (
 const ex & x) [inline]
```

References [x](#).

**5.1.3.298 iterated\_integral() [1/2]**

```
template<typename T1 , typename T2 >
function GiNaC::iterated_integral (
 const T1 & kernel_lst,
 const T2 & lambda) [inline]
```

References [GiNaC::iterated\\_integral2\\_SERIAL::serial](#).

Referenced by [iterated\\_integral2\\_eval\(\)](#), [iterated\\_integral3\\_eval\(\)](#), and [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.299 iterated\_integral() [2/2]**

```
template<typename T1 , typename T2 , typename T3 >
function GiNaC::iterated_integral (
 const T1 & kernel_lst,
 const T2 & lambda,
 const T3 & N_trunc) [inline]
```

References [GiNaC::iterated\\_integral3\\_SERIAL::serial](#).

**5.1.3.300 is\_the\_function< iterated\_integral\_SERIAL >()**

```
template<>
bool GiNaC::is_the_function< iterated_integral_SERIAL > (
 const ex & x) [inline]
```

References [x](#).

**5.1.3.301 is\_order\_function()**

```
bool GiNaC::is_order_function (
 const ex & e) [inline]
```

Check whether a function is the Order (O(n)) function.

References [is\\_ex\\_the\\_function](#).

Referenced by [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::op\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), and [GiNaC::integral::series\(\)](#).

**5.1.3.302 convert\_H\_to\_Li()**

```
ex GiNaC::convert_H_to_Li (
 const ex & parameterlst,
 const ex & arg)
```

Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding [GiNaC](#) functions.

References [m](#), and [x](#).

**5.1.3.303 EllipticK\_evalf()**

```
static ex GiNaC::EllipticK_evalf (
 const ex & k) [static]
```

References [GiNaC::ex::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

**5.1.3.304 EllipticK\_eval()**

```
static ex GiNaC::EllipticK_eval (
 const ex & k) [static]
```

References [\\_ex0](#), [GiNaC::info\\_flags::crational](#), [GiNaC::constant::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

**5.1.3.305 EllipticK\_deriv()**

```
static ex GiNaC::EllipticK_deriv (
 const ex & k,
 unsigned deriv_param) [static]
```

References [k](#).

**5.1.3.306 EllipticK\_series()**

```
static ex GiNaC::EllipticK_series (
 const ex & k,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::symbol::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.307 EllipticK\_print\_latex()**

```
static void GiNaC::EllipticK_print_latex (
 const ex & k,
 const print_context & c) [static]
```

References [c](#), and [k](#).

**5.1.3.308 REGISTER\_FUNCTION() [14/36]**

```
GiNaC::REGISTER_FUNCTION (
 EllipticK ,
 evalf_func(EllipticK_evalf). eval_func(EllipticK_eval). derivative_func(EllipticK_deriv).
 series_func(EllipticK_series). print_func< print_latex >(EllipticK_print_latex). do_not_↔
 evalf_params())
```

**5.1.3.309 EllipticE\_evalf()**

```
static ex GiNaC::EllipticE_evalf (
 const ex & k) [static]
```

References [GiNaC::ex::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [sqrt\(\)](#).

**5.1.3.310 EllipticE\_eval()**

```
static ex GiNaC::EllipticE_eval (
 const ex & k) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::info\\_flags::crational](#), [GiNaC::constant::evalf\(\)](#), [k](#), [GiNaC::info\\_flags::numeric](#), and [Pi](#).

**5.1.3.311 EllipticE\_deriv()**

```
static ex GiNaC::EllipticE_deriv (
 const ex & k,
 unsigned deriv_param) [static]
```

References [k](#).

**5.1.3.312 EllipticE\_series()**

```
static ex GiNaC::EllipticE_series (
 const ex & k,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [Pi](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::symbol::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.313 EllipticE\_print\_latex()**

```
static void GiNaC::EllipticE_print_latex (
 const ex & k,
 const print_context & c) [static]
```

References [c](#), and [k](#).

**5.1.3.314 REGISTER\_FUNCTION() [15/36]**

```
GiNaC::REGISTER_FUNCTION (
 EllipticE ,
 evalf_func(EllipticE_evalf). eval_func(EllipticE_eval). derivative_func(EllipticE_deriv).
 series_func(EllipticE_series). print_func< print_latex >(EllipticE_print_latex). do_not_↔
 evalf_params())
```

**5.1.3.315 iterated\_integral\_evalf\_impl()**

```
static ex GiNaC::iterated_integral_evalf_impl (
 const ex & kernel_lst,
 const ex & lambda,
 const ex & N_trunc) [static]
```

References [coeff\(\)](#), [Digits](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), [GiNaC::info\\_flags::list](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), and [one](#).

Referenced by [iterated\\_integral2\\_evalf\(\)](#), and [iterated\\_integral3\\_evalf\(\)](#).

**5.1.3.316 iterated\_integral2\_evalf()**

```
static ex GiNaC::iterated_integral2_evalf (
 const ex & kernel_lst,
 const ex & lambda) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.317 iterated\_integral3\_evalf()**

```
static ex GiNaC::iterated_integral3_evalf (
 const ex & kernel_lst,
 const ex & lambda,
 const ex & N_trunc) [static]
```

References [iterated\\_integral\\_evalf\\_impl\(\)](#).

**5.1.3.318 iterated\_integral2\_eval()**

```
static ex GiNaC::iterated_integral2_eval (
 const ex & kernel_lst,
 const ex & lambda) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**5.1.3.319 iterated\_integral3\_eval()**

```
static ex GiNaC::iterated_integral3_eval (
 const ex & kernel_lst,
 const ex & lambda,
 const ex & N_trunc) [static]
```

References [GiNaC::info\\_flags::crational](#), [GiNaC::function::evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [iterated\\_integral\(\)](#), and [GiNaC::info\\_flags::numeric](#).

**5.1.3.320 lgamma\_evalf()**

```
static ex GiNaC::lgamma_evalf (
 const ex & x) [static]
```

References [lgamma\(\)](#), and [x](#).

**5.1.3.321 lgamma\_eval()**

```
static ex GiNaC::lgamma_eval (
 const ex & x) [static]
```

Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.

Handles integer arguments as a special case.

**Exceptions**

|                                                    |                      |
|----------------------------------------------------|----------------------|
| <a href="#">GiNaC::pole_error("lgamma_eval()")</a> | logarithmic pole",0) |
|----------------------------------------------------|----------------------|

References [\\_ex\\_1](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [is\\_rational\(\)](#), [lgamma\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.322 lgamma\_deriv()**

```
static ex GiNaC::lgamma_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

**5.1.3.323 lgamma\_series()**

```
static ex GiNaC::lgamma_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [lgamma\(\)](#), [log\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::basic::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.324 lgamma\_conjugate()**

```
static ex GiNaC::lgamma_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lgamma\(\)](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.325 REGISTER\_FUNCTION()** [16/36]

```
GiNaC::REGISTER_FUNCTION (
 lgamma ,
 eval_func(lgamma_eval). evalf_func(lgamma_evalf). derivative_func(lgamma_deriv).
 series_func(lgamma_series). conjugate_func(lgamma_conjugate). latex_name("\\log \\Gamma"))
```

**5.1.3.326 tgamma\_evalf()**

```
static ex GiNaC::tgamma_evalf (
 const ex & x) [static]
```

References [tgamma\(\)](#), and [x](#).

**5.1.3.327 tgamma\_eval()**

```
static ex GiNaC::tgamma_eval (
 const ex & x) [static]
```

Evaluation of  $\text{tgamma}(x)$ , the true Gamma function.

Knows about integer arguments, half-integer arguments and that's it. Somebody ought to provide some good numerical evaluation some day...

**Exceptions**

|                                                        |
|--------------------------------------------------------|
| <code>pole_error("tgamma_eval() simple pole",0)</code> |
|--------------------------------------------------------|

References [\\_num1\\_2\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num\\_2\\_p](#), [abs\(\)](#), [GiNaC::numeric::div\(\)](#), [doublefactorial\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_even\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [is\\_rational\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [pow\(\)](#), [sqrt\(\)](#), [GiNaC::numeric::sub\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.328 tgamma\_deriv()**

```
static ex GiNaC::tgamma_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.329 tgamma\_series()**

```
static ex GiNaC::tgamma_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [tgamma\(\)](#).

**5.1.3.330 `tgamma_conjugate()`**

```
static ex GiNaC::tgamma_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.331 `REGISTER_FUNCTION()` [17/36]**

```
GiNaC::REGISTER_FUNCTION (
 tgamma ,
 eval_func(tgamma_eval) . evalf_func(tgamma_evalf) . derivative_func(tgamma_deriv) .
 series_func(tgamma_series) . conjugate_func(tgamma_conjugate) . latex_name("\\Gamma"))
```

**5.1.3.332 `beta_evalf()`**

```
static ex GiNaC::beta_evalf (
 const ex & x,
 const ex & y) [static]
```

References [exp\(\)](#), [lgamma\(\)](#), and [x](#).

**5.1.3.333 `beta_eval()`**

```
static ex GiNaC::beta_eval (
 const ex & x,
 const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_num\\_1\\_p](#), [evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [is\\_positive\(\)](#), [is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [is\\_real\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [tgamma\(\)](#), and [x](#).

**5.1.3.334 `beta_deriv()`**

```
static ex GiNaC::beta_deriv (
 const ex & x,
 const ex & y,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

**5.1.3.335 `beta_series()`**

```
static ex GiNaC::beta_series (
 const ex & arg1,
 const ex & arg2,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [GiNaC::ex::subs\(\)](#), and [tgamma\(\)](#).

**5.1.3.336 REGISTER\_FUNCTION()** [18/36]

```
GiNaC::REGISTER_FUNCTION (
 beta ,
 eval_func(beta_eval). evalf_func(beta_evalf). derivative_func(beta_deriv).
 series_func(beta_series). latex_name("\\mathrm{B}"). set_symmetry(sy_symm(0, 1)))
```

**5.1.3.337 psi1\_evalf()**

```
static ex GiNaC::psi1_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [psi\(\)](#), and [x](#).

**5.1.3.338 psi1\_eval()**

```
static ex GiNaC::psi1_eval (
 const ex & x) [static]
```

Evaluation of digamma-function  $\psi(x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1\\_2](#), [\\_ex2](#), [\\_num2\\_p](#), [\\_num\\_1\\_p](#), [Euler](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [psi\(\)](#), and [x](#).

**5.1.3.339 psi1\_deriv()**

```
static ex GiNaC::psi1_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [psi\(\)](#), and [x](#).

**5.1.3.340 psi1\_series()**

```
static ex GiNaC::psi1_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.341 psi2\_evalf()**

```
static ex GiNaC::psi2_evalf (
 const ex & n,
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [n](#), [psi\(\)](#), and [x](#).

**5.1.3.342 psi2\_eval()**

```
static ex GiNaC::psi2_eval (
 const ex & n,
 const ex & x) [static]
```

Evaluation of polygamma-function  $\psi(n,x)$ .

Somebody ought to provide some good numerical evaluation some day...

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_num1\\_2\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num\\_1\\_p](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [is\\_integer\(\)](#), [GiNaC::numeric::is\\_positive\(\)](#), [log\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [psi\(\)](#), [tgamma\(\)](#), [x](#), and [zeta\(\)](#).

**5.1.3.343 psi2\_deriv()**

```
static ex GiNaC::psi2_deriv (
 const ex & n,
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [GINAC\\_ASSERT](#), [n](#), [psi\(\)](#), and [x](#).

**5.1.3.344 psi2\_series()**

```
static ex GiNaC::psi2_series (
 const ex & n,
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [m](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [GiNaC::info\\_flags::positive](#), [psi\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.345 G2\_evalf()**

```
static ex GiNaC::G2_evalf (
 const ex & x_,
 const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [x](#).

**5.1.3.346 G2\_eval()**

```
static ex GiNaC::G2_eval (
 const ex & x_,
 const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [imag\(\)](#), [GiNaC::ex::info\(\)](#), [is\\_real\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), and [x](#).

**5.1.3.347 G3\_evalf()**

```
static ex GiNaC::G3_evalf (
 const ex & x_,
 const ex & s_,
 const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.348 G3\_eval()**

```
static ex GiNaC::G3_eval (
 const ex & x_,
 const ex & s_,
 const ex & y) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ex::begin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::end\(\)](#), [factorial\(\)](#), [G\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.349 Li\_evalf()**

```
static ex GiNaC::Li_evalf (
 const ex & m_,
 const ex & x_) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.350 Li\_eval()**

```
static ex GiNaC::Li_eval (
 const ex & m_,
 const ex & x_) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_48](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::ex::begin\(\)](#), [Catalan](#), [GiNaC::info\\_flags::crational](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [log\(\)](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), [x](#), and [zeta\(\)](#).

**5.1.3.351 Li\_series()**

```
static ex GiNaC::Li_series (
 const ex & m,
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [order](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.352 Li\_deriv()**

```
static ex GiNaC::Li_deriv (
 const ex & m_,
 const ex & x_,
 unsigned deriv_param) [static]
```

References [\\_ex0](#), [GINAC\\_ASSERT](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

**5.1.3.353 Li\_print\_latex()**

```
static void GiNaC::Li_print_latex (
 const ex & m_,
 const ex & x_,
 const print_context & c) [static]
```

References [GiNaC::ex::begin\(\)](#), [c](#), [GiNaC::ex::end\(\)](#), [m](#), and [x](#).

**5.1.3.354 REGISTER\_FUNCTION() [19/36]**

```
GiNaC::REGISTER_FUNCTION (
 Li ,
 evalf_func(Li_evalf). eval_func(Li_eval). series_func(Li_series). derivative_↔
func(Li_deriv). print_func< print_latex >(Li_print_latex). do_not_evalf_params())
```

**5.1.3.355 S\_evalf()**

```
static ex GiNaC::S_evalf (
 const ex & n,
 const ex & p,
 const ex & x) [static]
```

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [n](#), [GiNaC::info\\_flags::posint](#), and [x](#).

**5.1.3.356 S\_eval()**

```
static ex GiNaC::S_eval (
 const ex & n,
 const ex & p,
 const ex & x) [static]
```

References [\\_ex0](#), [GiNaC::container< C >::append\(\)](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [to\\_int\(\)](#), [x](#), and [zeta\(\)](#).

**5.1.3.357 S\_series()**

```
static ex GiNaC::S_series (
 const ex & n,
 const ex & p,
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex1](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::numeric](#), [options](#), [order](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.358 S\_deriv()**

```
static ex GiNaC::S_deriv (
 const ex & n,
 const ex & p,
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex0](#), [GINAC\\_ASSERT](#), [n](#), and [x](#).

**5.1.3.359 S\_print\_latex()**

```
static void GiNaC::S_print_latex (
 const ex & n,
 const ex & p,
 const ex & x,
 const print_context & c) [static]
```

References [c](#), [n](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.360 REGISTER\_FUNCTION() [20/36]**

```
GiNaC::REGISTER_FUNCTION (
 S ,
 evalf_func(S_evalf). eval_func(S_eval). series_func(S_series). derivative_↵
func(S_deriv). print_func< print_latex >(S_print_latex). do_not_evalf_params())
```

**5.1.3.361 H\_evalf()**

```
static ex GiNaC::H_evalf (
 const ex & x1,
 const ex & x2) [static]
```

References [GiNaC::container< C >::append\(\)](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::evalf\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::let\\_op\(\)](#), [m](#), [GiNaC::ex::nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [Pi](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.362 H\_eval()**

```
static ex GiNaC::H_eval (
 const ex & m_,
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::ex::evalf\(\)](#), [factorial\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [log\(\)](#), [m](#), [n](#), [GiNaC::basic::nops\(\)](#), [GiNaC::info\\_flags::numeric](#), [pow\(\)](#), [step\(\)](#), and [x](#).

**5.1.3.363 H\_series()**

```
static ex GiNaC::H_series (
 const ex & m,
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [m](#), and [x](#).

**5.1.3.364 H\_deriv()**

```
static ex GiNaC::H_deriv (
 const ex & m_,
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::begin\(\)](#), [GINAC\\_ASSERT](#), [m](#), and [x](#).

**5.1.3.365 H\_print\_latex()**

```
static void GiNaC::H_print_latex (
 const ex & m_,
 const ex & x,
 const print_context & c) [static]
```

References [GiNaC::ex::begin\(\)](#), [c](#), [m](#), [GiNaC::ex::print\(\)](#), and [x](#).

**5.1.3.366 REGISTER\_FUNCTION() [21/36]**

```
GiNaC::REGISTER_FUNCTION (
 H ,
 evalf_func(H_evalf). eval_func(H_eval). series_func(H_series). derivative_↵
func(H_deriv). print_func< print_latex >(H_print_latex). do_not_evalf_params())
```

**5.1.3.367 zeta1\_evalf()**

```
static ex GiNaC::zeta1_evalf (
 const ex & x) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [Digits](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), [r](#), [x](#), and [zeta\(\)](#).

Referenced by [zeta1\\_eval\(\)](#).

**5.1.3.368 zeta1\_eval()**

```
static ex GiNaC::zeta1_eval (
 const ex & m) [static]
```

References [\\_ex0](#), [\\_ex1\\_2](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [abs\(\)](#), [bernoulli\(\)](#), [GiNaC::info\\_flags::crational](#), [factorial\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::info\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [m](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [Pi](#), [GiNaC::info\\_flags::posint](#), [pow\(\)](#), [zeta\(\)](#), and [zeta1\\_evalf\(\)](#).

**5.1.3.369 zeta1\_deriv()**

```
static ex GiNaC::zeta1_deriv (
 const ex & m,
 unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), and [m](#).

**5.1.3.370 zeta1\_print\_latex()**

```
static void GiNaC::zeta1_print_latex (
 const ex & m_,
 const print_context & c) [static]
```

References [c](#), [m](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.371 zeta2\_evalf()**

```
static ex GiNaC::zeta2_evalf (
 const ex & x,
 const ex & s) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [evalf\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::info\\_flags::posint](#), [x](#), and [zeta\(\)](#).

**5.1.3.372 zeta2\_eval()**

```
static ex GiNaC::zeta2_eval (
 const ex & m,
 const ex & s_) [static]
```

References [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::info\\_flags::positive](#), and [zeta\(\)](#).

**5.1.3.373 zeta2\_deriv()**

```
static ex GiNaC::zeta2_deriv (
 const ex & m,
 const ex & s,
 unsigned deriv_param) [static]
```

References [\\_ex0](#), [\\_ex1](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::op\(\)](#), and [GiNaC::info\\_flags::positive](#).

**5.1.3.374 zeta2\_print\_latex()**

```
static void GiNaC::zeta2_print_latex (
 const ex & m_,
 const ex & s_,
 const print_context & c) [static]
```

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::ex::begin\(\)](#), [c](#), and [m](#).

**5.1.3.375 exp\_evalf()**

```
static ex GiNaC::exp_evalf (
 const ex & x) [static]
```

References [exp\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.376 exp\_eval()**

```
static ex GiNaC::exp_eval (
 const ex & x) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num2\\_p](#), [\\_num3\\_p](#), [\\_num4\\_p](#), [GiNaC::info\\_flags::crational](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [mod\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), and [x](#).

**5.1.3.377 exp\_expand()**

```
static ex GiNaC::exp_expand (
 const ex & arg,
 unsigned options) [static]
```

References [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [exp\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::nops\(\)](#), and [options](#).

#### 5.1.3.378 `exp_deriv()`

```
static ex GiNaC::exp_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [exp\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

#### 5.1.3.379 `exp_real_part()`

```
static ex GiNaC::exp_real_part (
 const ex & x) [static]
```

References [cos\(\)](#), [exp\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

#### 5.1.3.380 `exp_imag_part()`

```
static ex GiNaC::exp_imag_part (
 const ex & x) [static]
```

References [exp\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

#### 5.1.3.381 `exp_conjugate()`

```
static ex GiNaC::exp_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [exp\(\)](#), and [x](#).

#### 5.1.3.382 `exp_power()`

```
static ex GiNaC::exp_power (
 const ex & x,
 const ex & a) [static]
```

References [\\_ex\\_1](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::real](#), and [x](#).

#### 5.1.3.383 `exp_info()`

```
static bool GiNaC::exp_info (
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.384 REGISTER\_FUNCTION() [22/36]**

```
GiNaC::REGISTER_FUNCTION (
 exp ,
 eval_func(exp_eval). evalf_func(exp_evalf). info_func(exp_info). expand_←
func(exp_expand). derivative_func(exp_deriv). real_part_func(exp_real_part). imag_part_←
func(exp_imag_part). conjugate_func(exp_conjugate). power_func(exp_power). latex_name("\\exp")
)
```

**5.1.3.385 log\_evalf()**

```
static ex GiNaC::log_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [log\(\)](#), and [x](#).

**5.1.3.386 log\_eval()**

```
static ex GiNaC::log_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1\\_2](#), [GiNaC::info\\_flags::crational](#), [exp\(\)](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.387 log\_deriv()**

```
static ex GiNaC::log_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.388 log\_series()**

```
static ex GiNaC::log_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::pseries::coeff\(\)](#), [coeff\(\)](#), [csgn\(\)](#), [GiNaC::ex::diff\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::pseries::is\\_terminating\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::pseries::ldegree\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [n](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::pseries::nops\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::positive](#), [pow\(\)](#), [GiNaC::relational::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.389 log\_real\_part()**

```
static ex GiNaC::log_real_part (
 const ex & x) [static]
```

References [abs\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::nonnegative](#), and [x](#).

**5.1.3.390 log\_imag\_part()**

```
static ex GiNaC::log_imag_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.391 log\_expand()**

```
static ex GiNaC::log_expand (
 const ex & arg,
 unsigned options) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::end\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_func](#), [GiNaC::expand\\_options::expand\\_transcendental](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::ex::nops\(\)](#), [options](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.392 log\_conjugate()**

```
static ex GiNaC::log_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [log\(\)](#), [GiNaC::info\\_flags::positive](#), and [x](#).

**5.1.3.393 log\_info()**

```
static bool GiNaC::log_info (
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.394 REGISTER\_FUNCTION() [23/36]**

```
GiNaC::REGISTER_FUNCTION (
 log ,
 eval_func(log_eval). evalf_func(log_evalf). info_func(log_info). expand_↵
func(log_expand). derivative_func(log_deriv). series_func(log_series). real_part_func(log_real_part).
imag_part_func(log_imag_part). conjugate_func(log_conjugate). latex_name("\\ln")
```

**5.1.3.395 `sin_evalf()`**

```
static ex GiNaC::sin_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.396 `sin_eval()`**

```
static ex GiNaC::sin_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num15\\_p](#), [\\_num18\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [\\_num6\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sin\(\)](#), [sqrt\(\)](#), and [x](#).

**5.1.3.397 `sin_deriv()`**

```
static ex GiNaC::sin_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [cos\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.398 `sin_real_part()`**

```
static ex GiNaC::sin_real_part (
 const ex & x) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.399 `sin_imag_part()`**

```
static ex GiNaC::sin_imag_part (
 const ex & x) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.400 `sin_conjugate()`**

```
static ex GiNaC::sin_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.401 trig\_info()**

```
static bool GiNaC::trig_info (
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.402 REGISTER\_FUNCTION() [24/36]**

```
GiNaC::REGISTER_FUNCTION (
 sin ,
 eval_func(sin_eval). evalf_func(sin_evalf). info_func(trig_info). derivative_↵
func(sin_deriv). real_part_func(sin_real_part). imag_part_func(sin_imag_part). conjugate_↵
func(sin_conjugate). latex_name("\\sin"))
```

**5.1.3.403 cos\_evalf()**

```
static ex GiNaC::cos_evalf (
 const ex & x) [static]
```

References [cos\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.404 cos\_eval()**

```
static ex GiNaC::cos_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex1\\_4](#), [\\_ex2](#), [\\_ex3](#), [\\_ex5](#), [\\_ex6](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_3](#), [\\_ex\\_1\\_4](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num120\\_p](#), [\\_num12\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num24\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [cos\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sqrt\(\)](#), and [x](#).

**5.1.3.405 cos\_deriv()**

```
static ex GiNaC::cos_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [sin\(\)](#), and [x](#).

**5.1.3.406 cos\_real\_part()**

```
static ex GiNaC::cos_real_part (
 const ex & x) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.407 cos\_imag\_part()**

```
static ex GiNaC::cos_imag_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.408 cos\_conjugate()**

```
static ex GiNaC::cos_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [cos\(\)](#), and [x](#).

**5.1.3.409 REGISTER\_FUNCTION() [25/36]**

```
GiNaC::REGISTER_FUNCTION (
 cos ,
 eval_func(cos_eval). info_func(trig_info). evalf_func(cos_evalf). derivative_↔
func(cos_deriv). real_part_func(cos_real_part). imag_part_func(cos_imag_part). conjugate_↔
func(cos_conjugate). latex_name("\\cos"))
```

**5.1.3.410 tan\_evalf()**

```
static ex GiNaC::tan_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.411 tan\_eval()**

```
static ex GiNaC::tan_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_3](#), [\\_ex2](#), [\\_ex3](#), [\\_ex60](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [\\_num0\\_p](#), [\\_num10\\_p](#), [\\_num15\\_p](#), [\\_num20\\_p](#), [\\_num25\\_p](#), [\\_num30\\_p](#), [\\_num5\\_p](#), [\\_num60\\_p](#), [acos\(\)](#), [asin\(\)](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [mod\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [sqrt\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.412 tan\_deriv()**

```
static ex GiNaC::tan_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), [tan\(\)](#), and [x](#).

**5.1.3.413 tan\_real\_part()**

```
static ex GiNaC::tan_real_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.414 tan\_imag\_part()**

```
static ex GiNaC::tan_imag_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.415 tan\_series()**

```
static ex GiNaC::tan_series (
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [cos\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [options](#), [order](#), [Pi](#), [sin\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.416 tan\_conjugate()**

```
static ex GiNaC::tan_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tan\(\)](#), and [x](#).

**5.1.3.417 REGISTER\_FUNCTION() [26/36]**

```
GiNaC::REGISTER_FUNCTION (
 tan ,
 eval_func(tan_eval). evalf_func(tan_evalf). info_func(trig_info). derivative←
_func(tan_deriv). series_func(tan_series). real_part_func(tan_real_part). imag_part←
func(tan_imag_part). conjugate_func(tan_conjugate). latex_name("\\tan"))
```

**5.1.3.418 asin\_evalf()**

```
static ex GiNaC::asin_evalf (
 const ex & x) [static]
```

References [asin\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.419 asin\_eval()**

```
static ex GiNaC::asin_eval (
 const ex & x) [static]
```

References [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [asin\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.420 asin\_deriv()**

```
static ex GiNaC::asin_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.421 asin\_conjugate()**

```
static ex GiNaC::asin_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asin\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.422 asin\_info()**

```
static bool GiNaC::asin_info (
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), and [x](#).

**5.1.3.423 REGISTER\_FUNCTION() [27/36]**

```
GiNaC::REGISTER_FUNCTION (
 asin ,
 eval_func(asin_eval). evalf_func(asin_evalf). info_func(asin_info). derivative↔
 _func(asin_deriv). conjugate_func(asin_conjugate). latex_name("\\arcsin"))
```

**5.1.3.424 acos\_evalf()**

```
static ex GiNaC::acos_evalf (
 const ex & x) [static]
```

References [acos\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.425 acos\_eval()**

```
static ex GiNaC::acos_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex1\\_3](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acos\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.426 acos\_deriv()**

```
static ex GiNaC::acos_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.427 acos\_conjugate()**

```
static ex GiNaC::acos_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [acos\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.428 REGISTER\_FUNCTION() [28/36]**

```
GiNaC::REGISTER_FUNCTION (
 acos ,
 eval_func(acos_eval). evalf_func(acos_evalf). info_func(asin_info). derivative↔
 _func(acos_deriv). conjugate_func(acos_conjugate). latex_name("\\arccos")
```

**5.1.3.429 atan\_evalf()**

```
static ex GiNaC::atan_evalf (
 const ex & x) [static]
```

References [atan\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.430 atan\_eval()**

```
static ex GiNaC::atan_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_4](#), [\\_ex\\_1](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.431 atan\_deriv()**

```
static ex GiNaC::atan_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.432 atan\_series()**

```
static ex GiNaC::atan_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atan\(\)](#), [csgn\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.433 atan\_conjugate()**

```
static ex GiNaC::atan_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atan\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [x](#).

**5.1.3.434 atan\_info()**

```
static bool GiNaC::atan_info (
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.435 REGISTER\_FUNCTION() [29/36]**

```
GiNaC::REGISTER_FUNCTION (
 atan ,
 eval_func(atan_eval). evalf_func(atan_evalf). info_func(atan_info). derivative↔
_func(atan_deriv). series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\\arctan")
)
```

**5.1.3.436 atan2\_evalf()**

```
static ex GiNaC::atan2_evalf (
 const ex & y,
 const ex & x) [static]
```

References [atan\(\)](#), and [x](#).

**5.1.3.437 atan2\_eval()**

```
static ex GiNaC::atan2_eval (
 const ex & y,
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1\\_2](#), [\\_ex1\\_4](#), [\\_ex\\_1\\_2](#), [\\_ex\\_1\\_4](#), [atan\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [Pi](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.438 atan2\_deriv()**

```
static ex GiNaC::atan2_deriv (
 const ex & y,
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.439 atan2\_info()**

```
static bool GiNaC::atan2_info (
 const ex & y,
 const ex & x,
 unsigned inf) [static]
```

References [GiNaC::info\\_flags::expanded](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::real](#), and [x](#).

**5.1.3.440 REGISTER\_FUNCTION() [30/36]**

```
GiNaC::REGISTER_FUNCTION (
 atan2 ,
 eval_func(atan2_eval). evalf_func(atan2_evalf). info_func(atan2_info). evalf_↔
func(atan2_evalf). derivative_func(atan2_deriv))
```

**5.1.3.441 sinh\_evalf()**

```
static ex GiNaC::sinh_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.442 sinh\_eval()**

```
static ex GiNaC::sinh_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sin\(\)](#), [sinh\(\)](#), [sqrt\(\)](#), and [x](#).

**5.1.3.443 sinh\_deriv()**

```
static ex GiNaC::sinh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [cosh\(\)](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.444 sinh\_real\_part()**

```
static ex GiNaC::sinh_real_part (
 const ex & x) [static]
```

References [cos\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.445 sinh\_imag\_part()**

```
static ex GiNaC::sinh_imag_part (
 const ex & x) [static]
```

References [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), and [x](#).

**5.1.3.446 sinh\_conjugate()**

```
static ex GiNaC::sinh_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.447 REGISTER\_FUNCTION() [31/36]**

```
GiNaC::REGISTER_FUNCTION (
 sinh ,
 eval_func(sinh_eval). evalf_func(sinh_evalf). info_func(atan_info). derivative↔
_func(sinh_deriv). real_part_func(sinh_real_part). imag_part_func(sinh_imag_part). conjugate↔
_func(sinh_conjugate). latex_name("\\sinh"))
```

**5.1.3.448 cosh\_evalf()**

```
static ex GiNaC::cosh_evalf (
 const ex & x) [static]
```

References [cosh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.449 cosh\_eval()**

```
static ex GiNaC::cosh_eval (
 const ex & x) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [cos\(\)](#), [cosh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sqrt\(\)](#), and [x](#).

**5.1.3.450 cosh\_deriv()**

```
static ex GiNaC::cosh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [GINAC\\_ASSERT](#), [sinh\(\)](#), and [x](#).

**5.1.3.451 cosh\_real\_part()**

```
static ex GiNaC::cosh_real_part (
 const ex & x) [static]
```

References [cos\(\)](#), [cosh\(\)](#), [imag\\_part\(\)](#), [real\\_part\(\)](#), and [x](#).

**5.1.3.452 cosh\_imag\_part()**

```
static ex GiNaC::cosh_imag_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [sin\(\)](#), [sinh\(\)](#), and [x](#).

**5.1.3.453 cosh\_conjugate()**

```
static ex GiNaC::cosh_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [cosh\(\)](#), and [x](#).

**5.1.3.454 REGISTER\_FUNCTION() [32/36]**

```
GiNaC::REGISTER_FUNCTION (
 cosh ,
 eval_func(cosh_eval). evalf_func(cosh_evalf). info_func(exp_info). derivative↔
_func(cosh_deriv). real_part_func(cosh_real_part). imag_part_func(cosh_imag_part). conjugate↔
_func(cosh_conjugate). latex_name("\\cosh"))
```

**5.1.3.455 tanh\_evalf()**

```
static ex GiNaC::tanh_evalf (
 const ex & x) [static]
```

References [GiNaC::basic::hold\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.456 tanh\_eval()**

```
static ex GiNaC::tanh_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [acosh\(\)](#), [asinh\(\)](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [is\\_ex\\_the\\_function](#), [GiNaC::ex::is\\_zero\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::op\(\)](#), [Pi](#), [real\(\)](#), [sqrt\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.457 tanh\_deriv()**

```
static ex GiNaC::tanh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [GINAC\\_ASSERT](#), [tanh\(\)](#), and [x](#).

**5.1.3.458 tanh\_series()**

```
static ex GiNaC::tanh_series (
 const ex & x,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [cosh\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::relational::lhs\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::info\\_flags::odd](#), [options](#), [order](#), [Pi](#), [sinh\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

**5.1.3.459 tanh\_real\_part()**

```
static ex GiNaC::tanh_real_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.460 tanh\_imag\_part()**

```
static ex GiNaC::tanh_imag_part (
 const ex & x) [static]
```

References [imag\\_part\(\)](#), [real\\_part\(\)](#), [tan\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.461 tanh\_conjugate()**

```
static ex GiNaC::tanh_conjugate (
 const ex & x) [static]
```

References [GiNaC::ex::conjugate\(\)](#), [tanh\(\)](#), and [x](#).

**5.1.3.462 REGISTER\_FUNCTION() [33/36]**

```
GiNaC::REGISTER_FUNCTION (
 tanh ,
 eval_func(tanh_eval). evalf_func(tanh_evalf). info_func(atan_info). derivative↵
 _func(tanh_deriv). series_func(tanh_series). real_part_func(tanh_real_part). imag_part_↵
 func(tanh_imag_part). conjugate_func(tanh_conjugate). latex_name("\\tanh")
```

**5.1.3.463 asinh\_evalf()**

```
static ex GiNaC::asinh_evalf (
 const ex & x) [static]
```

References [asinh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.464 asinh\_eval()**

```
static ex GiNaC::asinh_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [asinh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [x](#).

**5.1.3.465 asinh\_deriv()**

```
static ex GiNaC::asinh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.466 asinh\_conjugate()**

```
static ex GiNaC::asinh_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [asinh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), and [x](#).

**5.1.3.467 REGISTER\_FUNCTION() [34/36]**

```
GiNaC::REGISTER_FUNCTION (
 asinh ,
 eval_func(asinh_eval). evalf_func(asinh_evalf). info_func(atan_info). derivative←
_func(asinh_deriv). conjugate_func(asinh_conjugate))
```

**5.1.3.468 acosh\_evalf()**

```
static ex GiNaC::acosh_evalf (
 const ex & x) [static]
```

References [acosh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.469 acosh\_eval()**

```
static ex GiNaC::acosh_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [acosh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [I](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [Pi](#), and [x](#).

**5.1.3.470 acosh\_deriv()**

```
static ex GiNaC::acosh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.471 acosh\_conjugate()**

```
static ex GiNaC::acosh_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [acosh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.472 REGISTER\_FUNCTION() [35/36]**

```
GiNaC::REGISTER_FUNCTION (
 acosh ,
 eval_func(acosh_eval). evalf_func(acosh_evalf). info_func(asin_info). derivative←
_func(acosh_deriv). conjugate_func(acosh_conjugate))
```

**5.1.3.473 atanh\_evalf()**

```
static ex GiNaC::atanh_evalf (
 const ex & x) [static]
```

References [atanh\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

**5.1.3.474 atanh\_eval()**

```
static ex GiNaC::atanh_eval (
 const ex & x) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex\\_1](#), [atanh\(\)](#), [GiNaC::info\\_flags::crational](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), and [x](#).

**5.1.3.475 atanh\_deriv()**

```
static ex GiNaC::atanh_deriv (
 const ex & x,
 unsigned deriv_param) [static]
```

References [\\_ex1](#), [\\_ex2](#), [\\_ex\\_1](#), [GINAC\\_ASSERT](#), and [x](#).

**5.1.3.476 atanh\_series()**

```
static ex GiNaC::atanh_series (
 const ex & arg,
 const relational & rel,
 int order,
 unsigned options) [static]
```

References [\\_ex0](#), [\\_ex1](#), [\\_ex1\\_2](#), [\\_ex\\_1](#), [\\_ex\\_1\\_2](#), [abs\(\)](#), [atanh\(\)](#), [csgn\(\)](#), [GINAC\\_ASSERT](#), [I](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::relational::lhs\(\)](#), [log\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [options](#), [order](#), [Pi](#), [GiNaC::info\\_flags::real](#), [GiNaC::relational::rhs\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::series\\_options::suppress\\_branchcut](#).

**5.1.3.477 atanh\_conjugate()**

```
static ex GiNaC::atanh_conjugate (
 const ex & x) [static]
```

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [atanh\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

**5.1.3.478 REGISTER\_FUNCTION() [36/36]**

```
GiNaC::REGISTER_FUNCTION (
 atanh ,
 eval_func(atanh_eval). evalf_func(atanh_evalf). info_func(asin_info). derivative←
 _func(atanh_deriv). series_func(atanh_series). conjugate_func(atanh_conjugate))
```

**5.1.3.479 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [11/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 integral ,
 basic ,
 print_func< print_dflt > &::do_print. print_func< print_python > &::do_print.
 print_func< print_latex > &::do_print_latex)
```

**5.1.3.480 subsvalue()**

```
ex GiNaC::subsvalue (
 const ex & var,
 const ex & value,
 const ex & fun)
```

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::subs\(\)](#), and [value](#).

Referenced by [adaptivesimpson\(\)](#).

**5.1.3.481 adaptivesimpson()**

```
GiNaC::ex GiNaC::adaptivesimpson (
 const ex & x,
 const ex & a_in,
 const ex & b_in,
 const ex & f,
 const ex & error)
```

Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.

Parameters are integration variable, left boundary, right boundary, function to be integrated and the relative integration error. The function should evalf into a number after substituting the integration variable by a number. Another thing to note is that this implementation is no good at integrating functions with discontinuities.

References [abs\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::integral::max\\_integration\\_level](#), [GiNaC::ex::subs\(\)](#), [subsvalue\(\)](#), and [x](#).

Referenced by [GiNaC::integral::evalf\(\)](#).

**5.1.3.482 GINAC\_BIND\_UNARCHIVER() [21/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
 integral)
```

**5.1.3.483 GINAC\_DECLARE\_UNARCHIVER() [22/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 integral)
```

**5.1.3.484 ifactor()**

```
ex GiNaC::ifactor (
 const numeric & n)
```

Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $\text{lst}(\text{lst}(p_1, \dots, pr), \text{lst}(a_1, \dots, ar))$  such that  $n = p_1^{a_1} \dots pr^{ar}$ .

References [GiNaC::container< C >::append\(\)](#), [irem\(\)](#), [n](#), and [GiNaC::info\\_flags::prime](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), and [kronecker\\_symbol\(\)](#).

**5.1.3.485 is\_discriminant\_of\_quadratic\_number\_field()**

```
bool GiNaC::is_discriminant_of_quadratic_number_field (
 const numeric & n)
```

Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.

Returns false otherwise.

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [abs\(\)](#), [ifactor\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [mod\(\)](#), [n](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

Referenced by [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#).

**5.1.3.486 kronecker\_symbol()**

```
numeric GiNaC::kronecker_symbol (
 const numeric & a,
 const numeric & n)
```

Returns the Kronecker symbol  $a$ : integer  $n$ : integer.

This routine defines  $\text{kronecker\_symbol}(1,0) = 1$   $\text{kronecker\_symbol}(-1,0) = 1$   $\text{kronecker\_symbol}(a,0) = 0$ ,  $a \neq 1, -1$

In particular  $\text{kronecker\_symbol}(-1,0) = 1$  (in agreement with Sage)

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), [ifactor\(\)](#), [GiNaC::numeric::is\\_even\(\)](#), [n](#), [GiNaC::container< C >::op\(\)](#), [pow\(\)](#), and [unit](#).

Referenced by [primitive\\_dirichlet\\_character\(\)](#).

**5.1.3.487 primitive\_dirichlet\_character()**

```
numeric GiNaC::primitive_dirichlet_character (
 const numeric & n,
 const numeric & a)
```

Defines a primitive Dirichlet character through the Kronecker symbol.

$n$ : integer  $a$ : discriminant of a quadratic field  $|a|$ : conductor

The character takes the values  $-1, 0, 1$ .

References [kronecker\\_symbol\(\)](#), and [n](#).

Referenced by [dirichlet\\_character\(\)](#), and [generalised\\_Bernoulli\\_number\(\)](#).

**5.1.3.488 `dirichlet_character()`**

```
numeric GiNaC::dirichlet_character (
 const numeric & n,
 const numeric & a,
 const numeric & N)
```

Defines a Dirichlet character through the Kronecker symbol.

n: integer a: discriminant of a quadratic field  $|a|$ : conductor N: modulus, needs to be multiple of  $|a|$

The character takes the values -1,0,1.

References [gcd\(\)](#), [n](#), and [primitive\\_dirichlet\\_character\(\)](#).

**5.1.3.489 `generalised_Bernoulli_number()`**

```
numeric GiNaC::generalised_Bernoulli_number (
 const numeric & k,
 const numeric & b)
```

The generalised Bernoulli number.

k: index / modular weight

b: discriminant of a quadratic field, defines primitive character  $\psi$   $M=|b|$ : conductor of primitive character  $\psi$

The generalised Bernoulli number is computed from the series expansion of the generating function. The generating function is given in eq.(34), arXiv:1704.08895

References [abs\(\)](#), [GiNaC::ex::coeff\(\)](#), [exp\(\)](#), [factorial\(\)](#), [k](#), [primitive\\_dirichlet\\_character\(\)](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [x](#).

**5.1.3.490 `Bernoulli_polynomial()`**

```
ex GiNaC::Bernoulli_polynomial (
 const numeric & k,
 const ex & x)
```

The Bernoulli polynomials.

References [GiNaC::ex::coeff\(\)](#), [exp\(\)](#), [factorial\(\)](#), [k](#), [GiNaC::ex::series\(\)](#), [series\\_to\\_poly\(\)](#), and [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#).

**5.1.3.491 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [12/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 integration_kernel ,
 basic ,
 print_func< print_context > &::do_print)
```

**5.1.3.492 GINAC\_BIND\_UNARCHIVER()** [22/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 integration_kernel)
```

**5.1.3.493 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [13/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 basic_log_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.494 GINAC\_BIND\_UNARCHIVER()** [23/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 basic_log_kernel)
```

**5.1.3.495 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [14/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 multiple_polylog_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

References [\\_ex1](#).

**5.1.3.496 GINAC\_BIND\_UNARCHIVER()** [24/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 multiple_polylog_kernel)
```

**5.1.3.497 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [15/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 ELi_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.498 GINAC\_BIND\_UNARCHIVER()** [25/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 ELi_kernel)
```

**5.1.3.499 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [16/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 Ebar_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.500 GINAC\_BIND\_UNARCHIVER()** [26/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 Ebar_kernel)
```

**5.1.3.501 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [17/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 Kronecker_dtau_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.502 GINAC\_BIND\_UNARCHIVER()** [27/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 Kronecker_dtau_kernel)
```

**5.1.3.503 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [18/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 Kronecker_dz_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.504 GINAC\_BIND\_UNARCHIVER()** [28/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 Kronecker_dz_kernel)
```

**5.1.3.505 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [19/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 Eisenstein_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.506 GINAC\_BIND\_UNARCHIVER()** [29/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 Eisenstein_kernel)
```

**5.1.3.507 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [20/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 Eisenstein_h_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.508 GINAC\_BIND\_UNARCHIVER()** [30/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 Eisenstein_h_kernel)
```

**5.1.3.509 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [21/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 modular_form_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.510 GINAC\_BIND\_UNARCHIVER()** [31/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 modular_form_kernel)
```

**5.1.3.511 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [22/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 user_defined_kernel ,
 integration_kernel ,
 print_func< print_context > &::do_print)
```

**5.1.3.512 GINAC\_BIND\_UNARCHIVER()** [32/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 user_defined_kernel)
```

**5.1.3.513 GINAC\_DECLARE\_UNARCHIVER()** [23/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 integration_kernel)
```

**5.1.3.514 GINAC\_DECLARE\_UNARCHIVER()** [24/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 basic_log_kernel)
```

**5.1.3.515 GINAC\_DECLARE\_UNARCHIVER()** [25/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 multiple_polylog_kernel)
```

**5.1.3.516 GINAC\_DECLARE\_UNARCHIVER()** [26/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 ELi_kernel)
```

**5.1.3.517 GINAC\_DECLARE\_UNARCHIVER()** [27/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 Ebar_kernel)
```

**5.1.3.518 GINAC\_DECLARE\_UNARCHIVER()** [28/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 Kronecker_dtau_kernel)
```

**5.1.3.519 GINAC\_DECLARE\_UNARCHIVER()** [29/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 Kronecker_dz_kernel)
```

**5.1.3.520 GINAC\_DECLARE\_UNARCHIVER()** [30/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 Eisenstein_kernel)
```

**5.1.3.521 GINAC\_DECLARE\_UNARCHIVER()** [31/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 Eisenstein_h_kernel)
```

**5.1.3.522 GINAC\_DECLARE\_UNARCHIVER()** [32/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 modular_form_kernel)
```

**5.1.3.523 GINAC\_DECLARE\_UNARCHIVER()** [33/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 user_defined_kernel)
```

**5.1.3.524 GINAC\_DECLARE\_UNARCHIVER()** [34/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 lst)
```

**5.1.3.525 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [23/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 matrix ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
::do_print_python_repr)
```

Default ctor.

Initializes to 1 x 1-dimensional zero-matrix.

References [GiNaC::status\\_flags::not\\_shareable](#).

**5.1.3.526 GINAC\_BIND\_UNARCHIVER()** [33/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 matrix)
```

**5.1.3.527 lst\_to\_matrix()**

```
ex GiNaC::lst_to_matrix (
 const lst & l)
```

Convert list of lists to matrix.

References [cols\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [rows\(\)](#).

**5.1.3.528 diag\_matrix()** [1/2]

```
ex GiNaC::diag_matrix (
 const lst & l)
```

Convert list of diagonal elements to matrix.

References [GiNaC::container< C >::nops\(\)](#).

**5.1.3.529 diag\_matrix()** [2/2]

```
ex GiNaC::diag_matrix (
 std::initializer_list< ex > l)
```

**5.1.3.530 unit\_matrix()** [1/2]

```
ex GiNaC::unit_matrix (
 unsigned r,
 unsigned c)
```

Create an r times c unit matrix.

References [\\_ex1](#), [c](#), [GiNaC::status\\_flags::evaluated](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [unit\\_matrix\(\)](#).

**5.1.3.531 symbolic\_matrix()** [1/2]

```
ex GiNaC::symbolic_matrix (
 unsigned r,
 unsigned c,
 const std::string & base_name,
 const std::string & tex_base_name)
```

Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

The base name for LaTeX output is specified separately.

References [c](#), [GiNaC::status\\_flags::evaluated](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [symbolic\\_matrix\(\)](#).

**5.1.3.532 reduced\_matrix()**

```
ex GiNaC::reduced_matrix (
 const matrix & m,
 unsigned r,
 unsigned c)
```

Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.

The determinant of the result is the Minor r, c.

References [c](#), [cols\(\)](#), [GiNaC::status\\_flags::evaluated](#), [m](#), [r](#), [rows\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.533 sub\_matrix()**

```
ex GiNaC::sub_matrix (
 const matrix & m,
 unsigned r,
 unsigned nr,
 unsigned c,
 unsigned nc)
```

Return the nr times nc submatrix starting at position r, c of matrix m.

References [c](#), [GiNaC::status\\_flags::evaluated](#), [m](#), [r](#), and [GiNaC::basic::setflag\(\)](#).

**5.1.3.534 GINAC\_DECLARE\_UNARCHIVER()** [35/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 matrix)
```

**5.1.3.535 nops()** [2/2]

```
size_t GiNaC::nops (
 const matrix & m) [inline]
```

References [m](#).

**5.1.3.536 expand()** [2/2]

```
ex GiNaC::expand (
 const matrix & m,
 unsigned options = 0) [inline]
```

References [GiNaC::ex::expand\(\)](#), [m](#), and [options](#).

**5.1.3.537 evalf()** [2/2]

```
ex GiNaC::evalf (
 const matrix & m) [inline]
```

References [GiNaC::ex::evalf\(\)](#), and [m](#).

**5.1.3.538 rows()**

```
unsigned GiNaC::rows (
 const matrix & m) [inline]
```

References [m](#).

Referenced by [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

**5.1.3.539 cols()**

```
unsigned GiNaC::cols (
 const matrix & m) [inline]
```

References [m](#).

Referenced by [lst\\_to\\_matrix\(\)](#), and [reduced\\_matrix\(\)](#).

#### 5.1.3.540 transpose()

```
matrix GiNaC::transpose (
 const matrix & m) [inline]
```

References [m](#), and [GiNaC::matrix::transpose\(\)](#).

#### 5.1.3.541 determinant()

```
ex GiNaC::determinant (
 const matrix & m,
 unsigned options = determinant_algo::automatic) [inline]
```

References [m](#), and [options](#).

#### 5.1.3.542 trace()

```
ex GiNaC::trace (
 const matrix & m) [inline]
```

References [m](#).

#### 5.1.3.543 charpoly()

```
ex GiNaC::charpoly (
 const matrix & m,
 const ex & lambda) [inline]
```

References [m](#).

#### 5.1.3.544 inverse() [1/3]

```
matrix GiNaC::inverse (
 const matrix & m) [inline]
```

References [GiNaC::solve\\_algo::automatic](#), [GiNaC::matrix::inverse\(\)](#), and [m](#).

#### 5.1.3.545 inverse() [2/3]

```
matrix GiNaC::inverse (
 const matrix & m,
 unsigned algo) [inline]
```

References [GiNaC::matrix::inverse\(\)](#), and [m](#).

**5.1.3.546 rank()** [1/2]

```
unsigned GiNaC::rank (
 const matrix & m) [inline]
```

References [m](#), and [GiNaC::matrix::rank\(\)](#).

**5.1.3.547 rank()** [2/2]

```
unsigned GiNaC::rank (
 const matrix & m,
 unsigned solve_algo) [inline]
```

References [m](#).

**5.1.3.548 unit\_matrix()** [2/2]

```
ex GiNaC::unit_matrix (
 unsigned x) [inline]
```

Create a x times x unit matrix.

References [unit\\_matrix\(\)](#), and [x](#).

**5.1.3.549 symbolic\_matrix()** [2/2]

```
ex GiNaC::symbolic_matrix (
 unsigned r,
 unsigned c,
 const std::string & base_name) [inline]
```

Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.

References [c](#), [r](#), and [symbolic\\_matrix\(\)](#).

**5.1.3.550 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [24/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 mul ,
 expairseq ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
 print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_tree > &::do_↵
 print_tree. print_func< print_python_repr > &::do_print_python_repr)
```

**5.1.3.551 tryfactsubs()**

```
bool GiNaC::tryfactsubs (
 const ex & origfactor,
 const ex & patternfactor,
 int & nummatches,
 exmap & repls)
```

References [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::match\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::power::subs\(\)](#).

**5.1.3.552 algebraic\_match\_mul\_with\_mul()**

```
bool GiNaC::algebraic_match_mul_with_mul (
 const mul & e,
 const ex & pat,
 exmap & repls,
 int factor,
 int & nummatches,
 const std::vector< bool > & substed,
 std::vector< bool > & matched)
```

Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.

This matching is in the sense of algebraic substitutions. Matching starts with pat.op(factor) of the pattern because the factors before this one have already been matched. The (possibly updated) number of matches is in nummatches. substed[i] is true for factors that already have been replaced by previous substitutions and matched[i] is true for factors that have been matched by the current match.

References [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::op\(\)](#), and [tryfactsubs\(\)](#).

Referenced by [algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), and [GiNaC::mul::has\(\)](#).

**5.1.3.553 GINAC\_BIND\_UNARCHIVER() [34/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
 mul)
```

**5.1.3.554 GINAC\_DECLARE\_UNARCHIVER() [36/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 mul)
```

**5.1.3.555 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [25/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 ncmul ,
 exprseq ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print↔
 _tree. print_func< print_csrc > &::do_print_csrc. print_func< print_python_repr > &::do_↔
 print_csrc)
```

**5.1.3.556 reeval\_ncmul()**

```
ex GiNaC::reeval_ncmul (
 const exvector & v)
```

**5.1.3.557 hold\_ncmul()**

```
ex GiNaC::hold_ncmul (
 const exvector & v)
```

Referenced by [GiNaC::basic::eval\\_ncmul\(\)](#), [GiNaC::color::eval\\_ncmul\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::eval\\_ncmul\(\)](#).

**5.1.3.558 GINAC\_BIND\_UNARCHIVER() [35/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
 ncmul)
```

**5.1.3.559 GINAC\_DECLARE\_UNARCHIVER() [37/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 ncmul)
```

**5.1.3.560 get\_first\_symbol()**

```
static bool GiNaC::get_first_symbol (
 const ex & e,
 ex & x) [static]
```

Return pointer to first symbol found in expression.

Due to [GiNaC](#)'s internal ordering of terms, it may not be obvious which symbol this function returns for a given expression.

**Parameters**

|          |                               |
|----------|-------------------------------|
| <i>e</i> | expression to search          |
| <i>x</i> | first symbol found (returned) |

**Returns**

"false" if no symbol was found, "true" otherwise

References [get\\_first\\_symbol\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [divide\(\)](#), [frac\\_cancel\(\)](#), [get\\_first\\_symbol\(\)](#), and [GiNaC::ex::unit\(\)](#).

**5.1.3.561 add\_symbol()**

```
static void GiNaC::add_symbol (
 const ex & s,
 sym_desc_vec & v) [static]
```

Referenced by [collect\\_symbols\(\)](#).

**5.1.3.562 collect\_symbols()**

```
static void GiNaC::collect_symbols (
 const ex & e,
 sym_desc_vec & v) [static]
```

References [add\\_symbol\(\)](#), [collect\\_symbols\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [collect\\_symbols\(\)](#), and [get\\_symbol\\_stats\(\)](#).

**5.1.3.563 get\_symbol\_stats()**

```
static void GiNaC::get_symbol_stats (
 const ex & a,
 const ex & b,
 sym_desc_vec & v) [static]
```

Collect statistical information about symbols in polynomials.

This function fills in a vector of "sym\_desc" structs which contain information about the highest and lowest degrees of all symbols that appear in two polynomials. The vector is then sorted by minimum degree (lowest to highest). The information gathered by this function is used by the GCD routines to identify trivial factors and to determine which variable to choose as the main variable for GCD computation.

**Parameters**

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>a</i> | first multivariate polynomial                          |
| <i>b</i> | second multivariate polynomial                         |
| <i>v</i> | vector of <a href="#">sym_desc</a> structs (filled in) |

References [collect\\_symbols\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [GiNaC::ex::ldegree\(\)](#), and [GiNaC::ex::nops\(\)](#).

Referenced by [divide\\_in\\_z\(\)](#), [gcd\(\)](#), and [sqrfree\(\)](#).

**5.1.3.564 lcmcoeff()**

```
static numeric GiNaC::lcmcoeff (
 const ex & e,
 const numeric & l) [static]
```

References [\\_num1\\_p](#), [c](#), [denom\(\)](#), [GiNaC::ex::info\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::info\\_flags::rational](#).

Referenced by [heur\\_gcd\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), and [multiply\\_lcm\(\)](#).

**5.1.3.565 lcm\_of\_coefficients\_denominators()**

```
static numeric GiNaC::lcm_of_coefficients_denominators (
 const ex & e) [static]
```

Compute LCM of denominators of coefficients of a polynomial.

Given a polynomial with rational coefficients, this function computes the LCM of the denominators of all coefficients. This can be used to bring a polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$ .

**Parameters**

|          |                                                |
|----------|------------------------------------------------|
| <i>e</i> | multivariate polynomial (need not be expanded) |
|----------|------------------------------------------------|

**Returns**

LCM of denominators of coefficients

References [\\_num1\\_p](#), and [lcmcoeff\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [heur\\_gcd\(\)](#), and [sqrfree\(\)](#).

**5.1.3.566 multiply\_lcm()**

```
static ex GiNaC::multiply_lcm (
 const ex & e,
 const numeric & lcm) [static]
```

Bring polynomial from  $\mathbb{Q}[X]$  to  $\mathbb{Z}[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.

**Parameters**

|            |                                                |
|------------|------------------------------------------------|
| <i>e</i>   | multivariate polynomial (need not be expanded) |
| <i>lcm</i> | LCM to multiply in                             |

References [\\_num1\\_p](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [frac\\_cancel\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).

**5.1.3.567 quo()**

```
ex GiNaC::quo (
 const ex & a,
 const ex & b,
 const ex & x,
 bool check_args)
```

Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

## Parameters

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <i>a</i>          | first polynomial in x (dividend)                                                      |
| <i>b</i>          | second polynomial in x (divisor)                                                      |
| <i>x</i>          | a and b are polynomials in x                                                          |
| <i>check_args</i> | check whether a and b are polynomials with rational coefficients (defaults to "true") |

## Returns

quotient of a and b in  $\mathbb{Q}[x]$

References [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [decomp\\_rational\(\)](#), [GiNaC::ex::primpart\(\)](#), [sqrfree\\_parfrac\(\)](#), [sqrfree\\_yun\(\)](#), and [GiNaC::ex::unitcontprim\(\)](#).

5.1.3.568 **rem()**

```
ex GiNaC::rem (
 const ex & a,
 const ex & b,
 const ex & x,
 bool check_args)
```

Remainder [r\(x\)](#) of polynomials a(x) and b(x) in  $\mathbb{Q}[x]$ .

It satisfies  $a(x)=b(x)*q(x)+r(x)$ .

## Parameters

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <i>a</i>          | first polynomial in x (dividend)                                                      |
| <i>b</i>          | second polynomial in x (divisor)                                                      |
| <i>x</i>          | a and b are polynomials in x                                                          |
| <i>check_args</i> | check whether a and b are polynomials with rational coefficients (defaults to "true") |

## Returns

remainder of a(x) and b(x) in  $\mathbb{Q}[x]$

References [\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [decomp\\_rational\(\)](#), and [sqrfree\\_parfrac\(\)](#).

5.1.3.569 **decomp\_rational()**

```
ex GiNaC::decomp_rational (
 const ex & a,
 const ex & x)
```

Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .

## Parameters

|          |                                    |
|----------|------------------------------------|
| <i>a</i> | rational function in <i>x</i>      |
| <i>x</i> | <i>a</i> is a function of <i>x</i> |

## Returns

decomposed function.

References [denom\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [quo\(\)](#), [rem\(\)](#), and [x](#).

**5.1.3.570 prem()**

```
ex GiNaC::prem (
 const ex & a,
 const ex & b,
 const ex & x,
 bool check_args)
```

Pseudo-remainder of polynomials *a*(*x*) and *b*(*x*) in  $\mathbb{Q}[x]$ .

## Parameters

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <i>a</i>          | first polynomial in <i>x</i> (dividend)                                                             |
| <i>b</i>          | second polynomial in <i>x</i> (divisor)                                                             |
| <i>x</i>          | <i>a</i> and <i>b</i> are polynomials in <i>x</i>                                                   |
| <i>check_args</i> | check whether <i>a</i> and <i>b</i> are polynomials with rational coefficients (defaults to "true") |

## Returns

pseudo-remainder of *a*(*x*) and *b*(*x*) in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [sr\\_gcd\(\)](#).

**5.1.3.571 sprem()**

```
ex GiNaC::sprem (
 const ex & a,
 const ex & b,
 const ex & x,
 bool check_args)
```

Sparse pseudo-remainder of polynomials *a*(*x*) and *b*(*x*) in  $\mathbb{Q}[x]$ .

## Parameters

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <i>a</i>          | first polynomial in <i>x</i> (dividend)                                                             |
| <i>b</i>          | second polynomial in <i>x</i> (divisor)                                                             |
| <i>x</i>          | <i>a</i> and <i>b</i> are polynomials in <i>x</i>                                                   |
| <i>check_args</i> | check whether <i>a</i> and <i>b</i> are polynomials with rational coefficients (defaults to "true") |

**Returns**

sparse pseudo-remainder of  $a(x)$  and  $b(x)$  in  $\mathbb{Q}[x]$

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

**5.1.3.572 divide()**

```
bool GiNaC::divide (
 const ex & a,
 const ex & b,
 ex & q,
 bool check_args)
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Q}[X]$ .

**Parameters**

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <i>a</i>          | first multivariate polynomial (dividend)                                              |
| <i>b</i>          | second multivariate polynomial (divisor)                                              |
| <i>q</i>          | quotient (returned)                                                                   |
| <i>check_args</i> | check whether a and b are polynomials with rational coefficients (defaults to "true") |

**Returns**

"true" when exact division succeeds (quotient returned in q), "false" otherwise (q left untouched)

References [\\_ex0](#), [\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [get\\_first\\_symbol\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [x](#).

Referenced by [divide\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [gcd\(\)](#), [quo\(\)](#), and [rem\(\)](#).

**5.1.3.573 divide\_in\_z()**

```
static bool GiNaC::divide_in_z (
 const ex & a,
 const ex & b,
 ex & q,
 sym_desc_vec::const_iterator var) [static]
```

Exact polynomial division of  $a(X)$  by  $b(X)$  in  $\mathbb{Z}[X]$ .

This functions works like [divide\(\)](#) but the input and output polynomials are in  $\mathbb{Z}[X]$  instead of  $\mathbb{Q}[X]$  (i.e. they have integer coefficients). Unlike [divide\(\)](#), it doesn't check whether the input polynomials really are integer polynomials, so be careful of what you pass in. Also, you have to run [get\\_symbol\\_stats\(\)](#) over the input polynomials before calling this function and pass an iterator to the first element of the [sym\\_desc](#) vector. This function is used internally by the [heur\\_gcd\(\)](#).

## Parameters

|            |                                                                         |
|------------|-------------------------------------------------------------------------|
| <i>a</i>   | first multivariate polynomial (dividend)                                |
| <i>b</i>   | second multivariate polynomial (divisor)                                |
| <i>q</i>   | quotient (returned)                                                     |
| <i>var</i> | iterator to first element of vector of <a href="#">sym_desc</a> structs |

## Returns

"true" when exact division succeeds (the quotient is returned in q), "false" otherwise.

## See also

[get\\_symbol\\_stats](#), [heur\\_gcd](#)

References [\\_ex0](#), [\\_ex1](#), [\\_num0\\_p](#), [\\_num1\\_p](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::find\(\)](#), [get\\_symbol\\_stats\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [qbar](#), [r](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [divide\\_in\\_z\(\)](#), [heur\\_gcd\\_z\(\)](#), and [sr\\_gcd\(\)](#).

5.1.3.574 [sr\\_gcd\(\)](#)

```
static ex GiNaC::sr_gcd (
 const ex & a,
 const ex & b,
 sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the subresultant PRS algorithm.

This function is used internally by [gcd\(\)](#).

## Parameters

|            |                                                                         |
|------------|-------------------------------------------------------------------------|
| <i>a</i>   | first multivariate polynomial                                           |
| <i>b</i>   | second multivariate polynomial                                          |
| <i>var</i> | iterator to first element of vector of <a href="#">sym_desc</a> structs |

## Returns

the GCD as a new expression

## See also

[gcd](#)

References [\\_ex0](#), [\\_ex1](#), [c](#), [GiNaC::ex::content\(\)](#), [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [GiNaC::ex::lcoeff\(\)](#), [pow\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [psi\(\)](#), [r](#), and [x](#).

Referenced by [gcd\(\)](#).

### 5.1.3.575 interpolate()

```
static ex GiNaC::interpolate (
 const ex & gamma,
 const numeric & xi,
 const ex & x,
 int degree_hint = 1) [static]
```

xi-adic polynomial interpolation

References [GiNaC::numeric::inverse\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pow\(\)](#), [GiNaC::ex::smod\(\)](#), and [x](#).

Referenced by [heur\\_gcd\\_z\(\)](#).

### 5.1.3.576 heur\_gcd\_z()

```
static bool GiNaC::heur_gcd_z (
 ex & res,
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb,
 sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

#### Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>a</i>   | first integer multivariate polynomial (expanded)                                 |
| <i>b</i>   | second integer multivariate polynomial (expanded)                                |
| <i>ca</i>  | cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor |
| <i>cb</i>  | cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor |
| <i>var</i> | iterator to first element of vector of <a href="#">sym_desc</a> structs          |
| <i>res</i> | the GCD (returned)                                                               |

#### Returns

true if GCD was computed, false otherwise.

#### See also

[gcd](#)

#### Exceptions

|                              |
|------------------------------|
| <code>gcdheu_failed()</code> |
|------------------------------|

References [GiNaC::ex::degree\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::numeric::int\\_length\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [interpolate\(\)](#), [GiNaC::numeric::inverse\(\)](#), [iquo\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [isqrt\(\)](#), [GiNaC::ex::max\\_coefficient\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [heur\\_gcd\(\)](#), and [heur\\_gcd\\_z\(\)](#).

### 5.1.3.577 [heur\\_gcd\(\)](#)

```
static bool GiNaC::heur_gcd (
 ex & res,
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb,
 sym_desc_vec::const_iterator var) [static]
```

Compute GCD of multivariate polynomials using the heuristic GCD algorithm.

[get\\_symbol\\_stats\(\)](#) must have been called previously with the input polynomials and an iterator to the first element of the [sym\\_desc](#) vector passed in. This function is used internally by [gcd\(\)](#).

#### Parameters

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| <i>a</i>   | first rational multivariate polynomial (expanded)                                |
| <i>b</i>   | second rational multivariate polynomial (expanded)                               |
| <i>ca</i>  | cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor |
| <i>cb</i>  | cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor |
| <i>var</i> | iterator to first element of vector of <a href="#">sym_desc</a> structs          |
| <i>res</i> | the GCD (returned)                                                               |

#### Returns

true if GCD was computed, false otherwise.

#### See also

[heur\\_gcd\\_z](#)  
[gcd](#)

References [heur\\_gcd\\_z\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), and [lcmcoeff\(\)](#).

Referenced by [gcd\(\)](#).

### 5.1.3.578 [gcd\\_pf\\_pow\(\)](#)

```
static ex GiNaC::gcd_pf_pow (
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb) [static]
```

References [\\_ex1](#), [expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::ldegree\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_pow\(\)](#).

### 5.1.3.579 gcd\_pf\_mul()

```
static ex GiNaC::gcd_pf_mul (
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb) [static]
```

References [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

Referenced by [gcd\(\)](#), and [gcd\\_pf\\_mul\(\)](#).

### 5.1.3.580 gcd() [1/2]

```
ex GiNaC::gcd (
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb,
 bool check_args,
 unsigned options)
```

Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $\mathbb{Z}[X]$ .

Optionally also compute the cofactors of  $a$  and  $b$ , defined by  $a = ca * \text{gcd}(a, b)$  and  $b = cb * \text{gcd}(a, b)$ .

#### Parameters

|                   |                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------|
| <i>a</i>          | first multivariate polynomial                                                             |
| <i>b</i>          | second multivariate polynomial                                                            |
| <i>ca</i>         | pointer to expression that will receive the cofactor of $a$ , or nullptr                  |
| <i>cb</i>         | pointer to expression that will receive the cofactor of $b$ , or nullptr                  |
| <i>check_args</i> | check whether $a$ and $b$ are polynomials with rational coefficients (defaults to "true") |
| <i>options</i>    | see <a href="#">GiNaC::gcd_options</a>                                                    |

#### Returns

the GCD as a new expression

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [expand\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [get\\_symbol\\_stats\(\)](#), [heur\\_gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [n](#), [GiNaC::gcd\\_options::no\\_heur\\_gcd](#), [GiNaC::gcd\\_options::no\\_part\\_factored](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [pow\(\)](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [sr\\_gcd\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [GiNaC::gcd\\_options::use\\_sr\\_gcd](#), and [x](#).

Referenced by [GiNaC::ex::content\(\)](#), [dirichlet\\_character\(\)](#), [find\\_common\\_factor\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_mul\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [heur\\_gcd\\_z\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [GiNaC::add::normal\(\)](#), [sqrfree\\_yun\(\)](#), and [sr\\_gcd\(\)](#).

**5.1.3.581 gcd\_pf\_pow\_pow()**

```
static ex GiNaC::gcd_pf_pow_pow (
 const ex & a,
 const ex & b,
 ex * ca,
 ex * cb) [static]
```

References [\\_ex1](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::op\(\)](#), and [pow\(\)](#).

Referenced by [gcd\\_pf\\_pow\(\)](#).

**5.1.3.582 lcm() [1/2]**

```
ex GiNaC::lcm (
 const ex & a,
 const ex & b,
 bool check_args)
```

Compute LCM (Least Common Multiple) of multivariate polynomials in  $\mathbb{Z}[X]$ .

**Parameters**

|                   |                                                                                       |
|-------------------|---------------------------------------------------------------------------------------|
| <i>a</i>          | first multivariate polynomial                                                         |
| <i>b</i>          | second multivariate polynomial                                                        |
| <i>check_args</i> | check whether a and b are polynomials with rational coefficients (defaults to "true") |

**Returns**

the LCM as a new expression

References [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [lcm\(\)](#), and [GiNaC::info\\_flags::rational\\_polynomial](#).

Referenced by [GiNaC::add::integer\\_content\(\)](#), [lcm\(\)](#), [lcmcoeff\(\)](#), [multiply\\_lcm\(\)](#), and [sqrfree\(\)](#).

**5.1.3.583 sqrfree\_yun()**

```
static epvector GiNaC::sqrfree_yun (
 const ex & a,
 const symbol & x) [static]
```

Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.

Used internally by [sqrfree\(\)](#).

**Parameters**

|          |                                                                                                                      |
|----------|----------------------------------------------------------------------------------------------------------------------|
| <i>a</i> | multivariate polynomial over $\mathbb{Z}[X]$ , treated here as univariate polynomial in $x$ (needs not be expanded). |
| <i>x</i> | variable to factor in                                                                                                |

**Returns**

vector of expairs (factor, exponent), sorted by exponents

References [\\_ex1](#), [GiNaC::ex::diff\(\)](#), [factors](#), [gcd\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [quo\(\)](#), and [x](#).

Referenced by [sqrfree\(\)](#), and [sqrfree\\_parfrac\(\)](#).

**5.1.3.584 sqrfree()**

```
ex GiNaC::sqrfree (
 const ex & a,
 const lst & l)
```

Compute a square-free factorization of a multivariate polynomial in  $\mathbb{Q}[X]$ .

**Parameters**

|          |                                                                      |
|----------|----------------------------------------------------------------------|
| <i>a</i> | multivariate polynomial over $\mathbb{Q}[X]$ (needs not be expanded) |
| <i>l</i> | lst of variables to factor in, may be left empty for autodetection   |

**Returns**

a square-free factorization of *a*.

**Note**

A polynomial  $p(X) \in C[X]$  is said *square-free* if, whenever any two polynomials  $q(X)$  and  $r(X)$  are such that

$$p(X) = q(X)^2 r(X),$$

we have  $q(X) \in C$ . This means that  $p(X)$  has no repeated factors, apart eventually from constants. Given a polynomial  $p(X) \in C[X]$ , we say that the decomposition

$$p(X) = b \cdot p_1(X)^{a_1} \cdot p_2(X)^{a_2} \cdots p_r(X)^{a_r}$$

is a *square-free factorization* of  $p(X)$  if the following conditions hold:

1.  $b \in C$  and  $b \neq 0$ ;
2.  $a_i$  is a positive integer for  $i = 1, \dots, r$ ;
3. the degree of the polynomial  $p_i$  is strictly positive for  $i = 1, \dots, r$ ;
4. the polynomial  $\prod_{i=1}^r p_i(X)$  is square-free.

Square-free factorizations need not be unique. For example, if  $a_i$  is even, we could change the polynomial  $p_i(X)$  into  $-p_i(X)$ . Observe also that the factors  $p_i(X)$  need not be irreducible polynomials.

References [\\_ex0](#), [GiNaC::container< C >::append\(\)](#), [factors](#), [get\\_symbol\\_stats\(\)](#), [lcm\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), [sqrfree\(\)](#), [sqrfree\\_yun\(\)](#), and [x](#).

Referenced by [sqrfree\(\)](#).

**5.1.3.585 sqrfree\_parfrac()**

```
ex GiNaC::sqrfree_parfrac (
 const ex & a,
 const symbol & x)
```

Compute square-free partial fraction decomposition of rational function  $a(x)$ .

## Parameters

|          |                                                                                  |
|----------|----------------------------------------------------------------------------------|
| <i>a</i> | rational function over $\mathbb{Z}[x]$ , treated as univariate polynomial in $x$ |
| <i>x</i> | variable to factor in                                                            |

## Returns

decomposed rational function

References [\\_ex1](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [degree\(\)](#), [denom\(\)](#), [GiNaC::ex::expand\(\)](#), [factor\(\)](#), [GINAC\\_ASSERT](#), [k](#), [n](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [quo\(\)](#), [r](#), [rem\(\)](#), [rhs\(\)](#), [GiNaC::matrix::solve\(\)](#), [sqrtfree\\_yun\(\)](#), [to\\_int\(\)](#), and [x](#).

5.1.3.586 `replace_with_symbol()` [1/2]

```
static ex GiNaC::replace_with_symbol (
 const ex & e,
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to `repl`, for a later application of [subs\(\)](#). An entry in the replacement table `repl` can be changed in some cases. If it was altered, we need to provide the modifier for the previously build expressions. The modifier is an (ordered) list, because those substitutions need to be done in the incremental order. As an example let us consider a rationalisation of the expression  $e = \exp(2x) \cdot \cos(\exp(2x)+1) \cdot \exp(x)$ . The first factor [GiNaC](#) denotes by something like `symbol1` and will record:  $e = \text{symbol1} \cdot \cos(\text{symbol1} + 1) \cdot \exp(x)$  `repl = {symbol1 :  $\exp(2x)$ }`. Similarly, the second factor would be denoted as `symbol2` and we will have  $e = \text{symbol1} \cdot \text{symbol2} \cdot \exp(x)$  `repl = {symbol1 :  $\exp(2x)$ , symbol2 :  $\cos(\text{symbol1} + 1)$ }`. Denoting the third term as `symbol3` [GiNaC](#) is willing to re-think  $\exp(2x)$  as  $\text{symbol3}^2$  rather than just `symbol1`. Here are two issues: 1) The replacement "`symbol1 -> symbol3^2`" in the previous part of the expression needs to be done outside of the present routine; 2) The pair "`symbol1 :  $\exp(2x)$` " shall be deleted from the replacement table `repl`. However, this will create illegal substitution "`symbol2 :  $\cos(\text{symbol1} + 1)$` " with undefined `symbol1`. These both problems are mitigated through the additions of the record "`symbol1==symbol3^2`" to the list modifier. Changed length of the modifier signals to the calling code that the previous portion of the expression needs to be altered (it solves 1). Thus [GiNaC](#) can record now  $e = \text{symbol3}^2 \cdot \text{symbol2} \cdot \text{symbol3}$  `repl = {symbol2 :  $\cos(\text{symbol1} + 1)$ , symbol3 :  $\exp(x)$ }` `modifier = {symbol1==symbol3^2}`. Then, doing the backward substitutions the list modifier will be used to restore such iterative substitutions in the right way (this solves 2).

## See also

[ex::normal](#)

References [\\_ex\\_1](#), [GiNaC::container< C >::append\(\)](#), [degree\(\)](#), [denom\(\)](#), [exp\(\)](#), [GiNaC::ex::find\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_ex\\_the\\_function](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [normal\(\)](#), [numer\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::basic::normal\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::basic::to\\_polynomial\(\)](#), [GiNaC::numeric::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::basic::to\\_rational\(\)](#), [GiNaC::numeric::to\\_rational\(\)](#), and [GiNaC::power::to\\_rational\(\)](#).

**5.1.3.587 replace\_with\_symbol()** [2/2]

```
static ex GiNaC::replace_with_symbol (
 const ex & e,
 exmap & repl) [static]
```

Create a symbol for replacing the expression "e" (or return a previously assigned symbol).

The symbol and expression are appended to repl, and the symbol is returned.

See also

[basic::to\\_rational](#)

[basic::to\\_polynomial](#)

References [GiNaC::subs\\_options::no\\_pattern](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.588 frac\_cancel()**

```
static ex GiNaC::frac_cancel (
 const ex & n,
 const ex & d) [static]
```

Fraction cancellation.

Parameters

|          |             |
|----------|-------------|
| <i>n</i> | numerator   |
| <i>d</i> | denominator |

Returns

cancelled fraction {n, d} as a list

References [\\_ex1](#), [\\_ex1](#), [\\_num1\\_p](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::ex::expand\(\)](#), [gcd\(\)](#), [get\\_first\\_symbol\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [is\\_negative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [multiply\\_lcm\(\)](#), [n](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::ex::unit\(\)](#), and [x](#).

Referenced by [GiNaC::add::normal\(\)](#), and [GiNaC::mul::normal\(\)](#).

**5.1.3.589 find\_common\_factor()**

```
static ex GiNaC::find_common_factor (
 const ex & e,
 ex & factor,
 exmap & repl) [static]
```

Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).

References [\\_ex0](#), [\\_ex1](#), [divide\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [gcd\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [pow\(\)](#), [GiNaC::ex::to\\_polynomial\(\)](#), and [x](#).

Referenced by [collect\\_common\\_factors\(\)](#), and [find\\_common\\_factor\(\)](#).

**5.1.3.590 collect\_common\_factors()**

```
ex GiNaC::collect_common_factors (
 const ex & e)
```

Collect common factors in sums.

This converts expressions like 'a\*(b\*x+b\*y)' to 'a\*b\*(x+y)'.

References [factor\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [r](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::power::to\\_polynomial\(\)](#).

**5.1.3.591 resultant()**

```
ex GiNaC::resultant (
 const ex & e1,
 const ex & e2,
 const ex & s)
```

Resultant of two expressions e1,e2 with respect to symbol s.

Method: Compute determinant of Sylvester matrix of e1,e2,s.

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [k](#), [GiNaC::ex::ldegree\(\)](#), [m](#), and [GiNaC::info\\_flags::polynomial](#).

**5.1.3.592 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [26/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 numeric ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_csrc_cl_N > &↵
::do_print_csrc_cl_N. print_func< print_tree > &::do_print_tree. print_func< print_python_repr
> &::do_print_python_repr)
```

default ctor.

Numerically it initializes to an integer zero.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::value](#).

**5.1.3.593 make\_real\_float()**

```
static const cln::cl_F GiNaC::make_real_float (
 const cln::cl_idcoded_float & dec) [static]
```

Construct a floating point number from sign, mantissa, and exponent.

References [x](#).

Referenced by [read\\_real\\_float\(\)](#).

#### 5.1.3.594 read\_real\_float()

```
static const cln::cl_F GiNaC::read_real_float (
 std::istream & s) [static]
```

Read serialized floating point number.

References [make\\_real\\_float\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::read\\_archive\(\)](#).

#### 5.1.3.595 GINAC\_BIND\_UNARCHIVER() [36/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 numeric)
```

#### 5.1.3.596 write\_real\_float()

```
static void GiNaC::write_real_float (
 std::ostream & s,
 const cln::cl_R & n) [static]
```

References [n](#).

Referenced by [GiNaC::numeric::archive\(\)](#).

#### 5.1.3.597 print\_real\_number()

```
static void GiNaC::print_real_number (
 const print_context & c,
 const cln::cl_R & x) [static]
```

Helper function to print a real number in a nicer way than is CLN's default.

Instead of printing 42.0L0 this just prints 42.0 to ostream os and instead of 3.99168L7 it prints 3.99168E7. This is fine in [GiNaC](#) as long as it only uses `cl_LF` and no other floating point types that we might want to visibly distinguish from `cl_LF`.

See also

[numeric::print\(\)](#)

References [c](#), and [x](#).

Referenced by [GiNaC::numeric::print\\_numeric\(\)](#), and [print\\_real\\_cl\\_N\(\)](#).

**5.1.3.598 print\_integer\_csrc()**

```
static void GiNaC::print_integer_csrc (
 const print_context & c,
 const cln::cl_I & x) [static]
```

Helper function to print integer number in C++ source format.

See also

[numeric::print\(\)](#)

References [c](#), and [x](#).

Referenced by [print\\_real\\_csrc\(\)](#).

**5.1.3.599 print\_real\_csrc()**

```
static void GiNaC::print_real_csrc (
 const print_context & c,
 const cln::cl_R & x) [static]
```

Helper function to print real number in C++ source format.

See also

[numeric::print\(\)](#)

References [c](#), [denom\(\)](#), [numer\(\)](#), [print\\_integer\\_csrc\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\(\)](#).

**5.1.3.600 coerce()**

```
template<typename T1 , typename T2 >
static bool GiNaC::coerce (
 T1 & dst,
 const T2 & arg) [inline], [static]
```

Referenced by [print\\_real\\_cl\\_N\(\)](#).

**5.1.3.601 coerce< int, cln::cl\_I >()**

```
template<>
bool GiNaC::coerce< int, cln::cl_I > (
 int & dst,
 const cln::cl_I & arg) [inline]
```

Check if CLN integer can be converted into int.

See also

<https://www.ginac.de/pipermail/cln-list/2006-October/000248.html>

**5.1.3.602 `coerce< unsigned int, cln::cl_I >()`**

```
template<>
bool GiNaC::coerce< unsigned int, cln::cl_I > (
 unsigned int & dst,
 const cln::cl_I & arg) [inline]
```

**5.1.3.603 `print_real_cl_N()`**

```
static void GiNaC::print_real_cl_N (
 const print_context & c,
 const cln::cl_R & x) [static]
```

Helper function to print real number in C++ source format using `cl_N` types.

See also

[numeric::print\(\)](#)

References [c](#), [coerce\(\)](#), [Digits](#), [print\\_real\\_number\(\)](#), and [x](#).

Referenced by [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#).

**5.1.3.604 `exp()`**

```
const numeric GiNaC::exp (
 const numeric & x)
```

Exponential function.

Returns

arbitrary precision numerical  $\exp(x)$ .

References [x](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [Bernoulli\\_polynomial\(\)](#), [beta\\_evalf\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [csgn\\_power\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [exp\\_conjugate\(\)](#), [exp\\_deriv\(\)](#), [exp\\_eval\(\)](#), [exp\\_evalf\(\)](#), [exp\\_expand\(\)](#), [exp\\_imag\\_part\(\)](#), [exp\\_power\(\)](#), [exp\\_real\\_part\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_val](#), [GiNaC::power::imag\\_part\(\)](#), [log\\_eval\(\)](#), [print\\_sym\\_pow\(\)](#), [GiNaC::power::real\\_part\(\)](#), [replace\\_with\\_symbol\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [tgamma\(\)](#).

**5.1.3.605 `log()`**

```
const numeric GiNaC::log (
 const numeric & x)
```

Natural logarithm.

## Parameters

|          |                |
|----------|----------------|
| <i>x</i> | complex number |
|----------|----------------|

## Returns

arbitrary precision numerical  $\log(x)$ .

## Exceptions

|                                               |
|-----------------------------------------------|
| <i>pole_error("log() logarithmic pole",0)</i> |
|-----------------------------------------------|

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [exp\\_eval\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_series\(\)](#), [Li2\\_deriv\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [log\\_conjugate\(\)](#), [log\\_eval\(\)](#), [log\\_evalf\(\)](#), [log\\_expand\(\)](#), [log\\_real\\_part\(\)](#), [log\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::power::real\\_part\(\)](#), [S\\_eval\(\)](#), and [GiNaC::power::series\(\)](#).

**5.1.3.606 sin()**

```
const numeric GiNaC::sin (
 const numeric & x)
```

Numeric sine (trigonometric function).

## Returns

arbitrary precision numerical  $\sin(x)$ .

References [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_deriv\(\)](#), [cos\\_imag\\_part\(\)](#), [cosh\\_imag\\_part\(\)](#), [exp\\_imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [lgamma\(\)](#), [sin\\_conjugate\(\)](#), [sin\\_eval\(\)](#), [sin\\_evalf\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tan\\_series\(\)](#).

**5.1.3.607 cos()**

```
const numeric GiNaC::cos (
 const numeric & x)
```

Numeric cosine (trigonometric function).

## Returns

arbitrary precision numerical  $\cos(x)$ .

References [x](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [cos\\_conjugate\(\)](#), [cos\\_eval\(\)](#), [cos\\_evalf\(\)](#), [cos\\_real\\_part\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_real\\_part\(\)](#), [exp\\_real\\_part\(\)](#), [GiNaC::power::real\\_part\(\)](#), [sin\\_deriv\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_real\\_part\(\)](#), and [tan\\_series\(\)](#).

### 5.1.3.608 `tan()`

```
const numeric GiNaC::tan (
 const numeric & x)
```

Numeric tangent (trigonometric function).

#### Returns

arbitrary precision numerical  $\tan(x)$ .

#### References [x](#).

Referenced by [tan\\_conjugate\(\)](#), [tan\\_deriv\(\)](#), [tan\\_eval\(\)](#), [tan\\_evalf\(\)](#), [tan\\_imag\\_part\(\)](#), [tan\\_real\\_part\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

### 5.1.3.609 `asin()`

```
const numeric GiNaC::asin (
 const numeric & x)
```

Numeric inverse sine (trigonometric function).

#### Returns

arbitrary precision numerical  $\arcsin(x)$ .

#### References [x](#).

Referenced by [asin\\_conjugate\(\)](#), [asin\\_eval\(\)](#), [asin\\_evalf\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

### 5.1.3.610 `acos()`

```
const numeric GiNaC::acos (
 const numeric & x)
```

Numeric inverse cosine (trigonometric function).

#### Returns

arbitrary precision numerical  $\arccos(x)$ .

#### References [x](#).

Referenced by [acos\\_conjugate\(\)](#), [acos\\_eval\(\)](#), [acos\\_evalf\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

### 5.1.3.611 `atan()` [1/2]

```
const numeric GiNaC::atan (
 const numeric & x)
```

Numeric arcustangent.

## Parameters

|          |                |
|----------|----------------|
| <i>x</i> | complex number |
|----------|----------------|

## Returns

$\operatorname{atan}(x)$

## Exceptions

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <i>pole_error("atan()"</i> | <i>logarithmic pole",0)</i> if $x==1$ or $x==-1$ . |
|----------------------------|----------------------------------------------------|

References [\\_num1\\_p](#), [abs\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

Referenced by [atan2\\_eval\(\)](#), [atan2\\_evalf\(\)](#), [atan\\_conjugate\(\)](#), [atan\\_eval\(\)](#), [atan\\_evalf\(\)](#), [atan\\_series\(\)](#), [cos\\_eval\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.612 atan()** [2/2]

```
const numeric GiNaC::atan (
 const numeric & y,
 const numeric & x)
```

Numeric arcustangent of two arguments, analytically continued in a suitable way.

## Parameters

|          |                |
|----------|----------------|
| <i>y</i> | complex number |
| <i>x</i> | complex number |

## Returns

$-i \log((x+iy)/\sqrt{x^2+y^2})$ , which is equal to  $\operatorname{atan}(y/x)$  if  $y$  and  $x$  are both real.

## Exceptions

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <i>pole_error("atan()"</i> | <i>logarithmic pole",0)</i> if $y/x==1$ or $y/x==-1$ . |
|----------------------------|--------------------------------------------------------|

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), and [x](#).

**5.1.3.613 sinh()**

```
const numeric GiNaC::sinh (
 const numeric & x)
```

Numeric hyperbolic sine (trigonometric function).

**Returns**

arbitrary precision numerical  $\sinh(x)$ .

**References** [x](#).

Referenced by [cos\\_imag\\_part\(\)](#), [cosh\\_deriv\(\)](#), [cosh\\_imag\\_part\(\)](#), [sin\\_imag\\_part\(\)](#), [sinh\\_conjugate\(\)](#), [sinh\\_eval\(\)](#), [sinh\\_evalf\(\)](#), [sinh\\_real\\_part\(\)](#), and [tanh\\_series\(\)](#).

**5.1.3.614 cosh()**

```
const numeric GiNaC::cosh (
 const numeric & x)
```

Numeric hyperbolic cosine (trigonometric function).

**Returns**

arbitrary precision numerical  $\cosh(x)$ .

**References** [x](#).

Referenced by [cos\\_real\\_part\(\)](#), [cosh\\_conjugate\(\)](#), [cosh\\_eval\(\)](#), [cosh\\_evalf\(\)](#), [cosh\\_real\\_part\(\)](#), [sin\\_real\\_part\(\)](#), [sinh\\_deriv\(\)](#), [sinh\\_imag\\_part\(\)](#), and [tanh\\_series\(\)](#).

**5.1.3.615 tanh()**

```
const numeric GiNaC::tanh (
 const numeric & x)
```

Numeric hyperbolic tangent (trigonometric function).

**Returns**

arbitrary precision numerical  $\tanh(x)$ .

**References** [x](#).

Referenced by [tan\\_imag\\_part\(\)](#), [tanh\\_conjugate\(\)](#), [tanh\\_deriv\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_evalf\(\)](#), [tanh\\_imag\\_part\(\)](#), and [tanh\\_real\\_part\(\)](#).

**5.1.3.616 asinh()**

```
const numeric GiNaC::asinh (
 const numeric & x)
```

Numeric inverse hyperbolic sine (trigonometric function).

**Returns**

arbitrary precision numerical  $\operatorname{asinh}(x)$ .

**References** [x](#).

Referenced by [asinh\\_conjugate\(\)](#), [asinh\\_eval\(\)](#), [asinh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

### 5.1.3.617 `acosh()`

```
const numeric GiNaC::acosh (
 const numeric & x)
```

Numeric inverse hyperbolic cosine (trigonometric function).

#### Returns

arbitrary precision numerical `acosh(x)`.

References [x](#).

Referenced by [acosh\\_conjugate\(\)](#), [acosh\\_eval\(\)](#), [acosh\\_evalf\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

### 5.1.3.618 `atanh()`

```
const numeric GiNaC::atanh (
 const numeric & x)
```

Numeric inverse hyperbolic tangent (trigonometric function).

#### Returns

arbitrary precision numerical `atanh(x)`.

References [x](#).

Referenced by [atanh\\_conjugate\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_evalf\(\)](#), [atanh\\_series\(\)](#), [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

### 5.1.3.619 `Li2_series()` [2/2]

```
static cln::cl_N GiNaC::Li2_series (
 const cln::cl_N & x,
 const cln::float_format_t & prec) [static]
```

Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.

References [x](#).

### 5.1.3.620 `Li2_projection()`

```
static cln::cl_N GiNaC::Li2_projection (
 const cln::cl_N & x,
 const cln::float_format_t & prec) [static]
```

Folds `Li2`'s argument inside a small rectangle to enhance convergence.

References [Li2\\_projection\(\)](#), [Li2\\_series\(\)](#), and [x](#).

Referenced by [Li2\\_\(\)](#), and [Li2\\_projection\(\)](#).

**5.1.3.621 Li2\_()**

```
const cln::cl_N GiNaC::Li2_ (
 const cln::cl_N & value)
```

Numeric evaluation of Dilogarithm.

The domain is the entire complex plane, the branch cut lies along the positive real axis, starting at 1 and continuous with quadrant IV.

**Returns**

arbitrary precision numerical Li2(x).

References [Li2\\_projection\(\)](#), and [value](#).

Referenced by [Li2\(\)](#).

**5.1.3.622 Li2()**

```
const numeric GiNaC::Li2 (
 const numeric & x)
```

References [\\_num0\\_p](#), [Li2\\_\(\)](#), and [x](#).

Referenced by [Li2\\_conjugate\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

**5.1.3.623 zeta() [3/3]**

```
const numeric GiNaC::zeta (
 const numeric & x)
```

Numeric evaluation of Riemann's Zeta function.

Currently works only for integer arguments.

References [x](#).

**5.1.3.624 guess\_precision()**

```
static cln::float_format_t GiNaC::guess_precision (
 const cln::cl_N & x) [static]
```

References [x](#).

Referenced by [lgamma\(\)](#), and [tgamma\(\)](#).

**5.1.3.625 lgamma()** [1/2]

```
const cln::cl_N GiNaC::lgamma (
 const cln::cl_N & x)
```

The Gamma function.

Use the Lanczos approximation. If the coefficients used here are not sufficiently many or sufficiently accurate, more can be calculated using the program `doc/examples/lanczos.cpp`. In that case, be sure to read the comments in that file.

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [lgamma\(\)](#), [log\(\)](#), [sin\(\)](#), [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#), and [x](#).

Referenced by [beta\\_evalf\(\)](#), [lgamma\(\)](#), [lgamma\(\)](#), [lgamma\\_conjugate\(\)](#), [lgamma\\_eval\(\)](#), [lgamma\\_evalf\(\)](#), and [lgamma\\_series\(\)](#).

**5.1.3.626 lgamma()** [2/2]

```
const numeric GiNaC::lgamma (
 const numeric & x)
```

References [lgamma\(\)](#), and [x](#).

**5.1.3.627 tgamma()** [1/2]

```
const cln::cl_N GiNaC::tgamma (
 const cln::cl_N & x)
```

References [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [exp\(\)](#), [GiNaC::lanczos\\_coeffs::get\\_order\(\)](#), [guess\\_precision\(\)](#), [sqrt\(\)](#), [GiNaC::lanczos\\_coeffs::sufficiently\\_accurate\(\)](#), [tgamma\(\)](#), and [x](#).

Referenced by [beta\\_eval\(\)](#), [beta\\_series\(\)](#), [psi2\\_eval\(\)](#), [tgamma\(\)](#), [tgamma\(\)](#), [tgamma\\_conjugate\(\)](#), [tgamma\\_deriv\(\)](#), [tgamma\\_eval\(\)](#), [tgamma\\_evalf\(\)](#), and [tgamma\\_series\(\)](#).

**5.1.3.628 tgamma()** [2/2]

```
const numeric GiNaC::tgamma (
 const numeric & x)
```

References [tgamma\(\)](#), and [x](#).

**5.1.3.629 psi()** [3/4]

```
const numeric GiNaC::psi (
 const numeric & x)
```

The psi function (aka polygamma function).

This is only a stub!

**5.1.3.630 psi()** [4/4]

```
const numeric GiNaC::psi (
 const numeric & n,
 const numeric & x)
```

The psi functions (aka polygamma functions).

This is only a stub!

**5.1.3.631 factorial()**

```
const numeric GiNaC::factorial (
 const numeric & n)
```

Factorial combinatorial function.

**Parameters**

|          |                           |
|----------|---------------------------|
| <i>n</i> | integer argument $\geq 0$ |
|----------|---------------------------|

**Exceptions**

|                    |                                      |
|--------------------|--------------------------------------|
| <i>range_error</i> | (argument must be integer $\geq 0$ ) |
|--------------------|--------------------------------------|

References [n](#).

Referenced by [Bernoulli\\_polynomial\(\)](#), [factorial\\_conjugate\(\)](#), [factorial\\_eval\(\)](#), [factorial\\_evalf\(\)](#), [factorial\\_real\\_part\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [multinomial\\_coefficient\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [S\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [symm\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.632 doublefactorial()**

```
const numeric GiNaC::doublefactorial (
 const numeric & n)
```

The double factorial combinatorial function.

(Scarcely used, but still useful in cases, like for exact results of  $tgamma(n+1/2)$  for instance.)

**Parameters**

|          |                            |
|----------|----------------------------|
| <i>n</i> | integer argument $\geq -1$ |
|----------|----------------------------|

**Returns**

$n!! == n * (n-2) * (n-4) * \dots * (\{1|2\})$  with  $0!! == (-1)!! == 1$

## Exceptions

|                          |                                       |
|--------------------------|---------------------------------------|
| <code>range_error</code> | (argument must be integer $\geq -1$ ) |
|--------------------------|---------------------------------------|

References [\\_num1\\_p](#), [\\_num\\_1\\_p](#), and [n](#).

Referenced by [tgamma\\_eval\(\)](#).

**5.1.3.633 binomial()**

```
const numeric GiNaC::binomial (
 const numeric & n,
 const numeric & k)
```

The Binomial coefficients.

It computes the binomial coefficients. For integer  $n$  and  $k$  and positive  $n$  this is the number of ways of choosing  $k$  objects from  $n$  distinct objects. If  $n$  is a negative integer, the formula  $\text{binomial}(n,k) == (-1)^k * \text{binomial}(k-n-1,k)$  (if  $k \geq 0$ )  $\text{binomial}(n,k) == (-1)^{(n-k)} * \text{binomial}(-k-1,n-k)$  (otherwise) is used to compute the result.

References [\\_num0\\_p](#), [\\_num1\\_p](#), [\\_num\\_1\\_p](#), [binomial\(\)](#), [k](#), [n](#), and [GiNaC::numeric::power\(\)](#).

Referenced by [binomial\(\)](#), [binomial\\_conjugate\(\)](#), [binomial\\_eval\(\)](#), [binomial\\_evalf\(\)](#), [binomial\\_real\\_part\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::imag\\_part\(\)](#), and [GiNaC::power::real\\_part\(\)](#).

**5.1.3.634 bernoulli()**

```
const numeric GiNaC::bernoulli (
 const numeric & nn)
```

Bernoulli number.

The  $n$ th Bernoulli number is the coefficient of  $x^n/n!$  in the expansion of the function  $x/(e^x-1)$ .

## Returns

the  $n$ th Bernoulli number (a rational number).

## Exceptions

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>range_error</code> | (argument must be integer $\geq 0$ ) |
|--------------------------|--------------------------------------|

References [\\_num1\\_p](#), [c](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [k](#), [n](#), and [GiNaC::numeric::to\\_int\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.635 fibonacci()**

```
const numeric GiNaC::fibonacci (
 const numeric & n)
```

Fibonacci number.

The nth Fibonacci number  $F(n)$  is defined by the recurrence formula  $F(n) = F(n-1) + F(n-2)$  with  $F(0) = 0$  and  $F(1) = 1$ .

#### Parameters

|     |            |
|-----|------------|
| $n$ | an integer |
|-----|------------|

#### Returns

the nth Fibonacci number  $F(n)$  (an integer number)

#### Exceptions

|                    |                               |
|--------------------|-------------------------------|
| <i>range_error</i> | (argument must be an integer) |
|--------------------|-------------------------------|

References [\\_num0\\_p](#), [fibonacci\(\)](#), [m](#), and [n](#).

Referenced by [fibonacci\(\)](#).

### 5.1.3.636 abs()

```
const numeric GiNaC::abs (
 const numeric & x)
```

Absolute value.

References [x](#).

Referenced by [abs\\_conjugate\(\)](#), [abs\\_eval\(\)](#), [abs\\_evalf\(\)](#), [abs\\_expand\(\)](#), [abs\\_expl\\_derivative\(\)](#), [abs\\_power\(\)](#), [abs\\_real\\_part\(\)](#), [adaptivesimpson\(\)](#), [atan\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [GiNaC::numeric::integer\\_content\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\(\)](#), [log\\_real\\_part\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::numeric::max\\_coefficient\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::power::real\\_part\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [zeta1\\_eval\(\)](#).

### 5.1.3.637 mod()

```
const numeric GiNaC::mod (
 const numeric & a,
 const numeric & b)
```

Modulus (in positive representation).

In general,  $\text{mod}(a,b)$  has the sign of  $b$  or is zero, and  $\text{rem}(a,b)$  has the sign of  $a$  or is zero. This is different from Maple's  $\text{modp}$ , where the sign of  $b$  is ignored. It is in agreement with Mathematica's  $\text{Mod}$ .

#### Returns

$a \bmod b$  in the range  $[0, \text{abs}(b)-1]$  with sign of  $b$  if both are integer, 0 otherwise.

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [cos\\_eval\(\)](#), [exp\\_eval\(\)](#), [is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

**5.1.3.638 smod()**

```
const numeric GiNaC::smod (
 const numeric & a_,
 const numeric & b_)
```

Modulus (in symmetric representation).

**Returns**

$a \bmod b$  in the range  $[-\text{iquo}(\text{abs}(b),2), \text{iquo}(\text{abs}(b),2)]$ .

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [m](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), and [GiNaC::numeric::smod\(\)](#).

**5.1.3.639 irem() [1/2]**

```
const numeric GiNaC::irem (
 const numeric & a,
 const numeric & b)
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b)` as far as sign conventions are concerned. In general, `mod(a,b)` has the sign of `b` or is zero, and `irem(a,b)` has the sign of `a` or is zero.

**Returns**

remainder of  $a/b$  if both are integer, 0 otherwise.

**Exceptions**

|                             |                                               |
|-----------------------------|-----------------------------------------------|
| <code>overflow_error</code> | (division by zero) if <code>b</code> is zero. |
|-----------------------------|-----------------------------------------------|

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), and [ifactor\(\)](#).

**5.1.3.640 irem() [2/2]**

```
const numeric GiNaC::irem (
 const numeric & a,
 const numeric & b,
 numeric & q)
```

Numeric integer remainder.

Equivalent to Maple's `irem(a,b,'q')` it obeys the relation `irem(a,b,q) == a - q*b`. In general, `mod(a,b)` has the sign of `b` or is zero, and `irem(a,b)` has the sign of `a` or is zero.

**Returns**

remainder of  $a/b$  and quotient stored in  $q$  if both are integer, 0 otherwise.

**Exceptions**

|                             |                                    |
|-----------------------------|------------------------------------|
| <code>overflow_error</code> | (division by zero) if $b$ is zero. |
|-----------------------------|------------------------------------|

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.641 iquo() [1/2]**

```
const numeric GiNaC::iquo (
 const numeric & a,
 const numeric & b)
```

Numeric integer quotient.

Equivalent to Maple's `iquo` as far as sign conventions are concerned.

**Returns**

truncated quotient of  $a/b$  if both are integer, 0 otherwise.

**Exceptions**

|                             |                                    |
|-----------------------------|------------------------------------|
| <code>overflow_error</code> | (division by zero) if $b$ is zero. |
|-----------------------------|------------------------------------|

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

Referenced by [GiNaC::power::eval\(\)](#), and [heur\\_gcd\\_z\(\)](#).

**5.1.3.642 iquo() [2/2]**

```
const numeric GiNaC::iquo (
 const numeric & a,
 const numeric & b,
 numeric & r)
```

Numeric integer quotient.

Equivalent to Maple's `iquo(a,b,'r')` it obeys the relation  $r == a - \text{iquo}(a,b,r)*b$ .

**Returns**

truncated quotient of  $a/b$  and remainder stored in  $r$  if both are integer, 0 otherwise.

## Exceptions

|                             |                                  |
|-----------------------------|----------------------------------|
| <code>overflow_error</code> | (division by zero) if b is zero. |
|-----------------------------|----------------------------------|

References [\\_num0\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [r](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.643 gcd()** [2/2]

```
const numeric GiNaC::gcd (
 const numeric & a,
 const numeric & b)
```

Greatest Common Divisor.

## Returns

The GCD of two numbers if both are integer, a numerical 1 if they are not.

References [\\_num1\\_p](#), [GiNaC::numeric::is\\_integer\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.644 lcm()** [2/2]

```
const numeric GiNaC::lcm (
 const numeric & a,
 const numeric & b)
```

Least Common Multiple.

## Returns

The LCM of two numbers if both are integer, the product of those two numbers if they are not.

References [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::mul\(\)](#), and [GiNaC::numeric::to\\_cl\\_N\(\)](#).

**5.1.3.645 sqrt()** [1/2]

```
const numeric GiNaC::sqrt (
 const numeric & x)
```

Numeric square root.

If possible, sqrt(x) should respect squares of exact numbers, i.e. sqrt(4) should return integer 2.

## Parameters

|                |                  |
|----------------|------------------|
| <code>x</code> | numeric argument |
|----------------|------------------|

**Returns**

square root of  $x$ . Branch cut along negative real axis, the negative real axis itself where  $\text{imag}(x)=0$  and  $\text{real}(x)<0$  belongs to the upper part where  $\text{imag}(x)>0$ .

**References** [x](#).

Referenced by [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [EllipticE\\_evalf\(\)](#), [EllipticK\\_evalf\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [tgamma\(\)](#), and [tgamma\\_eval\(\)](#).

**5.1.3.646 isqrt()**

```
const numeric GiNaC::isqrt (
 const numeric & x)
```

Integer numeric square root.

References [\\_num0\\_p](#), and [x](#).

Referenced by [heur\\_gcd\\_z\(\)](#).

**5.1.3.647 PiEvalf()**

```
ex GiNaC::PiEvalf ()
```

Floating point evaluation of Archimedes' constant Pi.

**5.1.3.648 EulerEvalf()**

```
ex GiNaC::EulerEvalf ()
```

Floating point evaluation of Euler's constant gamma.

**5.1.3.649 CatalanEvalf()**

```
ex GiNaC::CatalanEvalf ()
```

Floating point evaluation of Catalan's constant.

**5.1.3.650 operator<<() [6/16]**

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const _numeric_digits & e)
```

References [GiNaC::\\_numeric\\_digits::print\(\)](#).

**5.1.3.651 GINAC\_DECLARE\_UNARCHIVER()** [38/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 numeric)
```

**5.1.3.652 pow()** [1/3]

```
const numeric GiNaC::pow (
 const numeric & x,
 const numeric & y) [inline]
```

References [x](#).

Referenced by [abs\\_eval\(\)](#), [abs\\_power\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [beta\\_eval\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_and\\_derivative\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [factor\(\)](#), [find\\_common\\_factor\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [gcd\(\)](#), [gcd\\_pf\\_pow\(\)](#), [gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [H\\_eval\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [interpolate\(\)](#), [kronecker\\_symbol\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [lcmcoeff\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [Li\\_series\(\)](#), [log\\_series\(\)](#), [multiply\\_lcm\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::op\(\)](#), [Order\\_power\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [rem\(\)](#), [replace\\_with\\_symbol\(\)](#), [S\\_eval\(\)](#), [S\\_series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [sprem\(\)](#), [sqrtfree\\_parfrac\(\)](#), [sr\\_gcd\(\)](#), [GiNaC::power::subs\(\)](#), [tgamma\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::power::to\\_rational\(\)](#), and [zeta1\\_eval\(\)](#).

**5.1.3.653 inverse()** [3/3]

```
const numeric GiNaC::inverse (
 const numeric & x) [inline]
```

References [x](#).

**5.1.3.654 step()**

```
numeric GiNaC::step (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [abs\\_eval\(\)](#), [H\\_eval\(\)](#), [step\\_conjugate\(\)](#), [step\\_eval\(\)](#), [step\\_evalf\(\)](#), [step\\_real\\_part\(\)](#), and [step\\_series\(\)](#).

**5.1.3.655 csgn()**

```
int GiNaC::csgn (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [csgn\\_conjugate\(\)](#), [csgn\\_eval\(\)](#), [csgn\\_evalf\(\)](#), [csgn\\_power\(\)](#), [csgn\\_real\\_part\(\)](#), [csgn\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), and [log\\_series\(\)](#).

#### 5.1.3.656 `is_zero()` [2/2]

```
bool GiNaC::is_zero (
 const numeric & x) [inline]
```

References [GiNaC::ex::is\\_zero\(\)](#), and [x](#).

#### 5.1.3.657 `is_positive()`

```
bool GiNaC::is_positive (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#).

#### 5.1.3.658 `is_negative()`

```
bool GiNaC::is_negative (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::power::eval\(\)](#), and [frac\\_cancel\(\)](#).

#### 5.1.3.659 `is_integer()`

```
bool GiNaC::is_integer (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [binomial\\_eval\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::lddegree\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [GiNaC::power::series\(\)](#).

#### 5.1.3.660 `is_pos_integer()`

```
bool GiNaC::is_pos_integer (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

#### 5.1.3.661 `is_nonneg_integer()`

```
bool GiNaC::is_nonneg_integer (
 const numeric & x) [inline]
```

References [x](#).

### 5.1.3.662 is\_even()

```
bool GiNaC::is_even (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [abs\\_power\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 5.1.3.663 is\_odd()

```
bool GiNaC::is_odd (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [csgn\\_power\(\)](#).

### 5.1.3.664 is\_prime()

```
bool GiNaC::is_prime (
 const numeric & x) [inline]
```

References [x](#).

### 5.1.3.665 is\_rational()

```
bool GiNaC::is_rational (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [replace\\_with\\_symbol\(\)](#), and [tgamma\\_eval\(\)](#).

### 5.1.3.666 is\_real()

```
bool GiNaC::is_real (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [beta\\_eval\(\)](#), [fsolve\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), and [Li2\\_series\(\)](#).

### 5.1.3.667 is\_cinteger()

```
bool GiNaC::is_cinteger (
 const numeric & x) [inline]
```

References [x](#).

#### 5.1.3.668 is\_crational()

```
bool GiNaC::is_crational (
 const numeric & x) [inline]
```

References [x](#).

#### 5.1.3.669 to\_int()

```
int GiNaC::to_int (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::basic::operator\[\]\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [S\\_eval\(\)](#), [GiNaC::power::series\(\)](#), and [sqrfree\\_parfrac\(\)](#).

#### 5.1.3.670 to\_long()

```
long GiNaC::to_long (
 const numeric & x) [inline]
```

References [x](#).

#### 5.1.3.671 to\_double()

```
double GiNaC::to_double (
 const numeric & x) [inline]
```

References [x](#).

#### 5.1.3.672 real()

```
const numeric GiNaC::real (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [cosh\\_eval\(\)](#), [sinh\\_eval\(\)](#), and [tanh\\_eval\(\)](#).

#### 5.1.3.673 imag()

```
const numeric GiNaC::imag (
 const numeric & x) [inline]
```

References [x](#).

Referenced by [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [G2\\_eval\(\)](#), and [G2\\_evalf\(\)](#).

**5.1.3.674 numer()** [2/2]

```
const numeric GiNaC::numer (
 const numeric & x) [inline]
```

References [GiNaC::ex::numer\(\)](#), and [x](#).

**5.1.3.675 denom()** [2/2]

```
const numeric GiNaC::denom (
 const numeric & x) [inline]
```

References [GiNaC::ex::denom\(\)](#), and [x](#).

**5.1.3.676 exadd()**

```
static const ex GiNaC::exadd (
 const ex & lh,
 const ex & rh) [inline], [static]
```

Used internally by [operator+\(\)](#) to add two ex objects.

Referenced by [operator+\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator--\(\)](#), and [operator-=\(\)](#).

**5.1.3.677 exmul()**

```
static const ex GiNaC::exmul (
 const ex & lh,
 const ex & rh) [inline], [static]
```

Used internally by [operator\\*\(\)](#) to multiply two ex objects.

References [GiNaC::return\\_types::commutative](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [operator\\*\(\)](#), [operator\\*=\(\(\)\)](#), [operator/\(\)](#), and [operator/=\(\(\)\)](#).

**5.1.3.678 exminus()**

```
static const ex GiNaC::exminus (
 const ex & lh) [inline], [static]
```

Used internally by [operator-\(\)](#) and friends to change the sign of an argument.

References [\\_ex\\_1](#).

Referenced by [operator-\(\)](#), [operator-\(\)](#), and [operator-=\(\)](#).

**5.1.3.679 operator+()** [1/4]

```
const ex GiNaC::operator+ (
 const ex & lh,
 const ex & rh)
```

References [exadd\(\)](#).

**5.1.3.680 operator-()** [1/4]

```
const ex GiNaC::operator- (
 const ex & lh,
 const ex & rh)
```

References [exadd\(\)](#), and [exminus\(\)](#).

**5.1.3.681 operator\*()** [1/2]

```
const ex GiNaC::operator* (
 const ex & lh,
 const ex & rh)
```

References [exmul\(\)](#).

**5.1.3.682 operator/()** [1/2]

```
const ex GiNaC::operator/ (
 const ex & lh,
 const ex & rh)
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**5.1.3.683 operator+()** [2/4]

```
const numeric GiNaC::operator+ (
 const numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::add\(\)](#).

**5.1.3.684 operator-()** [2/4]

```
const numeric GiNaC::operator- (
 const numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::sub\(\)](#).

**5.1.3.685 operator\*()** [2/2]

```
const numeric GiNaC::operator* (
 const numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::mul\(\)](#).

**5.1.3.686 operator/()** [2/2]

```
const numeric GiNaC::operator/ (
 const numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::div\(\)](#).

**5.1.3.687 operator+=()** [1/2]

```
ex & GiNaC::operator+= (
 ex & lh,
 const ex & rh)
```

References [exadd\(\)](#).

**5.1.3.688 operator-=()** [1/2]

```
ex & GiNaC::operator-= (
 ex & lh,
 const ex & rh)
```

References [exadd\(\)](#), and [exminus\(\)](#).

**5.1.3.689 operator\*=()** [1/2]

```
ex & GiNaC::operator*= (
 ex & lh,
 const ex & rh)
```

References [exmul\(\)](#).

**5.1.3.690 operator/=()** [1/2]

```
ex & GiNaC::operator/= (
 ex & lh,
 const ex & rh)
```

References [\\_ex\\_1](#), and [exmul\(\)](#).

**5.1.3.691 operator+=()** [2/2]

```
numeric & GiNaC::operator+= (
 numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::add\(\)](#).

**5.1.3.692 operator-=()** [2/2]

```
numeric & GiNaC::operator-= (
 numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::sub\(\)](#).

**5.1.3.693 operator\*=()** [2/2]

```
numeric & GiNaC::operator*= (
 numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::mul\(\)](#).

**5.1.3.694 operator/=( )** [2/2]

```
numeric & GiNaC::operator/=(
 numeric & lh,
 const numeric & rh)
```

References [GiNaC::numeric::div\(\)](#).

**5.1.3.695 operator+()** [3/4]

```
const ex GiNaC::operator+ (
 const ex & lh)
```

**5.1.3.696 operator-()** [3/4]

```
const ex GiNaC::operator- (
 const ex & lh)
```

References [exminus\(\)](#).

**5.1.3.697 operator+()** [4/4]

```
const numeric GiNaC::operator+ (
 const numeric & lh)
```

**5.1.3.698 operator-() [4/4]**

```
const numeric GiNaC::operator- (
 const numeric & lh)
```

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::mul\(\)](#).

**5.1.3.699 operator++() [1/4]**

```
ex & GiNaC::operator++ (
 ex & rh)
```

Expression prefix increment.

Adds 1 and returns incremented ex.

References [\\_ex1](#), and [exadd\(\)](#).

**5.1.3.700 operator--() [1/4]**

```
ex & GiNaC::operator-- (
 ex & rh)
```

Expression prefix decrement.

Subtracts 1 and returns decremented ex.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**5.1.3.701 operator++() [2/4]**

```
const ex GiNaC::operator++ (
 ex & lh,
 int)
```

Expression postfix increment.

Returns the ex and leaves the original incremented by 1.

References [\\_ex1](#), and [exadd\(\)](#).

**5.1.3.702 operator--() [2/4]**

```
const ex GiNaC::operator-- (
 ex & lh,
 int)
```

Expression postfix decrement.

Returns the ex and leaves the original decremented by 1.

References [\\_ex\\_1](#), and [exadd\(\)](#).

**5.1.3.703 operator++()** [3/4]

```
numeric & GiNaC::operator++ (
 numeric & rh)
```

Numeric prefix increment.

Adds 1 and returns incremented number.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.704 operator--()** [3/4]

```
numeric & GiNaC::operator-- (
 numeric & rh)
```

Numeric prefix decrement.

Subtracts 1 and returns decremented number.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.705 operator++()** [4/4]

```
const numeric GiNaC::operator++ (
 numeric & lh,
 int)
```

Numeric postfix increment.

Returns the number and leaves the original incremented by 1.

References [\\_num1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.706 operator--()** [4/4]

```
const numeric GiNaC::operator-- (
 numeric & lh,
 int)
```

Numeric postfix decrement.

Returns the number and leaves the original decremented by 1.

References [\\_num\\_1\\_p](#), and [GiNaC::numeric::add\(\)](#).

**5.1.3.707 operator==()**

```
const relational GiNaC::operator== (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::equal](#).

#### 5.1.3.708 operator"!=()

```
const relational GiNaC::operator!= (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::not\\_equal](#).

#### 5.1.3.709 operator<()

```
const relational GiNaC::operator< (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::less](#).

#### 5.1.3.710 operator<=()

```
const relational GiNaC::operator<= (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::less\\_or\\_equal](#).

#### 5.1.3.711 operator>()

```
const relational GiNaC::operator> (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::greater](#).

#### 5.1.3.712 operator>=()

```
const relational GiNaC::operator>= (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::relational::greater\\_or\\_equal](#).

#### 5.1.3.713 my\_ios\_index()

```
static int GiNaC::my_ios_index () [static]
```

Referenced by [get\\_print\\_context\(\)](#), and [set\\_print\\_context\(\)](#).

#### 5.1.3.714 `my_ios_callback()`

```
static void GiNaC::my_ios_callback (
 std::ios_base::event ev,
 std::ios_base & s,
 int i) [static]
```

Referenced by [set\\_print\\_context\(\)](#).

#### 5.1.3.715 `get_print_context()`

```
static print_context * GiNaC::get_print_context (
 std::ios_base & s) [inline], [static]
```

References [my\\_ios\\_index\(\)](#).

Referenced by [get\\_print\\_options\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), and [set\\_print\\_options\(\)](#).

#### 5.1.3.716 `set_print_context()`

```
static void GiNaC::set_print_context (
 std::ios_base & s,
 const print_context & c) [static]
```

References [c](#), [callback\\_registered](#), [my\\_ios\\_callback\(\)](#), [my\\_ios\\_index\(\)](#), [options](#), and [GiNaC::print\\_context::options](#).

Referenced by [csrc\(\)](#), [csrc\\_cl\\_N\(\)](#), [csrc\\_double\(\)](#), [csrc\\_float\(\)](#), [dflt\(\)](#), [latex\(\)](#), [python\(\)](#), [python\\_repr\(\)](#), [set\\_print\\_options\(\)](#), and [tree\(\)](#).

#### 5.1.3.717 `get_print_options()`

```
static unsigned GiNaC::get_print_options (
 std::ios_base & s) [inline], [static]
```

References [get\\_print\\_context\(\)](#), and [GiNaC::print\\_context::options](#).

Referenced by [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

#### 5.1.3.718 `set_print_options()`

```
static void GiNaC::set_print_options (
 std::ostream & s,
 unsigned options) [static]
```

References [get\\_print\\_context\(\)](#), [options](#), [GiNaC::print\\_context::options](#), and [set\\_print\\_context\(\)](#).

Referenced by [dflt\(\)](#), [index\\_dimensions\(\)](#), and [no\\_index\\_dimensions\(\)](#).

**5.1.3.719 operator<<()** [7/16]

```
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const ex & e)
```

References [get\\_print\\_context\(\)](#), and [GiNaC::ex::print\(\)](#).

**5.1.3.720 operator>>()** [3/3]

```
std::istream & GiNaC::operator>> (
 std::istream & is,
 ex & e)
```

**5.1.3.721 dflt()**

```
std::ostream & GiNaC::dflt (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#), and [set\\_print\\_options\(\)](#).

**5.1.3.722 latex()**

```
std::ostream & GiNaC::latex (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.723 python()**

```
std::ostream & GiNaC::python (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.724 python\_repr()**

```
std::ostream & GiNaC::python_repr (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.725 tree()**

```
std::ostream & GiNaC::tree (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#).

**5.1.3.726 csrc()**

```
std::ostream & GiNaC::csrc (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.727 csrc\_float()**

```
std::ostream & GiNaC::csrc_float (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.728 csrc\_double()**

```
std::ostream & GiNaC::csrc_double (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.729 csrc\_cl\_N()**

```
std::ostream & GiNaC::csrc_cl_N (
 std::ostream & os)
```

References [set\\_print\\_context\(\)](#).

**5.1.3.730 index\_dimensions()**

```
std::ostream & GiNaC::index_dimensions (
 std::ostream & os)
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.731 no\_index\_dimensions()**

```
std::ostream & GiNaC::no_index_dimensions (
 std::ostream & os)
```

References [get\\_print\\_options\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [set\\_print\\_options\(\)](#).

**5.1.3.732 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT() [27/33]**

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 power ,
 basic ,
 print_func< print_dflt > &::do_print_dflt. print_func< print_latex > &::do_↔
_print_latex. print_func< print_csrc > &::do_print_csrc. print_func< print_python > &↔
::do_print_python. print_func< print_python_repr > &::do_print_python_repr. print_func<
print_csrc_cl_N > &::do_print_csrc_cl_N)
```

**5.1.3.733 print\_sym\_pow()**

```
static void GiNaC::print_sym_pow (
 const print_context & c,
 const symbol & x,
 int exp) [static]
```

References [c](#), [exp\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_sym\\_pow\(\)](#), and [x](#).

Referenced by [GiNaC::power::do\\_print\\_csrc\(\)](#), and [print\\_sym\\_pow\(\)](#).

**5.1.3.734 GINAC\_BIND\_UNARCHIVER() [37/49]**

```
GiNaC::GINAC_BIND_UNARCHIVER (
 power)
```

**5.1.3.735 GINAC\_DECLARE\_UNARCHIVER() [39/51]**

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 power)
```

**5.1.3.736 pow() [2/3]**

```
ex GiNaC::pow (
 const ex & b,
 const ex & e) [inline]
```

Symbolic exponentiation.

Returns a power-object as a new expression.

**Parameters**

|          |                         |
|----------|-------------------------|
| <i>b</i> | the basis expression    |
| <i>e</i> | the exponent expression |

**5.1.3.737 pow() [3/3]**

```
template<typename T1 , typename T2 >
ex GiNaC::pow (
 const T1 & b,
 const T2 & e) [inline]
```

**5.1.3.738 sqrt() [2/2]**

```
ex GiNaC::sqrt (
 const ex & a) [inline]
```

Square root expression.

Returns a power-object with exponent 1/2.

References [\\_ex1\\_2](#).

#### 5.1.3.739 `is_a()` [3/3]

```
template<class T >
bool GiNaC::is_a (
 const print_context & obj) [inline]
```

Check if obj is a T, including base classes.

#### 5.1.3.740 `GINAC_IMPLEMENT_REGISTERED_CLASS_OPT()` [28/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 pseries ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
 print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python > &::do_↵
 print_python. print_func< print_python_repr > &::do_print_python_repr)
```

#### 5.1.3.741 `GINAC_BIND_UNARCHIVER()` [38/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 pseries)
```

#### 5.1.3.742 `GINAC_DECLARE_UNARCHIVER()` [40/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 pseries)
```

#### 5.1.3.743 `series_to_poly()`

```
ex GiNaC::series_to_poly (
 const ex & e) [inline]
```

Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.

The result is undefined if the expression does not contain a pseries object at its top level.

##### Parameters

|          |            |
|----------|------------|
| <i>e</i> | expression |
|----------|------------|

## Returns

polynomial expression

## See also

[is\\_a<>](#)

[pseries::convert\\_to\\_poly](#)

Referenced by [Bernoulli\\_polynomial\(\)](#), [generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), and [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#).

**5.1.3.744 is\_terminating()**

```
bool GiNaC::is_terminating (
 const pseries & s) [inline]
```

References [GiNaC::pseries::is\\_terminating\(\)](#).

**5.1.3.745 make\_return\_type\_t()**

```
template<typename T >
return_type_t GiNaC::make_return_type_t (
 const unsigned rl = 0) [inline]
```

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**5.1.3.746 set\_print\_func() [1/2]**

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
 void fconst T &, const C &c, unsigned) [extern]
```

Add or replace a print method.

References [GiNaC::class\\_info< OPT >::options](#), and [options](#).

**5.1.3.747 set\_print\_func() [2/2]**

```
template<class Alg , class Ctx , class T , class C >
void GiNaC::set_print_func (
 void(T::*)(const C &, unsigned) f) [extern]
```

Add or replace a print method.

References [options](#).

**5.1.3.748 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [29/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 relational ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print↵
 _tree. print_func< print_python_repr > &::do_print_python_repr)
```

**5.1.3.749 GINAC\_BIND\_UNARCHIVER()** [39/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 relational)
```

**5.1.3.750 print\_operator()**

```
static void GiNaC::print_operator (
 const print_context & c,
 relational::operators o) [static]
```

References [c](#), [GiNaC::relational::equal](#), [GiNaC::relational::greater](#), [GiNaC::relational::greater\\_or\\_equal](#), [GiNaC::relational::less](#), [GiNaC::relational::less\\_or\\_equal](#), and [GiNaC::relational::not\\_equal](#).

Referenced by [GiNaC::relational::do\\_print\(\)](#), and [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#).

**5.1.3.751 GINAC\_DECLARE\_UNARCHIVER()** [41/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 relational)
```

**5.1.3.752 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [30/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 symbol ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_latex > &::do_↵
 print_latex. print_func< print_tree > &::do_print_tree. print_func< print_python_repr > &↵
 ::do_print_python_repr)
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.753 get\_default\_TeX\_name()**

```
static const std::string & GiNaC::get_default_TeX_name (
 const std::string & name) [static]
```

Return default TeX name for symbol.

This recognizes some greek letters.

Referenced by [GiNaC::symbol::do\\_print\\_latex\(\)](#), and [GiNaC::symbol::get\\_TeX\\_name\(\)](#).

**5.1.3.754 GINAC\_BIND\_UNARCHIVER()** [40/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 symbol)
```

**5.1.3.755 GINAC\_BIND\_UNARCHIVER()** [41/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 realsymbol)
```

**5.1.3.756 GINAC\_BIND\_UNARCHIVER()** [42/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 possymbol)
```

**5.1.3.757 GINAC\_DECLARE\_UNARCHIVER()** [42/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 symbol)
```

**5.1.3.758 GINAC\_DECLARE\_UNARCHIVER()** [43/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 realsymbol)
```

**5.1.3.759 GINAC\_DECLARE\_UNARCHIVER()** [44/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 possymbol)
```

**5.1.3.760 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [31/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 symmetry ,
 basic ,
 print_func< print_context > &::do_print, print_func< print_tree > &::do_print↵
 _tree)
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.761 GINAC\_BIND\_UNARCHIVER()** [43/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 symmetry)
```

**5.1.3.762 index0()**

```
static const symmetry & GiNaC::index0 () [static]
```

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.763 index1()**

```
static const symmetry & GiNaC::index1 () [static]
```

Referenced by [antisymmetric2\(\)](#), [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric2\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.764 index2()**

```
static const symmetry & GiNaC::index2 () [static]
```

Referenced by [antisymmetric3\(\)](#), [antisymmetric4\(\)](#), [symmetric3\(\)](#), and [symmetric4\(\)](#).

**5.1.3.765 index3()**

```
static const symmetry & GiNaC::index3 () [static]
```

Referenced by [antisymmetric4\(\)](#), and [symmetric4\(\)](#).

**5.1.3.766 not\_symmetric()**

```
const symmetry & GiNaC::not_symmetric ()
```

Referenced by [GiNaC::indexed::indexed\(\)](#), and [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.767 symmetric2()**

```
const symmetry & GiNaC::symmetric2 ()
```

References [index0\(\)](#), [index1\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [delta\\_tensor\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [lorentz\\_g\(\)](#), and [metric\\_tensor\(\)](#).

**5.1.3.768 symmetric3()**

```
const symmetry & GiNaC::symmetric3 ()
```

References [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [color\\_d\(\)](#).

**5.1.3.769 symmetric4()**

```
const symmetry & GiNaC::symmetric4 ()
```

References [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), [index3\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.770 antisymmetric2()**

```
const symmetry & GiNaC::antisymmetric2 ()
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), and [index1\(\)](#).

Referenced by [epsilon\\_tensor\(\)](#), and [spinor\\_metric\(\)](#).

**5.1.3.771 antisymmetric3()**

```
const symmetry & GiNaC::antisymmetric3 ()
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), [index1\(\)](#), and [index2\(\)](#).

Referenced by [color\\_f\(\)](#), and [epsilon\\_tensor\(\)](#).

**5.1.3.772 antisymmetric4()**

```
const symmetry & GiNaC::antisymmetric4 ()
```

References [GiNaC::symmetry::antisymmetric](#), [index0\(\)](#), [index1\(\)](#), [index2\(\)](#), and [index3\(\)](#).

Referenced by [lorentz\\_eps\(\)](#).

**5.1.3.773 canonicalize()**

```
int GiNaC::canonicalize (
 exvector::iterator v,
 const symmetry & symm)
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

**Parameters**

|             |                            |
|-------------|----------------------------|
| <i>v</i>    | Start of expression vector |
| <i>symm</i> | Root node of symmetry tree |

**Returns**

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

Referenced by [GiNaC::function::eval\(\)](#), and [GiNaC::indexed::eval\(\)](#).

**5.1.3.774 symm()**

```
static ex GiNaC::symm (
 const ex & e,
 exvector::const_iterator first,
 exvector::const_iterator last,
 bool asymmetric) [static]
```

References [GiNaC::container< C >::append\(\)](#), [factorial\(\)](#), [last](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [permutation\\_sign\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [symmetrize\(\)](#), and [GiNaC::ex::symmetrize\(\)](#).

**5.1.3.775 symmetrize()** [3/4]

```
ex GiNaC::symmetrize (
 const ex & e,
 exvector::const_iterator first,
 exvector::const_iterator last)
```

Symmetrize expression over a set of objects (symbols, indices).

References [last](#), and [symm\(\)](#).

**5.1.3.776 antisymmetrize()** [3/4]

```
ex GiNaC::antisymmetrize (
 const ex & e,
 exvector::const_iterator first,
 exvector::const_iterator last)
```

Antisymmetrize expression over a set of objects (symbols, indices).

References [last](#), and [symm\(\)](#).

**5.1.3.777 symmetrize\_cyclic()** [3/4]

```
ex GiNaC::symmetrize_cyclic (
 const ex & e,
 exvector::const_iterator first,
 exvector::const_iterator last)
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::container< C >::append\(\)](#), [last](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::container< C >::remove\\_first\(\)](#), and [GiNaC::ex::subs\(\)](#).

**5.1.3.778 GINAC\_DECLARE\_UNARCHIVER()** [45/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 symmetry)
```

**5.1.3.779 sy\_none() [1/4]**

```
symmetry GiNaC::sy_none () [inline]
```

**5.1.3.780 sy\_none() [2/4]**

```
symmetry GiNaC::sy_none (
 const symmetry & c1,
 const symmetry & c2) [inline]
```

References [GiNaC::symmetry::none](#).

**5.1.3.781 sy\_none() [3/4]**

```
symmetry GiNaC::sy_none (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).

**5.1.3.782 sy\_none() [4/4]**

```
symmetry GiNaC::sy_none (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3,
 const symmetry & c4) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::none](#).

**5.1.3.783 sy\_symm() [1/4]**

```
symmetry GiNaC::sy_symm () [inline]
```

References [GiNaC::symmetry::set\\_type\(\)](#), and [GiNaC::symmetry::symmetric](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.784 sy\_symm() [2/4]**

```
symmetry GiNaC::sy_symm (
 const symmetry & c1,
 const symmetry & c2) [inline]
```

References [GiNaC::symmetry::symmetric](#).

**5.1.3.785 sy\_symm() [3/4]**

```

symmetry GiNaC::sy_symm (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.786 sy\_symm() [4/4]**

```

symmetry GiNaC::sy_symm (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3,
 const symmetry & c4) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::symmetric](#).

**5.1.3.787 sy\_anti() [1/4]**

```

symmetry GiNaC::sy_anti () [inline]

```

References [GiNaC::symmetry::antisymmetric](#), and [GiNaC::symmetry::set\\_type\(\)](#).

Referenced by [GiNaC::indexed::read\\_archive\(\)](#).

**5.1.3.788 sy\_anti() [2/4]**

```

symmetry GiNaC::sy_anti (
 const symmetry & c1,
 const symmetry & c2) [inline]

```

References [GiNaC::symmetry::antisymmetric](#).

**5.1.3.789 sy\_anti() [3/4]**

```

symmetry GiNaC::sy_anti (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**5.1.3.790 sy\_anti() [4/4]**

```

symmetry GiNaC::sy_anti (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3,
 const symmetry & c4) [inline]

```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::antisymmetric](#).

**5.1.3.791 sy\_cycl()** [1/4]

```
symmetry GiNaC::sy_cycl () [inline]
```

References [GiNaC::symmetry::cyclic](#), and [GiNaC::symmetry::set\\_type\(\)](#).

**5.1.3.792 sy\_cycl()** [2/4]

```
symmetry GiNaC::sy_cycl (
 const symmetry & c1,
 const symmetry & c2) [inline]
```

References [GiNaC::symmetry::cyclic](#).

**5.1.3.793 sy\_cycl()** [3/4]

```
symmetry GiNaC::sy_cycl (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**5.1.3.794 sy\_cycl()** [4/4]

```
symmetry GiNaC::sy_cycl (
 const symmetry & c1,
 const symmetry & c2,
 const symmetry & c3,
 const symmetry & c4) [inline]
```

References [GiNaC::symmetry::add\(\)](#), and [GiNaC::symmetry::cyclic](#).

**5.1.3.795 symmetrize()** [4/4]

```
ex GiNaC::symmetrize (
 const ex & e,
 const exvector & v) [inline]
```

Symmetrize expression over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**5.1.3.796 antisymmetrize()** [4/4]

```
ex GiNaC::antisymmetrize (
 const ex & e,
 const exvector & v) [inline]
```

Antisymmetrize expression over a set of objects (symbols, indices).

References [antisymmetrize\(\)](#), and [GiNaC::ex::begin\(\)](#).

**5.1.3.797 symmetrize\_cyclic()** [4/4]

```
ex GiNaC::symmetrize_cyclic (
 const ex & e,
 const exvector & v) [inline]
```

Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).

References [GiNaC::ex::begin\(\)](#), and [symmetrize\(\)](#).

**5.1.3.798 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [32/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 tensdelta ,
 tensor ,
 print_func< print_dflt > &::do_print. print_func< print_latex > &::do_print_↵
 latex)
```

**5.1.3.799 print\_func< print\_dflt >()** [3/3]

```
GiNaC::print_func< print_dflt > (
 &tensmetric::do_print) &
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.800 GINAC\_BIND\_UNARCHIVER()** [44/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 minkmetric)
```

**5.1.3.801 GINAC\_BIND\_UNARCHIVER()** [45/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 tensepsilon)
```

**5.1.3.802 GINAC\_BIND\_UNARCHIVER()** [46/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 tensdelta)
```

**5.1.3.803 GINAC\_BIND\_UNARCHIVER()** [47/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 tensmetric)
```

**5.1.3.804 GINAC\_BIND\_UNARCHIVER()** [48/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 spinmetric)
```

**5.1.3.805 delta\_tensor()**

```
ex GiNaC::delta_tensor (
 const ex & i1,
 const ex & i2)
```

Create a delta tensor with specified indices.

The indices must be of class `idx` or a subclass. The delta tensor is always symmetric and its trace is the dimension of the index space.

**Parameters**

|           |              |
|-----------|--------------|
| <i>i1</i> | First index  |
| <i>i2</i> | Second index |

**Returns**

newly constructed delta tensor

References [symmetric2\(\)](#).

Referenced by [color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**5.1.3.806 metric\_tensor()**

```
ex GiNaC::metric_tensor (
 const ex & i1,
 const ex & i2)
```

Create a symmetric metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. A metric tensor with one covariant and one contravariant index is equivalent to the delta tensor.

**Parameters**

|           |              |
|-----------|--------------|
| <i>i1</i> | First index  |
| <i>i2</i> | Second index |

**Returns**

newly constructed metric tensor

References [symmetric2\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#).

### 5.1.3.807 lorentz\_g()

```
ex GiNaC::lorentz_g (
 const ex & i1,
 const ex & i2,
 bool pos_sig = false)
```

Create a Minkowski metric tensor with specified indices.

The indices must be of class `varidx` or a subclass. The Lorentz metric is a symmetric tensor with a matrix representation of  $\text{diag}(1, -1, -1, \dots)$  (negative signature, the default) or  $\text{diag}(-1, 1, 1, \dots)$  (positive signature).

#### Parameters

|                |                                   |
|----------------|-----------------------------------|
| <i>i1</i>      | First index                       |
| <i>i2</i>      | Second index                      |
| <i>pos_sig</i> | Whether the signature is positive |

#### Returns

newly constructed Lorentz metric tensor

References [symmetric2\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#).

### 5.1.3.808 spinor\_metric()

```
ex GiNaC::spinor_metric (
 const ex & i1,
 const ex & i2)
```

Create a spinor metric tensor with specified indices.

The indices must be of class `spinidx` or a subclass and have a dimension of 2. The spinor metric is an antisymmetric tensor with a matrix representation of  $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ .

#### Parameters

|           |              |
|-----------|--------------|
| <i>i1</i> | First index  |
| <i>i2</i> | Second index |

#### Returns

newly constructed spinor metric tensor

References [antisymmetric2\(\)](#).

**5.1.3.809 epsilon\_tensor()** [1/2]

```
ex GiNaC::epsilon_tensor (
 const ex & i1,
 const ex & i2)
```

Create an epsilon tensor in a Euclidean space with two indices.

The indices must be of class `idx` or a subclass, and have a dimension of 2.

**Parameters**

|           |              |
|-----------|--------------|
| <i>i1</i> | First index  |
| <i>i2</i> | Second index |

**Returns**

newly constructed epsilon tensor

References [\\_ex2](#), [antisymmetric2\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**5.1.3.810 epsilon\_tensor()** [2/2]

```
ex GiNaC::epsilon_tensor (
 const ex & i1,
 const ex & i2,
 const ex & i3)
```

Create an epsilon tensor in a Euclidean space with three indices.

The indices must be of class `idx` or a subclass, and have a dimension of 3.

**Parameters**

|           |              |
|-----------|--------------|
| <i>i1</i> | First index  |
| <i>i2</i> | Second index |
| <i>i3</i> | Third index  |

**Returns**

newly constructed epsilon tensor

References [\\_ex3](#), [antisymmetric3\(\)](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

**5.1.3.811 lorentz\_eps()**

```
ex GiNaC::lorentz_eps (
 const ex & i1,
 const ex & i2,
```

```

const ex & i3,
const ex & i4,
bool pos_sig = false)

```

Create an epsilon tensor in a Minkowski space with four indices.

The indices must be of class `varidx` or a subclass, and have a dimension of 4.

#### Parameters

|                |                                                 |
|----------------|-------------------------------------------------|
| <i>i1</i>      | First index                                     |
| <i>i2</i>      | Second index                                    |
| <i>i3</i>      | Third index                                     |
| <i>i4</i>      | Fourth index                                    |
| <i>pos_sig</i> | Whether the signature of the metric is positive |

#### Returns

newly constructed epsilon tensor

References [\\_ex4](#), [antisymmetric4\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

#### 5.1.3.812 GINAC\_DECLARE\_UNARCHIVER() [46/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
 tensdelta)

```

#### 5.1.3.813 GINAC\_DECLARE\_UNARCHIVER() [47/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
 tensmetric)

```

#### 5.1.3.814 GINAC\_DECLARE\_UNARCHIVER() [48/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
 minkmetric)

```

#### 5.1.3.815 GINAC\_DECLARE\_UNARCHIVER() [49/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
 spinmetric)

```

#### 5.1.3.816 GINAC\_DECLARE\_UNARCHIVER() [50/51]

```

GiNaC::GINAC_DECLARE_UNARCHIVER (
 tensepsilon)

```

**5.1.3.817 log2()**

```
unsigned GiNaC::log2 (
 unsigned n)
```

Integer binary logarithm.

References [k](#), and [n](#).

Referenced by [GiNaC::remember\\_table::remember\\_table\(\)](#).

**5.1.3.818 multinomial\_coefficient()**

```
const numeric GiNaC::multinomial_coefficient (
 const std::vector< unsigned > & p)
```

Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.

$p$  is a partition of  $n$ .

References [GiNaC::numeric::add\(\)](#), [GiNaC::numeric::div\(\)](#), [factorial\(\)](#), [GiNaC::numeric::mul\(\)](#), and [n](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

**5.1.3.819 rotate\_left()**

```
unsigned GiNaC::rotate_left (
 unsigned n) [inline]
```

Rotate bits of unsigned value by one bit to the left.

This can be necessary if the user wants to define its own hashes.

References [n](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), and [GiNaC::symmetry::calchash\(\)](#).

**5.1.3.820 compare\_pointers()**

```
template<class T >
int GiNaC::compare_pointers (
 const T * a,
 const T * b) [inline]
```

Compare two pointers (just to establish some sort of canonical order).

**Returns**

-1, 0, or 1

Referenced by [GiNaC::basic::compare\\_same\\_type\(\)](#).

**5.1.3.821 golden\_ratio\_hash()**

```
unsigned GiNaC::golden_ratio_hash (
 uintptr_t n) [inline]
```

Truncated multiplication with golden ratio, for computing hash values.

References [n](#).

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), and [make\\_hash\\_seed\(\)](#).

**5.1.3.822 permutation\_sign() [1/2]**

```
template<class It >
int GiNaC::permutation_sign (
 It first,
 It last)
```

References [last](#), [swap\(\)](#), and [std::swap\(\)](#).

Referenced by [GiNaC::matrix::determinant\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), and [symm\(\)](#).

**5.1.3.823 permutation\_sign() [2/2]**

```
template<class It , class Cmp , class Swap >
int GiNaC::permutation_sign (
 It first,
 It last,
 Cmp comp,
 Swap swapit)
```

References [last](#).

**5.1.3.824 shaker\_sort()**

```
template<class It , class Cmp , class Swap >
void GiNaC::shaker_sort (
 It first,
 It last,
 Cmp comp,
 Swap swapit)
```

References [last](#).

Referenced by [find\\_free\\_and\\_dummy\(\)](#), and [rename\\_dummy\\_indices\(\)](#).

**5.1.3.825 cyclic\_permutation()**

```
template<class It , class Swap >
void GiNaC::cyclic_permutation (
 It first,
 It last,
 It new_first,
 Swap swapit)
```

References [last](#).

**5.1.3.826 format\_index\_value()** [1/2]

```
template<typename T >
std::enable_if< has_distance< T >::value, typename std::iterator_traits< T >::difference_type
>::type GiNaC::format_index_value (
 const T & a,
 const T & b)
```

For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.

However, we may print the difference to the starting point.

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

**5.1.3.827 format\_index\_value()** [2/2]

```
template<typename T >
std::enable_if<!has_distance< T >::value, T >::type GiNaC::format_index_value (
 const T & a,
 const T & b)
```

For all other cases we simply print the value.

**5.1.3.828 operator<<()** [8/16]

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const basic_multi_iterator< T > & v) [inline]
```

Output operator.

A multi\_iterator prints out as [basic\\_multi\\_iterator](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.829 operator<<() [9/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_ordered< T > & v) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered](#) prints out as [multi\\_iterator\\_ordered](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.830 operator<<() [10/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_ordered_eq< T > & v) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered\\_eq](#) prints out as [multi\\_iterator\\_ordered\\_eq](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.831 operator<<() [11/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_ordered_eq_indv< T > & v) [inline]
```

Output operator.

A [multi\\_iterator\\_ordered\\_eq\\_indv](#) prints out as [multi\\_iterator\\_ordered\\_eq\\_indv](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.832 operator<<() [12/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_counter< T > & v) [inline]
```

Output operator.

A [multi\\_iterator\\_counter](#) prints out as [multi\\_iterator\\_counter](#)(  $n_0, n_1, \dots$  ).

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.833 operator<<() [13/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_counter_indv< T > & v) [inline]
```

Output operator.

A `multi_iterator_counter_indv` prints out as `multi_iterator_counter_indv(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.834 operator<<() [14/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_permutation< T > & v) [inline]
```

Output operator.

A `multi_iterator_permutation` prints out as `multi_iterator_permutation(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.835 operator<<() [15/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_shuffle< T > & v) [inline]
```

Output operator.

A `multi_iterator_shuffle` prints out as `multi_iterator_shuffle(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.836 operator<<() [16/16]**

```
template<class T >
std::ostream & GiNaC::operator<< (
 std::ostream & os,
 const multi_iterator_shuffle_prime< T > & v) [inline]
```

Output operator.

A `multi_iterator_shuffle_prime` prints out as `multi_iterator_shuffle_prime(  $n_0, n_1, \dots$  )`.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [format\\_index\\_value\(\)](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

**5.1.3.837 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT()** [33/33]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
 wildcard ,
 basic ,
 print_func< print_context > &::do_print. print_func< print_tree > &::do_print←
_tree. print_func< print_python_repr > &::do_print_python_repr)
```

References [GiNaC::status\\_flags::evaluated](#), and [GiNaC::status\\_flags::expanded](#).

**5.1.3.838 GINAC\_BIND\_UNARCHIVER()** [49/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
 wildcard)
```

**5.1.3.839 haswild()**

```
bool GiNaC::haswild (
 const ex & x)
```

Check whether x has a wildcard anywhere as a subexpression.

References [haswild\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [x](#).

Referenced by [GiNaC::integral::eval\(\)](#), and [haswild\(\)](#).

**5.1.3.840 GINAC\_DECLARE\_UNARCHIVER()** [51/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
 wildcard)
```

**5.1.3.841 wild()**

```
ex GiNaC::wild (
 unsigned label = 0) [inline]
```

Create a wildcard object with the specified label.

**5.1.4 Variable Documentation****5.1.4.1 unarch\_table\_instance**

```
unarchive_table_t GiNaC::unarch_table_instance [static]
```

**5.1.4.2 map\_evalm**

```
GiNaC::evalm_map_function GiNaC::map_evalm
```

Referenced by [GiNaC::basic::evalm\(\)](#).

#### 5.1.4.3 map\_eval\_integ

`GiNaC::eval_integ_map_function` `GiNaC::map_eval_integ`

Referenced by `GiNaC::basic::eval_integ()`.

#### 5.1.4.4 tensor

`GiNaC::tensor`

#### 5.1.4.5 Pi

```
const constant GiNaC::Pi (
 "Pi" ,
 PiEvalf ,
 "\\pi" ,
 domain::positive)
```

Pi.

(3.14159...) Diverts straight into CLN for `evalf()`.

Referenced by `acos_eval()`, `acosh_eval()`, `asin_eval()`, `atan2_eval()`, `atan_eval()`, `atan_series()`, `atanh_series()`, `GiNaC::Eisenstein_h_kernel::coefficient_a0()`, `GiNaC::Eisenstein_h_kernel::coefficient_an()`, `cos_eval()`, `cosh_eval()`, `EllipticE_eval()`, `EllipticE_evalf()`, `EllipticE_series()`, `EllipticK_eval()`, `EllipticK_evalf()`, `EllipticK_series()`, `eta_eval()`, `eta_evalf()`, `exp_eval()`, `GiNaC::Kronecker_dtau_kernel::get_numerical_value()`, `GiNaC::Kronecker_dz_kernel::get_numerical_value()`, `H_evalf()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `log_eval()`, `log_series()`, `GiNaC::constant::read_archive()`, `GiNaC::Kronecker_dtau_kernel::series_coeff_impl()`, `sin_eval()`, `sinh_eval()`, `tan_eval()`, `tan_series()`, `tanh_eval()`, `tanh_series()`, `tgamma_eval()`, and `zeta1_eval()`.

#### 5.1.4.6 Euler

```
const constant GiNaC::Euler (
 "Euler" ,
 EulerEvalf ,
 "\\gamma_E" ,
 domain::positive)
```

Euler's constant.

(0.57721...) Sometimes called Euler-Mascheroni constant. Diverts straight into CLN for `evalf()`.

Referenced by `psi1_eval()`, and `GiNaC::constant::read_archive()`.

#### 5.1.4.7 Catalan

```
const constant GiNaC::Catalan (
 "Catalan" ,
 CatalanEvalf ,
 "G" ,
 domain::positive)
```

Catalan's constant.

(0.91597...) Diverts straight into CLN for `evalf()`.

Referenced by `Li2_eval()`, `Li_eval()`, and `GiNaC::constant::read_archive()`.

#### 5.1.4.8 crctab

```
unsigned const GiNaC::crctab[256] [static]
```

Referenced by [crc32\(\)](#).

#### 5.1.4.9 library\_initializer

```
library_init GiNaC::library_initializer [static]
```

For construction of flyweights, etc.

#### 5.1.4.10 \_num0\_bp

```
const basic * GiNaC::_num0_bp
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.11 idx

```
GiNaC::idx
```

Referenced by [expand\\_dummy\\_sum\(\)](#).

#### 5.1.4.12 force\_include\_tgamma

```
unsigned GiNaC::force_include_tgamma = tgamma_SERIAL::serial
```

#### 5.1.4.13 force\_include\_zeta1

```
unsigned GiNaC::force_include_zeta1 = zeta1_SERIAL::serial
```

#### 5.1.4.14 GINAC\_BIND\_UNARCHIVER

```
template<>
GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(lst, basic, print_func< print_context >(&lst::do_print).
print_func< print_tree >(&lst::do_print_tree)) template<> bool ls GiNaC::GINAC_BIND_UNARCHIVER)
(lst) (
 lst)
```

Specialization of [container::info\(\)](#) for lst.

#### 5.1.4.15 I

```
const numeric GiNaC::I = numeric(cln::complex(cln::cl_I(0),cln::cl_I(1)))
```

Imaginary unit.

This is not a constant but a numeric since we are natively handing complex numbers anyways, so in each expression containing an I it is automatically eval'ed away anyhow.

Referenced by [acosh\\_eval\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_a0\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [color\\_h\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [cosh\\_eval\(\)](#), [csgn\\_eval\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [eta\\_imag\\_part\(\)](#), [exp\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#), [H\\_evalf\(\)](#), [GiNaC::numeric::has\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_eval\(\)](#), [log\\_eval\(\)](#), [log\\_series\(\)](#), [GiNaC::numeric::normal\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), [sinh\\_eval\(\)](#), [step\\_eval\(\)](#), [tanh\\_eval\(\)](#), [tanh\\_series\(\)](#), [GiNaC::numeric::to\\_polynomial\(\)](#) and [GiNaC::numeric::to\\_rational\(\)](#).

#### 5.1.4.16 Digits

```
_numeric_digits GiNaC::Digits
```

Accuracy in decimal digits.

Only object of this type! Can be set using assignment from C++ unsigned ints and evaluated like any built-in type.

Referenced by [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::numeric::numeric\(\)](#), [print\\_real\\_cl\\_N\(\)](#), and [zeta1\\_evalf\(\)](#).

#### 5.1.4.17 next\_print\_context\_id

```
unsigned GiNaC::next_print_context_id = 0
```

Next free ID for [print\\_context](#) types.

#### 5.1.4.18 version\_major

```
const int GiNaC::version_major = GINACLIB_MAJOR_VERSION
```

#### 5.1.4.19 version\_minor

```
const int GiNaC::version_minor = GINACLIB_MINOR_VERSION
```

#### 5.1.4.20 version\_micro

```
const int GiNaC::version_micro = GINACLIB_MICRO_VERSION
```

#### 5.1.4.21 `_num_120_p`

```
const numeric * GiNaC::_num_120_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.22 `_ex_120`

```
const ex GiNaC::_ex_120 = ex(*_num_120_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.23 `_num_60_p`

```
const numeric * GiNaC::_num_60_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.24 `_ex_60`

```
const ex GiNaC::_ex_60 = ex(*_num_60_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.25 `_num_48_p`

```
const numeric * GiNaC::_num_48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.26 `_ex_48`

```
const ex GiNaC::_ex_48 = ex(*_num_48_p)
```

Referenced by [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.27 `_num_30_p`

```
const numeric * GiNaC::_num_30_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.28 `_ex_30`

```
const ex GiNaC::_ex_30 = ex(*_num_30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.29 `_num_25_p`

```
const numeric * GiNaC::_num_25_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.30 `_ex_25`

```
const ex GiNaC::_ex_25 = ex(*_num_25_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.31 `_num_24_p`

```
const numeric * GiNaC::_num_24_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.32 `_ex_24`

```
const ex GiNaC::_ex_24 = ex(*_num_24_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.33 `_num_20_p`

```
const numeric * GiNaC::_num_20_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.34 `_ex_20`

```
const ex GiNaC::_ex_20 = ex(*_num_20_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.35 `_num_18_p`

```
const numeric * GiNaC::_num_18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.36 `_ex_18`

```
const ex GiNaC::_ex_18 = ex(*_num_18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.37 `_num_15_p`

```
const numeric * GiNaC::_num_15_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.38 `_ex_15`

```
const ex GiNaC::_ex_15 = ex(*_num_15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.39 `_num_12_p`

```
const numeric * GiNaC::_num_12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.40 `_ex_12`

```
const ex GiNaC::_ex_12 = ex(*_num_12_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.41 `_num_11_p`

```
const numeric * GiNaC::_num_11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.42 `_ex_11`

```
const ex GiNaC::_ex_11 = ex(*_num_11_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.43 `_num_10_p`

```
const numeric * GiNaC::_num_10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.44 `_ex_10`

```
const ex GiNaC::_ex_10 = ex(*_num_10_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.45 `_num_9_p`

```
const numeric * GiNaC::_num_9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.46 `_ex_9`

```
const ex GiNaC::_ex_9 = ex(*_num_9_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.47 `_num_8_p`

```
const numeric * GiNaC::_num_8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.48 `_ex_8`

```
const ex GiNaC::_ex_8 = ex(*_num_8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.49 `_num_7_p`

```
const numeric * GiNaC::_num_7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.50 `_ex_7`

```
const ex GiNaC::_ex_7 = ex(*_num_7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.51 `_num_6_p`

```
const numeric * GiNaC::_num_6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.52 `_ex_6`

```
const ex GiNaC::_ex_6 = ex(*_num_6_p)
```

Referenced by [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.53 `_num_5_p`

```
const numeric * GiNaC::_num_5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.54 `_ex_5`

```
const ex GiNaC::_ex_5 = ex(*_num_5_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.55 `_num_4_p`

```
const numeric * GiNaC::_num_4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.56 `_ex_4`

```
const ex GiNaC::_ex_4 = ex(*_num_4_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.57 `_num_3_p`

```
const numeric * GiNaC::_num_3_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.58 `_ex_3`

```
const ex GiNaC::_ex_3 = ex(*_num_3_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.59 `_num_2_p`

```
const numeric * GiNaC::_num_2_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [tgamma\\_eval\(\)](#).

**5.1.4.60 `_ex_2`**

```
const ex GiNaC::_ex_2 = ex(*_num_2_p)
```

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.61 `_num_1_p`**

```
const numeric * GiNaC::_num_1_p
```

Referenced by [acos\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [beta\\_eval\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator-\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi1\\_eval\(\)](#), and [psi2\\_eval\(\)](#).

**5.1.4.62 `_ex_1`**

```
const ex GiNaC::_ex_1 = ex(*_num_1_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [acosh\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_deriv\(\)](#), [atan\\_deriv\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_deriv\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [exminus\(\)](#), [exp\\_eval\(\)](#), [exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [frac\\_cancel\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [lgamma\\_eval\(\)](#), [Li2\\_eval\(\)](#), [Li\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_deriv\(\)](#), [log\\_expand\(\)](#), [log\\_series\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator/\(\)](#), [operator/\(\)](#), [operator/\(\)](#), [psi1\\_series\(\)](#), [psi2\\_eval\(\)](#), [psi2\\_series\(\)](#), [replace\\_with\\_symbol\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.63 `_num_1_2_p`**

```
const numeric * GiNaC::_num_1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.64 `_ex_1_2`**

```
const ex GiNaC::_ex_1_2 = ex(*_num_1_2_p)
```

Referenced by [acos\\_deriv\(\)](#), [acos\\_eval\(\)](#), [acosh\\_deriv\(\)](#), [asin\\_deriv\(\)](#), [asin\\_eval\(\)](#), [asinh\\_deriv\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [cosh\\_eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [zeta1\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.65 `_num_1_3_p`**

```
const numeric * GiNaC::_num_1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.66 `_ex_1_3`

```
const ex GiNaC::_ex_1_3 = ex(*_num_1_3_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.67 `_num_1_4_p`

```
const numeric * GiNaC::_num_1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.68 `_ex_1_4`

```
const ex GiNaC::_ex_1_4 = ex(*_num_1_4_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.69 `_num0_p`

```
const numeric * GiNaC::_num0_p
```

Referenced by [GiNaC::numeric::add\\_dyn\(\)](#), [atan\(\)](#), [binomial\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [fibonacci\(\)](#), [GiNaC::numeric::has\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [iquo\(\)](#), [iquo\(\)](#), [irem\(\)](#), [irem\(\)](#), [isqrt\(\)](#), [Li2\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [mod\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [sin\\_eval\(\)](#), [smod\(\)](#), [GiNaC::numeric::sub\\_dyn\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.70 `_ex0`

```
const ex GiNaC::_ex0 = ex(*_num0_p)
```

Referenced by [acos\\_eval\(\)](#), [acosh\\_eval\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [asinh\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_eval\(\)](#), [atanh\\_series\(\)](#), [beta\\_eval\(\)](#), [binomial\\_sym\(\)](#), [GiNaC::relational::canonical\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [color\\_trace\(\)](#), [GiNaC::ex::content\(\)](#), [cos\\_eval\(\)](#), [csgn\\_series\(\)](#), [GiNaC::expairseq::default\\_overall\\_coeff\(\)](#), [GiNaC::basic::derivative\(\)](#), [GiNaC::constant::derivative\(\)](#), [GiNaC::idx::derivative\(\)](#), [GiNaC::indexed::derivative\(\)](#), [GiNaC::symbol::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [divide\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [EllipticE\\_eval\(\)](#), [EllipticE\\_series\(\)](#), [EllipticK\\_eval\(\)](#), [EllipticK\\_series\(\)](#), [eta\\_eval\(\)](#), [eta\\_evalf\(\)](#), [eta\\_series\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::indexed::expand\(\)](#), [find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [G2\\_eval\(\)](#), [G2\\_evalf\(\)](#), [G3\\_eval\(\)](#), [G3\\_evalf\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [gcd\(\)](#), [H\\_deriv\(\)](#), [H\\_eval\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [Li2\\_eval\(\)](#), [Li2\\_series\(\)](#), [Li\\_deriv\(\)](#), [Li\\_eval\(\)](#), [Li\\_evalf\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [Order\\_eval\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [rem\(\)](#), [S\\_deriv\(\)](#), [S\\_eval\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::symbol::series\(\)](#), [sin\\_eval\(\)](#), [sinh\\_eval\(\)](#), [sprem\(\)](#), [sqrfree\(\)](#), [sr\\_gcd\(\)](#), [step\\_series\(\)](#), [tan\\_eval\(\)](#), [tanh\\_eval\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [zeta1\\_deriv\(\)](#), [zeta1\\_eval\(\)](#), [zeta2\\_deriv\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.71** `_num1_4_p`

```
const numeric * GiNaC::_num1_4_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.72** `_ex1_4`

```
const ex GiNaC::_ex1_4 = ex(*_num1_4_p)
```

Referenced by [atan2\\_eval\(\)](#), [atan\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.73** `_num1_3_p`

```
const numeric * GiNaC::_num1_3_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

**5.1.4.74** `_ex1_3`

```
const ex GiNaC::_ex1_3 = ex(*_num1_3_p)
```

Referenced by [acos\\_eval\(\)](#), [cos\\_eval\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.75** `_num1_2_p`

```
const numeric * GiNaC::_num1_2_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), [psi2\\_eval\(\)](#), and [tgamma\\_eval\(\)](#).

**5.1.4.76** `_ex1_2`

```
const ex GiNaC::_ex1_2 = ex(*_num1_2_p)
```

Referenced by [acos\\_eval\(\)](#), [asin\\_eval\(\)](#), [atan2\\_eval\(\)](#), [atan\\_series\(\)](#), [atanh\\_series\(\)](#), [cos\\_eval\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [log\\_eval\(\)](#), [psi1\\_eval\(\)](#), [psi2\\_eval\(\)](#), [sin\\_eval\(\)](#), [sqrt\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

**5.1.4.77** `_num1_p`

```
const numeric * GiNaC::_num1_p
```

Referenced by [acos\\_conjugate\(\)](#), [acosh\\_conjugate\(\)](#), [asin\\_conjugate\(\)](#), [asinh\\_conjugate\(\)](#), [atan\(\)](#), [atan\\_conjugate\(\)](#), [atanh\\_conjugate\(\)](#), [bernoulli\(\)](#), [binomial\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\\_dyn\(\)](#), [divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [doublefactorial\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [exp\\_eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [frac\\_cancel\(\)](#), [gcd\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::basic::integer\\_content\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [lcm\\_of\\_coefficients\\_denominators\(\)](#), [lcmcoeff\(\)](#), [Li2\\_conjugate\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [GiNaC::basic::max\\_coefficient\(\)](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [multiply\\_lcm\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [GiNaC::numeric::power\\_dyn\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [psi2\\_eval\(\)](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [tgamma\\_eval\(\)](#), and [zeta1\\_eval\(\)](#).

### 5.1.4.78 `_ex1`

```
const ex GiNaC::_ex1 = ex(*_num1_p)
```

Referenced by `acos_eval()`, `acosh_deriv()`, `acosh_eval()`, `GiNaC::pseries::add_series()`, `asin_eval()`, `asinh_deriv()`, `atan_deriv()`, `atan_eval()`, `atan_series()`, `atanh_deriv()`, `atanh_eval()`, `atanh_series()`, `beta_eval()`, `binomial_sym()`, `GiNaC::basic::coeff()`, `GiNaC::ncmul::coeff()`, `GiNaC::power::coeff()`, `GiNaC::basic::collect()`, `color_trace()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::combine_ex_with_coeff_to_pair()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `GiNaC::su3t::contract_with()`, `GiNaC::su3f::contract_with()`, `GiNaC::su3d::contract_with()`, `GiNaC::matrix::contract_with()`, `GiNaC::spinmetric::contract_with()`, `GiNaC::tensepsilon::contract_with()`, `cos_eval()`, `cosh_eval()`, `GiNaC::mul::default_overall_coeff()`, `GiNaC::mul::derivative()`, `GiNaC::power::derivative()`, `GiNaC::symbol::derivative()`, `GiNaC::matrix::determinant_minor()`, `divide()`, `divide_in_z()`, `GiNaC::add::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `EllipticE_eval()`, `EllipticE_series()`, `EllipticK_series()`, `GiNaC::mul::eval()`, `GiNaC::ncmul::eval()`, `GiNaC::power::eval()`, `GiNaC::tensdelta::eval_indexed()`, `GiNaC::minkmetric::eval_indexed()`, `GiNaC::spinmetric::eval_indexed()`, `GiNaC::mul::evalm()`, `exp_eval()`, `GiNaC::expairseq::expair_needs_further_processing()`, `GiNaC::mul::expair_needs_further_processing()`, `GiNaC::mul::expand()`, `GiNaC::ncmul::expand()`, `GiNaC::power::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_mul()`, `find_common_factor()`, `frac_cancel()`, `GiNaC::matrix::fraction_free_elimination()`, `G2_eval()`, `G2_evalf()`, `G3_eval()`, `G3_evalf()`, `gcd()`, `gcd_pf_pow()`, `gcd_pf_pow_pow()`, `GINAC_IMPLEMENT_REGISTERED_CLASS`, `H_deriv()`, `H_eval()`, `GiNaC::power::imag_part()`, `GiNaC::matrix::inverse()`, `lgamma_series()`, `Li2_deriv()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `Li_evalf()`, `Li_series()`, `GiNaC::library_init::library_init()`, `log_eval()`, `log_expand()`, `log_series()`, `GiNaC::expairseq::make_flat()`, `GiNaC::expairseq::make_flat()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::pseries::mul_series()`, `GiNaC::basic::normal()`, `GiNaC::power::normal()`, `GiNaC::pseries::normal()`, `GiNaC::symbol::normal()`, `operator++()`, `operator++()`, `Order_eval()`, `Order_series()`, `GiNaC::matrix::pow()`, `GiNaC::pseries::power_const()`, `prem()`, `GiNaC::ex::primpart()`, `GiNaC::pseries::print_series()`, `psi1_deriv()`, `psi1_series()`, `psi2_deriv()`, `psi2_eval()`, `psi2_series()`, `quo()`, `GiNaC::power::real_part()`, `GiNaC::mul::recombine_pair_to_ex()`, `GiNaC::tensor::replace_contr_index()`, `S_series()`, `GiNaC::basic::series()`, `GiNaC::add::series()`, `GiNaC::integral::series()`, `GiNaC::pseries::series()`, `GiNaC::mul::series()`, `GiNaC::power::series()`, `GiNaC::symbol::series()`, `sin_eval()`, `sinh_eval()`, `GiNaC::expairseq::split_ex_to_pair()`, `GiNaC::add::split_ex_to_pair()`, `GiNaC::mul::split_ex_to_pair()`, `sprem()`, `sqrfree_pfrac()`, `sqrfree_yun()`, `sr_gcd()`, `tan_deriv()`, `tan_eval()`, `tanh_deriv()`, `tanh_eval()`, `tgamma_series()`, `GiNaC::expairseq::to_polynomial()`, `GiNaC::expairseq::to_rational()`, `GiNaC::ex::unit()`, `unit_matrix()`, `GiNaC::ex::unitcontprim()`, `zeta1_deriv()`, `zeta2_deriv()`, and `GiNaC::library_init::~~library_init()`.

### 5.1.4.79 `_num2_p`

```
const numeric * GiNaC::_num2_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `exp_eval()`, `GiNaC::power::expand_add_2()`, `Li2_series()`, `GiNaC::library_init::library_init()`, `GiNaC::matrix::pow()`, `psi1_eval()`, `psi2_eval()`, `tgamma_eval()`, and `zeta1_eval()`.

### 5.1.4.80 `_ex2`

```
const ex GiNaC::_ex2 = ex(*_num2_p)
```

Referenced by `acos_deriv()`, `asin_deriv()`, `asinh_deriv()`, `atan2_deriv()`, `atan_deriv()`, `atanh_deriv()`, `GiNaC::spinmetric::contract_with()`, `cos_eval()`, `cosh_eval()`, `csgn_power()`, `epsilon_tensor()`, `exp_eval()`, `GiNaC::power::expand_add_2()`, `Li2_eval()`, `Li2_series()`, `Li_eval()`, `GiNaC::library_init::library_init()`, `product_to_exvector()`, `psi1_eval()`, `sin_eval()`, `sinh_eval()`, `tan_deriv()`, `tan_eval()`, `tanh_deriv()`, `tanh_eval()`, and `GiNaC::library_init::~~library_init()`.

### 5.1.4.81 `_num3_p`

```
const numeric * GiNaC::_num3_p
```

Referenced by `GiNaC::ex::construct_from_int()`, `GiNaC::ex::construct_from_long()`, `GiNaC::ex::construct_from_uint()`, `GiNaC::ex::construct_from_ulong()`, `exp_eval()`, and `GiNaC::library_init::library_init()`.

#### 5.1.4.82 `_ex3`

```
const ex GiNaC::_ex3 = ex(*_num3_p)
```

Referenced by [color\\_trace\(\)](#), [cos\\_eval\(\)](#), [epsilon\\_tensor\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.83 `_num4_p`

```
const numeric * GiNaC::_num4_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [exp\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.84 `_ex4`

```
const ex GiNaC::_ex4 = ex(*_num4_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), [lorentz\\_eps\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.85 `_num5_p`

```
const numeric * GiNaC::_num5_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.86 `_ex5`

```
const ex GiNaC::_ex5 = ex(*_num5_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.87 `_num6_p`

```
const numeric * GiNaC::_num6_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 5.1.4.88 `_ex6`

```
const ex GiNaC::_ex6 = ex(*_num6_p)
```

Referenced by [cos\\_eval\(\)](#), [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.89 `_num7_p`

```
const numeric * GiNaC::_num7_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.90 `_ex7`

```
const ex GiNaC::_ex7 = ex(*_num7_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.91 `_num8_p`

```
const numeric * GiNaC::_num8_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.92 `_ex8`

```
const ex GiNaC::_ex8 = ex(*_num8_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.93 `_num9_p`

```
const numeric * GiNaC::_num9_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.94 `_ex9`

```
const ex GiNaC::_ex9 = ex(*_num9_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.95 `_num10_p`

```
const numeric * GiNaC::_num10_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.96 `_ex10`

```
const ex GiNaC::_ex10 = ex(*_num10_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.97 `_num11_p`

```
const numeric * GiNaC::_num11_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.98 `_ex11`

```
const ex GiNaC::_ex11 = ex(*_num11_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.99 `_num12_p`

```
const numeric * GiNaC::_num12_p
```

Referenced by [GiNaC::ex::construct\\_from\\_int\(\)](#), [GiNaC::ex::construct\\_from\\_long\(\)](#), [GiNaC::ex::construct\\_from\\_uint\(\)](#), [GiNaC::ex::construct\\_from\\_ulong\(\)](#), [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.100 `_ex12`

```
const ex GiNaC::_ex12 = ex(*_num12_p)
```

Referenced by [Li2\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.101 `_num15_p`

```
const numeric * GiNaC::_num15_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.102 `_ex15`

```
const ex GiNaC::_ex15 = ex(*_num15_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.103 `_num18_p`

```
const numeric * GiNaC::_num18_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 5.1.4.104 `_ex18`

```
const ex GiNaC::_ex18 = ex(*_num18_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.105 `_num20_p`

```
const numeric * GiNaC::_num20_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.106 `_ex20`

```
const ex GiNaC::_ex20 = ex(*_num20_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.107 `_num24_p`

```
const numeric * GiNaC::_num24_p
```

Referenced by [cos\\_eval\(\)](#), and [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.108 `_ex24`

```
const ex GiNaC::_ex24 = ex(*_num24_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.109 `_num25_p`

```
const numeric * GiNaC::_num25_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.110 `_ex25`

```
const ex GiNaC::_ex25 = ex(*_num25_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.111 `_num30_p`

```
const numeric * GiNaC::_num30_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.112 `_ex30`

```
const ex GiNaC::_ex30 = ex(*_num30_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.113 `_num48_p`

```
const numeric * GiNaC::_num48_p
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#).

#### 5.1.4.114 `_ex48`

```
const ex GiNaC::_ex48 = ex(*_num48_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.115 `_num60_p`

```
const numeric * GiNaC::_num60_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), and [tan\\_eval\(\)](#).

#### 5.1.4.116 `_ex60`

```
const ex GiNaC::_ex60 = ex(*_num60_p)
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), [sin\\_eval\(\)](#), [tan\\_eval\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

#### 5.1.4.117 `_num120_p`

```
const numeric * GiNaC::_num120_p
```

Referenced by [cos\\_eval\(\)](#), [GiNaC::library\\_init::library\\_init\(\)](#), and [sin\\_eval\(\)](#).

#### 5.1.4.118 `_ex120`

```
const ex GiNaC::_ex120 = ex(*_num120_p)
```

Referenced by [GiNaC::library\\_init::library\\_init\(\)](#), and [GiNaC::library\\_init::~~library\\_init\(\)](#).

## 5.2 GiNaC::internal Namespace Reference

### Classes

- struct [\\_iter\\_rep](#)

## 5.3 std Namespace Reference

### Classes

- struct [equal\\_to< GiNaC::ex >](#)  
*Specialization of `std::equal_to()` for `ex` objects.*
- struct [hash< GiNaC::ex >](#)  
*Specialization of `std::hash()` for `ex` objects.*
- struct [less< GiNaC::ptr< T > >](#)  
*Specialization of `std::less` for `ptr< T >` to enable ordering of `ptr< T >` objects (e.g.*

### Functions

- template<> void [swap](#) ([GiNaC::ex](#) &a, [GiNaC::ex](#) &b)  
*Specialization of [std::swap\(\)](#) for `ex` objects.*

### 5.3.1 Function Documentation

#### 5.3.1.1 swap()

```
template<>
void std::swap (
 GiNaC::ex & a,
 GiNaC::ex & b) [inline]
```

Specialization of [std::swap\(\)](#) for `ex` objects.

References [GiNaC::ex::swap\(\)](#).

Referenced by [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), and [GiNaC::permutation\\_sign\(\)](#).

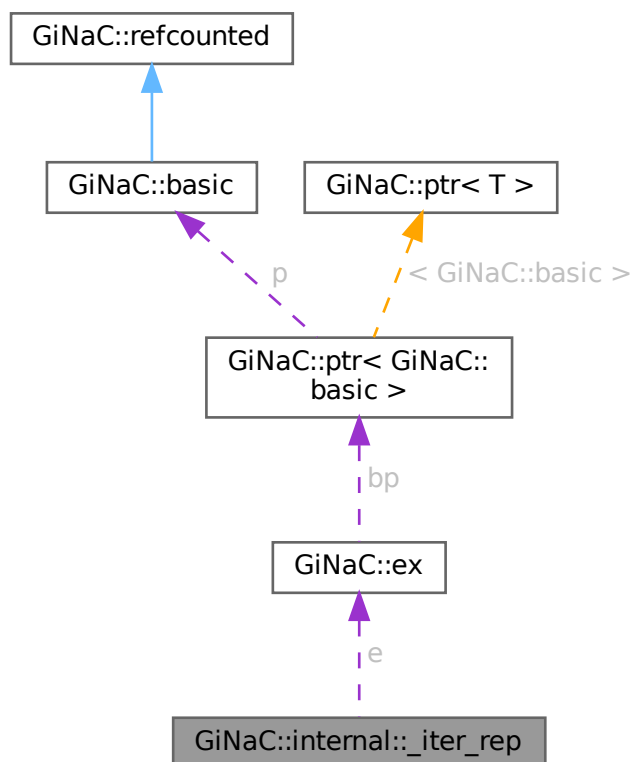
## Chapter 6

# Class Documentation

### 6.1 GiNaC::internal::\_iter\_rep Struct Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::internal::\_iter\_rep:



## Public Member Functions

- [\\_iter\\_rep](#) (const [ex](#) &[e\\_](#), [size\\_t](#) [i\\_](#), [size\\_t](#) [i\\_end\\_](#))
- bool [operator==](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept
- bool [operator!=](#) (const [\\_iter\\_rep](#) &[other](#)) const noexcept

## Public Attributes

- [ex](#) [e](#)
- [size\\_t](#) [i](#)
- [size\\_t](#) [i\\_end](#)

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 [\\_iter\\_rep](#)()

```
GiNaC::internal::_iter_rep::_iter_rep (
 const ex & e_,
 size_t i_,
 size_t i_end_) [inline]
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 [operator==](#)()

```
bool GiNaC::internal::_iter_rep::operator== (
 const _iter_rep & other) const [inline], [noexcept]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [e](#), and [i](#).

### 6.1.2.2 [operator!="](#)()

```
bool GiNaC::internal::_iter_rep::operator!= (
 const _iter_rep & other) const [inline], [noexcept]
```

## 6.1.3 Member Data Documentation

### 6.1.3.1 [e](#)

```
ex GiNaC::internal::_iter_rep::e
```

Referenced by [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 6.1.3.2 i

```
size_t GiNaC::internal::_iter_rep::i
```

Referenced by [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), and [operator==\(\)](#).

### 6.1.3.3 i\_end

```
size_t GiNaC::internal::_iter_rep::i_end
```

Referenced by [GiNaC::const\\_preorder\\_iterator::increment\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.2 GiNaC::\_numeric\_digits Class Reference

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

```
#include <numeric.h>
```

### Public Member Functions

- [\\_numeric\\_digits](#) ()  
*\_numeric\_digits* default ctor, checking for singleton invariance.
- [\\_numeric\\_digits & operator=](#) (long prec)  
*Assign a native long to global Digits object.*
- [operator long](#) ()  
*Convert global Digits object to native type long.*
- void [print](#) (std::ostream &os) const  
*Append global Digits object to ostream.*
- void [add\\_callback](#) ([digits\\_changed\\_callback](#) callback)  
*Add a new callback function.*

### Private Attributes

- long [digits](#)  
*Number of decimal digits.*
- std::vector< [digits\\_changed\\_callback](#) > [callbacklist](#)

### Static Private Attributes

- static bool [too\\_late](#) = false  
*Already one object present.*

## 6.2.1 Detailed Description

This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.

We need an object rather than a dumb basic type since as a side-effect we let it change `cl_default_float_format` when it gets changed. The only other meaningful thing to do with it is converting it to an unsigned, for temporarily storing its value e.g. The user must not create an own working object of this class! Since C++ forces us to make the class definition visible in order to use an object we put in a flag which prevents other objects of that class to be created.

## 6.2.2 Constructor & Destructor Documentation

### 6.2.2.1 `_numeric_digits()`

```
GiNaC::_numeric_digits::_numeric_digits ()
```

[\\_numeric\\_digits](#) default ctor, checking for singleton invariance.

References [too\\_late](#).

## 6.2.3 Member Function Documentation

### 6.2.3.1 `operator=()`

```
_numeric_digits & GiNaC::_numeric_digits::operator= (
 long prec)
```

Assign a native long to global Digits object.

References [callbacklist](#), and [digits](#).

### 6.2.3.2 `operator long()`

```
GiNaC::_numeric_digits::operator long ()
```

Convert global Digits object to native type long.

### 6.2.3.3 `print()`

```
void GiNaC::_numeric_digits::print (
 std::ostream & os) const
```

Append global Digits object to ostream.

References [digits](#).

Referenced by [GiNaC::operator<<\(\)](#).

#### 6.2.3.4 add\_callback()

```
void GiNaC::_numeric_digits::add_callback (
 digits_changed_callback callback)
```

Add a new callback function.

References [callbacklist](#).

### 6.2.4 Member Data Documentation

#### 6.2.4.1 digits

```
long GiNaC::_numeric_digits::digits [private]
```

Number of decimal digits.

Referenced by [operator=\(\)](#), and [print\(\)](#).

#### 6.2.4.2 too\_late

```
bool GiNaC::_numeric_digits::too_late = false [static], [private]
```

Already one object present.

Referenced by [\\_numeric\\_digits\(\)](#).

#### 6.2.4.3 callbacklist

```
std::vector<digits_changed_callback> GiNaC::_numeric_digits::callbacklist [private]
```

Referenced by [add\\_callback\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following files:

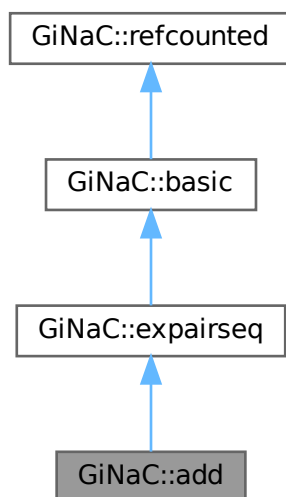
- [numeric.h](#)
- [numeric.cpp](#)

### 6.3 GiNaC::add Class Reference

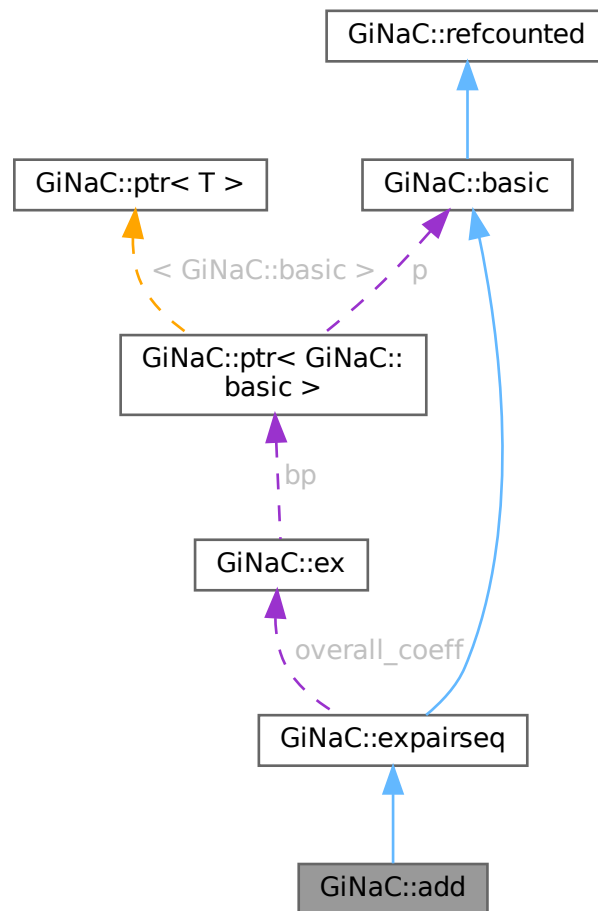
Sum of expressions.

```
#include <add.h>
```

Inheritance diagram for GiNaC::add:



Collaboration diagram for GiNaC::add:



### Public Member Functions

- `add` (const `ex` &lh, const `ex` &rh)
- `add` (const `exvector` &v)
- `add` (const `epvector` &v)
- `add` (const `epvector` &v, const `ex` &oc)
- `add` (`epvector` &&v)
- `add` (`epvector` &&v, const `ex` &oc)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- bool `info` (unsigned inf) const override  
Information about the object.
- bool `is_polynomial` (const `ex` &var) const override  
Check whether this is a polynomial in the given variables.
- int `degree` (const `ex` &s) const override  
Return degree of highest power in object s.

- `int ldegree (const ex &s) const` override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override  
*Return coefficient of degree n in object s.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &r, int order, unsigned options=0) const` override  
*Implementation of [ex::series\(\)](#) for sums.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override  
*Implementation of [ex::normal\(\)](#) for a sum.*
- `numeric integer_content ()` const override
- `ex smod (const numeric &xi) const` override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient ()` const override  
*Implementation [ex::max\\_coefficient\(\)](#).*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- `ex eval_ncmul (const exvector &v) const` override

## Public Member Functions inherited from [GiNaC::expairseq](#)

- `expairseq (const ex &lh, const ex &rh)`
- `expairseq (const exvector &v)`
- `expairseq (const epvector &v, const ex &oc, bool do_index_renaming=false)`
- `expairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false)`
- `unsigned precedence ()` const override  
*Return relative operator precedence (for parenthezing output).*
- `bool info (unsigned inf) const` override  
*Information about the object.*
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i) const` override  
*Return operand/member at position i.*
- `ex map (map\_function &f) const` override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval ()` const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational (exmap &repl) const` override  
*Implementation of [ex::to\\_rational\(\)](#) for expairseqs.*
- `ex to_polynomial (exmap &repl) const` override  
*Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.*
- `bool match (const ex &pattern, exmap &repl_lst) const` override  
*Check whether the expression matches a given pattern.*
- `ex subs (const exmap &m, unsigned options=0) const` override  
*Substitute a set of objects by arbitrary expressions.*
- `ex conjugate ()` const override
- `void archive (archive\_node &n) const` override  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive\_node &n, lst &syms) override`  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const`  
*Try to contract two indexed expressions that appear in the same product.*
- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like print(), but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like print(), but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for subs().*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of nth derivative ex::diff(s, n).*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*

- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a sum.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex thisexpairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false) const override  
*Create an object of this type.*
- [ex thisexpairseq](#) ([epvector](#) &&vp, const [ex](#) &oc, bool do\_index\_renaming=false) const override
- [expair split\\_ex\\_to\\_pair](#) (const [ex](#) &e) const override  
*Form an expair from an ex, using the corresponding semantics.*
- [expair combine\\_ex\\_with\\_coeff\\_to\\_pair](#) (const [ex](#) &e, const [ex](#) &c) const override
- [expair combine\\_pair\\_with\\_coeff\\_to\\_pair](#) (const [expair](#) &p, const [ex](#) &c) const override
- [ex recombine\\_pair\\_to\\_ex](#) (const [expair](#) &p) const override  
*Form an ex out of an expair, using the corresponding semantics.*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- void [print\\_add](#) (const [print\\_context](#) &c, const char \*openbrace, const char \*closebrace, const char \*mul\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::expairseq](#)

- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [return\\_type](#) () const override
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*

- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- virtual bool `expair_needs_further_processing` (epp it)
- virtual `ex` `default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expair` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expairseq` (const `expairseq` &s1, const `expairseq` &s2)
- void `construct_from_expairseq_ex` (const `expairseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
*Combine this expairseq with argument exvector.*
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
*Combine this expairseq with argument epvector.*
- void `canonicalize` ()  
*Brings this expairseq into a sorted (canonical) form.*
- void `combine_same_terms_sorted_seq` ()  
*Compact a presorted expairseq by combining all matching expairs to one each.*
- bool `is_canonical` () const  
*Check if this expairseq is in sorted (canonical) form.*
- `epvector` `expandchildren` (unsigned `options`) const  
*Member-wise expand the expairs in this sequence.*
- `epvector` `evalchildren` () const  
*Member-wise evaluate the expairs in this sequence.*
- `epvector` `subchildren` (const `exmap` &m, unsigned `options`=0) const  
*Member-wise substitute in this sequence.*

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Friends

- class [mul](#)
- class [power](#)

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::expairseq](#)

- [epvector seq](#)
- [ex overall\\_coeff](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.3.1 Detailed Description

Sum of expressions.

## 6.3.2 Constructor & Destructor Documentation

### 6.3.2.1 `add()` [1/6]

```
GiNaC::add::add (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [conjugate\(\)](#).

### 6.3.2.2 `add()` [2/6]

```
GiNaC::add::add (
 const exvector & v)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.3 add()** [3/6]

```
GiNaC::add::add (
 const epvector & v)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.4 add()** [4/6]

```
GiNaC::add::add (
 const epvector & v,
 const ex & oc)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.5 add()** [5/6]

```
GiNaC::add::add (
 epvector && v)
```

References [GiNaC::\\_ex0](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.2.6 add()** [6/6]

```
GiNaC::add::add (
 epvector && v,
 const ex & oc)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.3 Member Function Documentation****6.3.3.1 precedence()**

```
unsigned GiNaC::add::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_csrc\(\)](#), and [print\\_add\(\)](#).

### 6.3.3.2 info()

```
bool GiNaC::add::info (
 unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [info\(\)](#).

### 6.3.3.3 is\_polynomial()

```
bool GiNaC::add::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

### 6.3.3.4 degree()

```
int GiNaC::add::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.5 ldegree()

```
int GiNaC::add::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.6 coeff()

```
ex GiNaC::add::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::clifford\\_max\\_label\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::dirac\\_ONE\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

Referenced by [print\\_add\(\)](#), and [smod\(\)](#).

### 6.3.3.7 eval()

```
ex GiNaC::add::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x stands for a symbolic variables of type ex and c stands for such an expression that contain a plain number.

- $+(;c) \rightarrow c$
- $+(x;0) \rightarrow x$

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

### 6.3.3.8 evalm()

```
ex GiNaC::add::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::matrix::add\(\)](#), [GiNaC::ex::evalm\(\)](#), [m](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.3.3.9 series()

```
ex GiNaC::add::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for sums.

This performs series addition when adding pseries objects.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 6.3.3.10 normal()

```
ex GiNaC::add::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a sum.

It expands terms and performs fractional addition.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [expand\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.11 integer\_content()

```
numeric GiNaC::add::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::denom\(\)](#), [GiNaC::gcd\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lcm\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.3.3.12 smod()

```
ex GiNaC::add::smod (
 const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

## Parameters

|      |         |
|------|---------|
| $xi$ | modulus |
|------|---------|

## Returns

mapped polynomial

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

**6.3.3.13 max\_coefficient()**

```
numeric GiNaC::add::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

**6.3.3.14 conjugate()**

```
ex GiNaC::add::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.3.3.15 real\_part()**

```
ex GiNaC::add::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.3.3.16 `imag_part()`

```
ex GiNaC::add::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.3.3.17 `get_free_indices()`

```
exvector GiNaC::add::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indices\\_consistent\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

### 6.3.3.18 `eval_ncmul()`

```
ex GiNaC::add::eval_ncmul (
 const exvector & v) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

### 6.3.3.19 `derivative()`

```
ex GiNaC::add::derivative (
 const symbol & y) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a sum.

It differentiates each term.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

### 6.3.3.20 `return_type()`

```
unsigned GiNaC::add::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), and [GiNaC::expairseq::seq](#).

**6.3.3.21 return\_type\_tinfo()**

```
return_type_t GiNaC::add::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::seq](#).

**6.3.3.22 thisexpairseq() [1/2]**

```
ex GiNaC::add::thisexpairseq (
 const epvector & v,
 const ex & oc,
 bool do_index_renaming = false) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.23 thisexpairseq() [2/2]**

```
ex GiNaC::add::thisexpairseq (
 epvector && vp,
 const ex & oc,
 bool do_index_renaming = false) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.3.3.24 split\_ex\_to\_pair()**

```
expair GiNaC::add::split_ex_to_pair (
 const ex & e) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [evalm\(\)](#), [imag\\_part\(\)](#), and [real\\_part\(\)](#).

**6.3.3.25 combine\_ex\_with\_coeff\_to\_pair()**

```

expair GiNaC::add::combine_ex_with_coeff_to_pair (
 const ex & e,
 const ex & c) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::ex::is\\_equal\(\)](#), [likely](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.3.3.26 combine\_pair\_with\_coeff\_to\_pair()**

```

expair GiNaC::add::combine_pair_with_coeff_to_pair (
 const expair & p,
 const ex & c) const [override], [protected], [virtual]

```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::expair::rest](#).

Referenced by [GiNaC::mul::eval\(\)](#), and [GiNaC::power::expand\\_add\\_2\(\)](#).

**6.3.3.27 recombine\_pair\_to\_ex()**

```

ex GiNaC::add::recombine_pair_to_ex (
 const expair & p) const [override], [protected], [virtual]

```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::expair::coeff](#), and [GiNaC::expair::rest](#).

Referenced by [eval\(\)](#), [evalm\(\)](#), [GiNaC::power::expand\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [normal\(\)](#), and [real\\_part\(\)](#).

**6.3.3.28 expand()**

```

ex GiNaC::add::expand (
 unsigned options = 0) const [override], [protected], [virtual]

```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [normal\(\)](#).

**6.3.3.29 print\_add()**

```
void GiNaC::add::print_add (
 const print_context & c,
 const char * openbrace,
 const char * closebrace,
 const char * mul_sym,
 unsigned level) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**6.3.3.30 do\_print()**

```
void GiNaC::add::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_add\(\)](#).

**6.3.3.31 do\_print\_latex()**

```
void GiNaC::add::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_add\(\)](#).

**6.3.3.32 do\_print\_csrc()**

```
void GiNaC::add::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::info\\_flags::real](#), and [GiNaC::expairseq::seq](#).

**6.3.3.33 do\_print\_python\_repr()**

```
void GiNaC::add::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), and [GiNaC::ex::print\(\)](#).

## 6.3.4 Friends And Related Symbol Documentation

### 6.3.4.1 mul

```
friend class mul [friend]
```

### 6.3.4.2 power

```
friend class power [friend]
```

The documentation for this class was generated from the following files:

- [add.h](#)
- [add.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.4 GiNaC::archive Class Reference

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

```
#include <archive.h>
```

### Classes

- struct [archived\\_ex](#)  
*Archived expression descriptor.*

### Public Member Functions

- [archive](#) ()
- [~archive](#) ()
- [archive](#) (const [ex](#) &e)  
*Construct archive from expression using the default name "ex".*
- [archive](#) (const [ex](#) &e, const char \*n)  
*Construct archive from expression using the specified name.*
- void [archive\\_ex](#) (const [ex](#) &e, const char \*name)  
*Archive an expression.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, const char \*name) const  
*Retrieve expression from archive by name.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve expression from archive by index.*
- [ex unarchive\\_ex](#) (const [lst](#) &sym\_lst, std::string &name, unsigned index=0) const  
*Retrieve expression and its name from archive by index.*
- unsigned [num\\_expressions](#) () const  
*Return number of archived expressions.*

- const [archive\\_node](#) & [get\\_top\\_node](#) (unsigned index=0) const  
*Return reference to top node of an expression specified by index.*
- void [clear](#) ()  
*Clear all archived expressions.*
- [archive\\_node\\_id](#) [add\\_node](#) (const [archive\\_node](#) &n)  
*Add [archive\\_node](#) to archive if the corresponding expression is not already archived.*
- [archive\\_node](#) & [get\\_node](#) ([archive\\_node\\_id](#) id)  
*Retrieve [archive\\_node](#) by ID.*
- void [forget](#) ()  
*Delete cached unarchived expressions in all [archive\\_nodes](#) (mainly for debugging).*
- void [prinraw](#) (std::ostream &os) const  
*Print archive to stream in ugly raw format (for debugging).*
- [archive\\_atom](#) [atomize](#) (const std::string &s) const  
*Atomize a string (i.e.*
- const std::string & [unatomize](#) ([archive\\_atom](#) id) const  
*Unatomize a string (i.e.*

### Private Types

- typedef std::map< std::string, [archive\\_atom](#) >::const\_iterator [inv\\_at\\_cit](#)  
*The map of from strings to indices of the atoms vectors allows for faster archiving.*

### Private Attributes

- std::vector< [archive\\_node](#) > [nodes](#)  
*Vector of archived nodes.*
- std::vector< [archived\\_ex](#) > [exprs](#)  
*Vector of archived expression descriptors.*
- std::vector< std::string > [atoms](#)  
*Vector of atomized strings (using a vector allows faster unarchiving).*
- std::map< std::string, [archive\\_atom](#) > [inverse\\_atoms](#)
- std::map< [ex](#), [archive\\_node\\_id](#), [ex\\_is\\_less](#) > [exprtable](#)  
*Map of stored expressions to nodes for faster archiving.*

### Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [archive](#) &ar)  
*Write archive to binary data stream.*
- std::istream & [operator>>](#) (std::istream &is, [archive](#) &ar)  
*Read archive from binary data stream.*

## 6.4.1 Detailed Description

This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).

An archive can be constructed from an expression and then written to a stream; or it can be read from a stream and then unarchived, yielding back the expression. Archives can hold multiple expressions which can be referred to by name or index number. The main component of the archive class is a vector of [archive\\_nodes](#) which each store one object of class [basic](#) (or a derived class).

## 6.4.2 Member Typedef Documentation

### 6.4.2.1 `inv_at_cit`

```
typedef std::map<std::string, archive_atom>::const_iterator GiNaC::archive::inv_at_cit [private]
```

The map of from strings to indices of the atoms vectors allows for faster archiving.

## 6.4.3 Constructor & Destructor Documentation

### 6.4.3.1 `archive()` [1/3]

```
GiNaC::archive::archive () [inline]
```

### 6.4.3.2 `~archive()`

```
GiNaC::archive::~~archive () [inline]
```

### 6.4.3.3 `archive()` [2/3]

```
GiNaC::archive::archive (
 const ex & e) [inline]
```

Construct archive from expression using the default name "ex".

References [archive\\_ex\(\)](#).

### 6.4.3.4 `archive()` [3/3]

```
GiNaC::archive::archive (
 const ex & e,
 const char * n) [inline]
```

Construct archive from expression using the specified name.

References [archive\\_ex\(\)](#), and [n](#).

## 6.4.4 Member Function Documentation

### 6.4.4.1 `archive_ex()`

```
void GiNaC::archive::archive_ex (
 const ex & e,
 const char * name)
```

Archive an expression.

## Parameters

|             |                                           |
|-------------|-------------------------------------------|
| <i>e</i>    | the expression to be archived             |
| <i>name</i> | name under which the expression is stored |

References [add\\_node\(\)](#), [atomize\(\)](#), and [exprs](#).

Referenced by [archive\(\)](#), and [archive\(\)](#).

#### 6.4.4.2 unarchive\_ex() [1/3]

```
ex GiNaC::archive::unarchive_ex (
 const lst & sym_lst,
 const char * name) const
```

Retrieve expression from archive by name.

## Parameters

|                |                             |
|----------------|-----------------------------|
| <i>sym_lst</i> | list of pre-defined symbols |
| <i>name</i>    | name of expression          |

References [atomize\(\)](#), [exprs](#), and [nodes](#).

#### 6.4.4.3 unarchive\_ex() [2/3]

```
ex GiNaC::archive::unarchive_ex (
 const lst & sym_lst,
 unsigned index = 0) const
```

Retrieve expression from archive by index.

## Parameters

|                |                             |
|----------------|-----------------------------|
| <i>sym_lst</i> | list of pre-defined symbols |
| <i>index</i>   | index of expression         |

## See also

[count\\_expressions](#)

References [exprs](#), and [nodes](#).

#### 6.4.4.4 unarchive\_ex() [3/3]

```
ex GiNaC::archive::unarchive_ex (
 const lst & sym_lst,
```

```
std::string & name,
unsigned index = 0) const
```

Retrieve expression and its name from archive by index.

## Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>sym_lst</i> | list of pre-defined symbols         |
| <i>name</i>    | receives the name of the expression |
| <i>index</i>   | index of expression                 |

## See also

`count_expressions`

References [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 6.4.4.5 num\_expressions()

```
unsigned GiNaC::archive::num_expressions () const
```

Return number of archived expressions.

References [exprs](#).

#### 6.4.4.6 get\_top\_node()

```
const archive_node & GiNaC::archive::get_top_node (
 unsigned index = 0) const
```

Return reference to top node of an expression specified by index.

References [exprs](#), and [nodes](#).

#### 6.4.4.7 clear()

```
void GiNaC::archive::clear ()
```

Clear all archived expressions.

References [atoms](#), [exprs](#), [exprtable](#), [inverse\\_atoms](#), and [nodes](#).

#### 6.4.4.8 add\_node()

```
archive_node_id GiNaC::archive::add_node (
 const archive_node & n)
```

Add [archive\\_node](#) to archive if the corresponding expression is not already archived.

## Returns

ID of archived node

References [exprtable](#), [n](#), and [nodes](#).

Referenced by [GiNaC::archive\\_node::add\\_ex\(\)](#), and [archive\\_ex\(\)](#).

#### 6.4.4.9 `get_node()`

```
archive_node & GiNaC::archive::get_node (
 archive_node_id id)
```

Retrieve [archive\\_node](#) by ID.

References [nodes](#).

Referenced by [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_by\\_loc\(\)](#), and [GiNaC::archive\\_node::find\\_ex\\_node\(\)](#).

#### 6.4.4.10 `forget()`

```
void GiNaC::archive::forget ()
```

Delete cached unarchived expressions in all `archive_nodes` (mainly for debugging).

References [GiNaC::archive\\_node::forget\(\)](#), and [nodes](#).

#### 6.4.4.11 `printraw()`

```
void GiNaC::archive::printraw (
 std::ostream & os) const
```

Print archive to stream in ugly raw format (for debugging).

References [atoms](#), [GiNaC::ex::begin\(\)](#), [exprs](#), [nodes](#), and [unatomize\(\)](#).

#### 6.4.4.12 `atomize()`

```
archive_atom GiNaC::archive::atomize (
 const std::string & s) const
```

Atomize a string (i.e.

convert it into an ID number that uniquely represents the string).

References [atoms](#), and [inverse\\_atoms](#).

Referenced by [GiNaC::archive\\_node::add\\_bool\(\)](#), [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_string\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [archive\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_bool\(\)](#), [GiNaC::archive\\_node::find\\_ex\(\)](#), [GiNaC::archive\\_node::find\\_ex\\_node\(\)](#), [GiNaC::archive\\_node::find\\_first\(\)](#), [GiNaC::archive\\_node::find\\_last\(\)](#), [GiNaC::archive\\_node::find\\_property\\_range\(\)](#), [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::archive\\_node::find\\_unsigned\(\)](#), and [unarchive\\_ex\(\)](#).

#### 6.4.4.13 unatomize()

```
const std::string & GiNaC::archive::unatomize (
 archive_atom id) const
```

Unatomize a string (i.e.

convert the ID number back to the string).

References [atoms](#).

Referenced by [GiNaC::archive\\_node::find\\_string\(\)](#), [GiNaC::archive\\_node::get\\_properties\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [printraw\(\)](#), and [unarchive\\_ex\(\)](#).

### 6.4.5 Friends And Related Symbol Documentation

#### 6.4.5.1 operator<<

```
std::ostream & operator<< (
 std::ostream & os,
 const archive & ar) [friend]
```

Write archive to binary data stream.

#### 6.4.5.2 operator>>

```
std::istream & operator>> (
 std::istream & is,
 archive & ar) [friend]
```

Read archive from binary data stream.

### 6.4.6 Member Data Documentation

#### 6.4.6.1 nodes

```
std::vector<archive_node> GiNaC::archive::nodes [private]
```

Vector of archived nodes.

Referenced by [add\\_node\(\)](#), [clear\(\)](#), [forget\(\)](#), [get\\_node\(\)](#), [get\\_top\\_node\(\)](#), [printraw\(\)](#), [unarchive\\_ex\(\)](#), [unarchive\\_ex\(\)](#), and [unarchive\\_ex\(\)](#).

#### 6.4.6.2 exprs

```
std::vector<archived_ex> GiNaC::archive::exprs [private]
```

Vector of archived expression descriptors.

Referenced by [archive\\_ex\(\)](#), [clear\(\)](#), [get\\_top\\_node\(\)](#), [num\\_expressions\(\)](#), [printraw\(\)](#), [unarchive\\_ex\(\)](#), [unarchive\\_ex\(\)](#), and [unarchive\\_ex\(\)](#).

#### 6.4.6.3 atoms

```
std::vector<std::string> GiNaC::archive::atoms [mutable], [private]
```

Vector of atomized strings (using a vector allows faster unarchiving).

Referenced by [atomize\(\)](#), [clear\(\)](#), [printraw\(\)](#), and [unatomize\(\)](#).

#### 6.4.6.4 inverse\_atoms

```
std::map<std::string, archive_atom> GiNaC::archive::inverse_atoms [mutable], [private]
```

Referenced by [atomize\(\)](#), and [clear\(\)](#).

#### 6.4.6.5 exprtable

```
std::map<ex, archive_node_id, ex_is_less> GiNaC::archive::exprtable [mutable], [private]
```

Map of stored expressions to nodes for faster archiving.

Referenced by [add\\_node\(\)](#), and [clear\(\)](#).

The documentation for this class was generated from the following files:

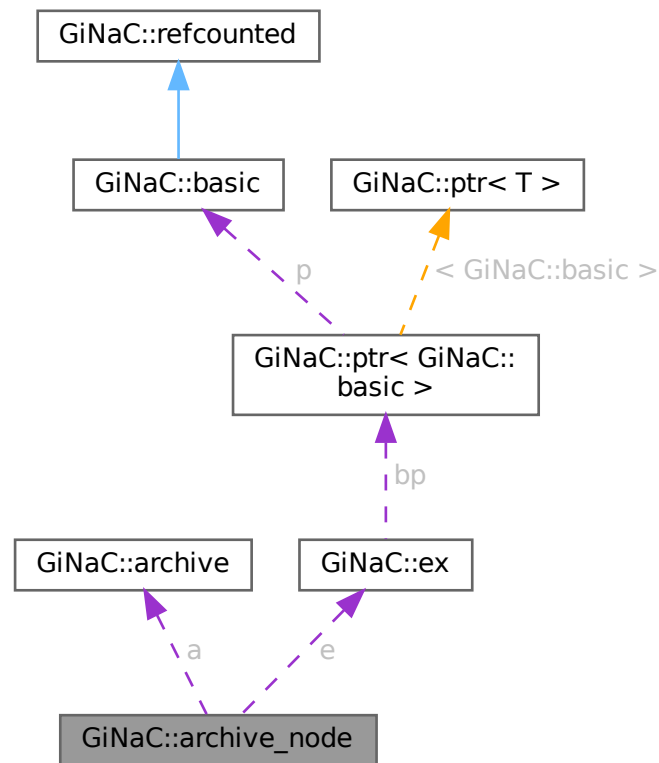
- [archive.h](#)
- [archive.cpp](#)

## 6.5 GiNaC::archive\_node Class Reference

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node:



## Classes

- struct [archive\\_node\\_cit\\_range](#)
- struct [property](#)  
*Archived property (data type, name and associated data)*
- struct [property\\_info](#)  
*Information about a stored property.*

## Public Types

- enum [property\\_type](#) { [PTYPE\\_BOOL](#) , [PTYPE\\_UNSIGNED](#) , [PTYPE\\_STRING](#) , [PTYPE\\_NODE](#) }
- typedef std::vector< [property\\_info](#) > [propinfovector](#)
- typedef std::vector< [property](#) >::const\_iterator [archive\\_node\\_cit](#)

## Public Member Functions

- [archive\\_node](#) ([archive](#) &ar)
- [archive\\_node](#) ([archive](#) &ar, const [ex](#) &expr)  
*Recursively construct archive node from expression.*
- const [archive\\_node](#) & [operator=](#) (const [archive\\_node](#) &other)  
*Assignment operator of [archive\\_node](#).*
- void [add\\_bool](#) (const std::string &name, bool [value](#))  
*Add property of type "bool" to node.*
- void [add\\_unsigned](#) (const std::string &name, unsigned [value](#))  
*Add property of type "unsigned int" to node.*
- void [add\\_string](#) (const std::string &name, const std::string &[value](#))  
*Add property of type "string" to node.*
- void [add\\_ex](#) (const std::string &name, const [ex](#) &[value](#))  
*Add property of type "ex" to node.*
- bool [find\\_bool](#) (const std::string &name, bool &ret, unsigned index=0) const  
*Retrieve property of type "bool" from node.*
- bool [find\\_unsigned](#) (const std::string &name, unsigned &ret, unsigned index=0) const  
*Retrieve property of type "unsigned" from node.*
- bool [find\\_string](#) (const std::string &name, std::string &ret, unsigned index=0) const  
*Retrieve property of type "string" from node.*
- [archive\\_node\\_cit](#) [find\\_first](#) (const std::string &name) const  
*Find the location in the vector of properties of the first/last property with a given name.*
- [archive\\_node\\_cit](#) [find\\_last](#) (const std::string &name) const
- [archive\\_node\\_cit\\_range](#) [find\\_property\\_range](#) (const std::string &name1, const std::string &name2) const  
*Find a range of locations in the vector of properties.*
- bool [find\\_ex](#) (const std::string &name, [ex](#) &ret, [lst](#) &sym\_lst, unsigned index=0) const  
*Retrieve property of type "ex" from node.*
- void [find\\_ex\\_by\\_loc](#) ([archive\\_node\\_cit](#) loc, [ex](#) &ret, [lst](#) &sym\_lst) const  
*Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.*
- const [archive\\_node](#) & [find\\_ex\\_node](#) (const std::string &name, unsigned index=0) const  
*Retrieve property of type "ex" from node, returning the node of the sub-expression.*
- void [get\\_properties](#) ([propinfovector](#) &v) const  
*Return vector of properties stored in node.*
- [ex](#) [unarchive](#) ([lst](#) &sym\_lst) const  
*Convert archive node to [GiNaC](#) expression.*
- bool [has\\_same\\_ex\\_as](#) (const [archive\\_node](#) &other) const  
*Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).*
- bool [has\\_ex](#) () const
- [ex](#) [get\\_ex](#) () const
- void [forget](#) ()  
*Delete cached unarchived expressions from node (for debugging).*
- void [prinraw](#) (std::ostream &os) const  
*Output [archive\\_node](#) to stream in ugly raw format (for debugging).*

### Private Attributes

- [archive](#) & [a](#)  
*Reference to the archive to which this node belongs.*
- `std::vector< property >` [props](#)  
*Vector of stored properties.*
- `bool` [has\\_expression](#)  
*Flag indicating whether a cached unarchived representation of this node exists.*
- `ex e`  
*The cached unarchived representation of this node (if any).*

### Friends

- `std::ostream &` [operator<<](#) (`std::ostream &os`, `const archive\_node &ar`)  
*Write [archive\\_node](#) to binary data stream.*
- `std::istream &` [operator>>](#) (`std::istream &is`, `archive\_node &ar`)  
*Read [archive\\_node](#) from binary data stream.*

## 6.5.1 Detailed Description

This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).

Each property is addressed by its name and data type.

## 6.5.2 Member Typedef Documentation

### 6.5.2.1 propinfovector

```
typedef std::vector<property_info> GiNaC::archive_node::propinfovector
```

### 6.5.2.2 archive\_node\_cit

```
typedef std::vector<property>::const_iterator GiNaC::archive_node::archive_node_cit
```

## 6.5.3 Member Enumeration Documentation

### 6.5.3.1 property\_type

```
enum GiNaC::archive_node::property_type
```

Property data types.

Enumerator

|                |  |
|----------------|--|
| PTYPE_BOOL     |  |
| PTYPE_UNSIGNED |  |
| PTYPE_STRING   |  |
| PTYPE_NODE     |  |

## 6.5.4 Constructor & Destructor Documentation

### 6.5.4.1 `archive_node()` [1/2]

```
GiNaC::archive_node::archive_node (
 archive & ar) [inline]
```

Referenced by [add\\_ex\(\)](#).

### 6.5.4.2 `archive_node()` [2/2]

```
GiNaC::archive_node::archive_node (
 archive & ar,
 const ex & expr)
```

Recursively construct archive node from expression.

References [GiNaC::ex::bp](#).

## 6.5.5 Member Function Documentation

### 6.5.5.1 `operator=()`

```
const archive_node & GiNaC::archive_node::operator= (
 const archive_node & other)
```

Assignment operator of [archive\\_node](#).

References [e](#), [has\\_expression](#), and [props](#).

### 6.5.5.2 `add_bool()`

```
void GiNaC::archive_node::add_bool (
 const std::string & name,
 bool value)
```

Add property of type "bool" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_BOOL](#), and [value](#).

### 6.5.5.3 `add_unsigned()`

```
void GiNaC::archive_node::add_unsigned (
 const std::string & name,
 unsigned value)
```

Add property of type "unsigned int" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_UNSIGNED](#), and [value](#).

#### 6.5.5.4 add\_string()

```
void GiNaC::archive_node::add_string (
 const std::string & name,
 const std::string & value)
```

Add property of type "string" to node.

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_STRING](#), and [value](#).

#### 6.5.5.5 add\_ex()

```
void GiNaC::archive_node::add_ex (
 const std::string & name,
 const ex & value)
```

Add property of type "ex" to node.

References [a](#), [GiNaC::archive::add\\_node\(\)](#), [archive\\_node\(\)](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_NODE](#), and [value](#).

Referenced by [GiNaC::container< C >::archive\(\)](#).

#### 6.5.5.6 find\_bool()

```
bool GiNaC::archive_node::find_bool (
 const std::string & name,
 bool & ret,
 unsigned index = 0) const
```

Retrieve property of type "bool" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), and [PTYPE\\_BOOL](#).

#### 6.5.5.7 find\_unsigned()

```
bool GiNaC::archive_node::find_unsigned (
 const std::string & name,
 unsigned & ret,
 unsigned index = 0) const
```

Retrieve property of type "unsigned" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), and [PTYPE\\_UNSIGNED](#).

#### 6.5.5.8 find\_string()

```
bool GiNaC::archive_node::find_string (
 const std::string & name,
 std::string & ret,
 unsigned index = 0) const
```

Retrieve property of type "string" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [props](#), [PTYPE\\_STRING](#), and [GiNaC::archive::unatomize\(\)](#).

Referenced by [unarchive\(\)](#).

#### 6.5.5.9 find\_first()

```
archive_node::archive_node_cit GiNaC::archive_node::find_first (
 const std::string & name) const
```

Find the location in the vector of properties of the first/last property with a given name.

References [a](#), [GiNaC::archive::atomize\(\)](#), and [props](#).

#### 6.5.5.10 find\_last()

```
archive_node::archive_node_cit GiNaC::archive_node::find_last (
 const std::string & name) const
```

References [a](#), [GiNaC::archive::atomize\(\)](#), and [props](#).

#### 6.5.5.11 find\_property\_range()

```
archive_node::archive_node_cit_range GiNaC::archive_node::find_property_range (
 const std::string & name1,
 const std::string & name2) const
```

Find a range of locations in the vector of properties.

The result begins at the first property with name1 and ends one past the last property with name2.

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::begin](#), [GiNaC::archive\\_node::archive\\_node\\_cit\\_range::end](#), and [props](#).

#### 6.5.5.12 find\_ex()

```
bool GiNaC::archive_node::find_ex (
 const std::string & name,
 ex & ret,
 lst & sym_lst,
 unsigned index = 0) const
```

Retrieve property of type "ex" from node.

##### Returns

"true" if property was found, "false" otherwise

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive::get\\_node\(\)](#), [props](#), [PTYPE\\_NODE](#), and [unarchive\(\)](#).

#### 6.5.5.13 find\_ex\_by\_loc()

```
void GiNaC::archive_node::find_ex_by_loc (
 archive_node_cit loc,
 ex & ret,
 lst & sym_lst) const
```

Retrieve property of type "ex" from the node if it is known that this node in fact contains such a property at the given location.

This is much more efficient than the preceding function.

References [a](#), [GiNaC::archive::get\\_node\(\)](#), and [unarchive\(\)](#).

#### 6.5.5.14 find\_ex\_node()

```
const archive_node & GiNaC::archive_node::find_ex_node (
 const std::string & name,
 unsigned index = 0) const
```

Retrieve property of type "ex" from node, returning the node of the sub-expression.

References [a](#), [GiNaC::archive::atomize\(\)](#), [GiNaC::archive::get\\_node\(\)](#), [props](#), and [PTYPE\\_NODE](#).

#### 6.5.5.15 get\_properties()

```
void GiNaC::archive_node::get_properties (
 propinfovector & v) const
```

Return vector of properties stored in node.

References [a](#), [props](#), and [GiNaC::archive::unatomize\(\)](#).

#### 6.5.5.16 unarchive()

```
ex GiNaC::archive_node::unarchive (
 lst & sym_lst) const
```

Convert archive node to [GiNaC](#) expression.

References [GiNaC::status\\_flags::dynamallocated](#), [e](#), [GiNaC::find\\_factory\\_fcn\(\)](#), [find\\_string\(\)](#), and [has\\_expression](#).

Referenced by [find\\_ex\(\)](#), and [find\\_ex\\_by\\_loc\(\)](#).

#### 6.5.5.17 has\_same\_ex\_as()

```
bool GiNaC::archive_node::has_same_ex_as (
 const archive_node & other) const
```

Check if the [archive\\_node](#) stores the same expression as another [archive\\_node](#).

##### Returns

"true" if expressions are the same

References [GiNaC::ex::bp](#), [e](#), and [has\\_expression](#).

#### 6.5.5.18 has\_ex()

```
bool GiNaC::archive_node::has_ex () const [inline]
```

References [has\\_expression](#).

#### 6.5.5.19 get\_ex()

```
ex GiNaC::archive_node::get_ex () const [inline]
```

References [e](#).

#### 6.5.5.20 forget()

```
void GiNaC::archive_node::forget ()
```

Delete cached unarchived expressions from node (for debugging).

References [e](#), and [has\\_expression](#).

Referenced by [GiNaC::archive::forget\(\)](#).

### 6.5.5.21 printraw()

```
void GiNaC::archive_node::printraw (
 std::ostream & os) const
```

Output [archive\\_node](#) to stream in ugly raw format (for debugging).

References [a](#), [GiNaC::ex::begin\(\)](#), [GiNaC::ex::bp](#), [e](#), [has\\_expression](#), [props](#), [PTYPE\\_BOOL](#), [PTYPE\\_NODE](#), [PTYPE\\_STRING](#), [PTYPE\\_UNSIGNED](#), and [GiNaC::archive::unatomize\(\)](#).

## 6.5.6 Friends And Related Symbol Documentation

### 6.5.6.1 operator<<

```
std::ostream & operator<< (
 std::ostream & os,
 const archive_node & ar) [friend]
```

Write [archive\\_node](#) to binary data stream.

### 6.5.6.2 operator>>

```
std::istream & operator>> (
 std::istream & is,
 archive_node & ar) [friend]
```

Read [archive\\_node](#) from binary data stream.

## 6.5.7 Member Data Documentation

### 6.5.7.1 a

```
archive& GiNaC::archive_node::a [private]
```

Reference to the archive to which this node belongs.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_by\\_loc\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), and [printraw\(\)](#).

### 6.5.7.2 props

```
std::vector<property> GiNaC::archive_node::props [private]
```

Vector of stored properties.

Referenced by [add\\_bool\(\)](#), [add\\_ex\(\)](#), [add\\_string\(\)](#), [add\\_unsigned\(\)](#), [find\\_bool\(\)](#), [find\\_ex\(\)](#), [find\\_ex\\_node\(\)](#), [find\\_first\(\)](#), [find\\_last\(\)](#), [find\\_property\\_range\(\)](#), [find\\_string\(\)](#), [find\\_unsigned\(\)](#), [get\\_properties\(\)](#), [operator=\(\)](#), and [printraw\(\)](#).

### 6.5.7.3 has\_expression

```
bool GiNaC::archive_node::has_expression [mutable], [private]
```

Flag indicating whether a cached unarchived representation of this node exists.

Referenced by [forget\(\)](#), [has\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [printraw\(\)](#), and [unarchive\(\)](#).

### 6.5.7.4 e

```
ex GiNaC::archive_node::e [mutable], [private]
```

The cached unarchived representation of this node (if any).

Referenced by [forget\(\)](#), [get\\_ex\(\)](#), [has\\_same\\_ex\\_as\(\)](#), [operator=\(\)](#), [printraw\(\)](#), and [unarchive\(\)](#).

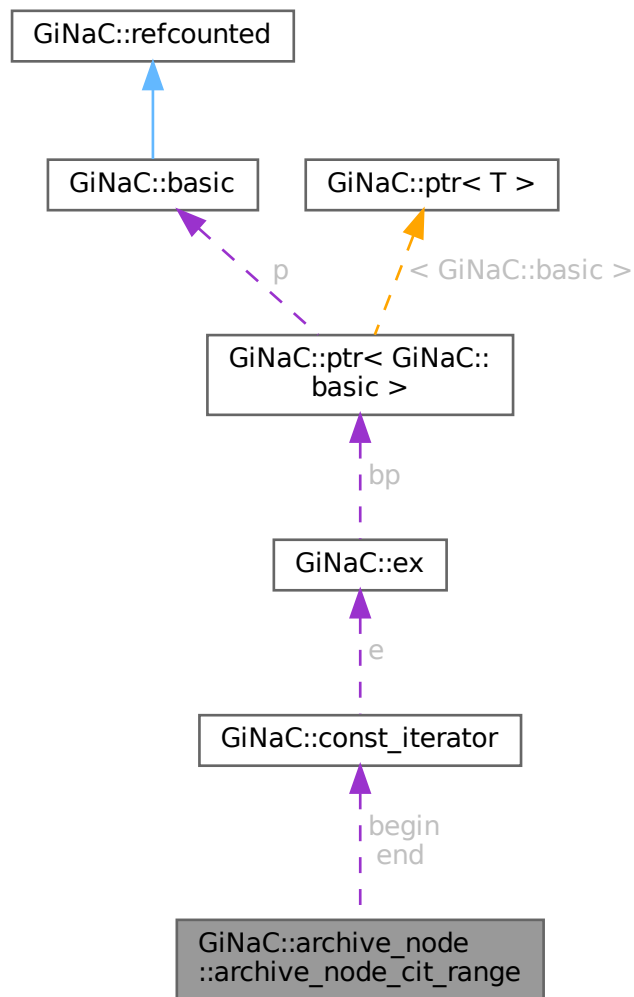
The documentation for this class was generated from the following files:

- [archive.h](#)
- [archive.cpp](#)

## 6.6 GiNaC::archive\_node::archive\_node\_cit\_range Struct Reference

```
#include <archive.h>
```

Collaboration diagram for GiNaC::archive\_node::archive\_node\_cit\_range:



## Public Attributes

- [archive\\_node\\_cit begin](#)
- [archive\\_node\\_cit end](#)

## 6.6.1 Member Data Documentation

### 6.6.1.1 begin

[archive\\_node\\_cit](#) `GiNaC::archive_node::archive_node_cit_range::begin`

Referenced by [GiNaC::archive\\_node::find\\_property\\_range\(\)](#).

### 6.6.1.2 end

`archive_node_cit` `GiNaC::archive_node::archive_node_cit_range::end`

Referenced by `GiNaC::archive_node::find_property_range()`.

The documentation for this struct was generated from the following file:

- `archive.h`

## 6.7 GiNaC::archive::archived\_ex Struct Reference

Archived expression descriptor.

### Public Member Functions

- `archived_ex` ()
- `archived_ex` (`archive_atom` `n`, `archive_node_id` `node`)

### Public Attributes

- `archive_atom` `name`  
*Name of expression.*
- `archive_node_id` `root`  
*ID of root node.*

### 6.7.1 Detailed Description

Archived expression descriptor.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 archived\_ex() [1/2]

`GiNaC::archive::archived_ex::archived_ex ( )` [inline]

#### 6.7.2.2 archived\_ex() [2/2]

`GiNaC::archive::archived_ex::archived_ex (`  
`archive_atom n,`  
`archive_node_id node )` [inline]

## 6.7.3 Member Data Documentation

### 6.7.3.1 name

`archive_atom` GiNaC::archive::archived\_ex::name

Name of expression.

### 6.7.3.2 root

`archive_node_id` GiNaC::archive::archived\_ex::root

ID of root node.

The documentation for this struct was generated from the following file:

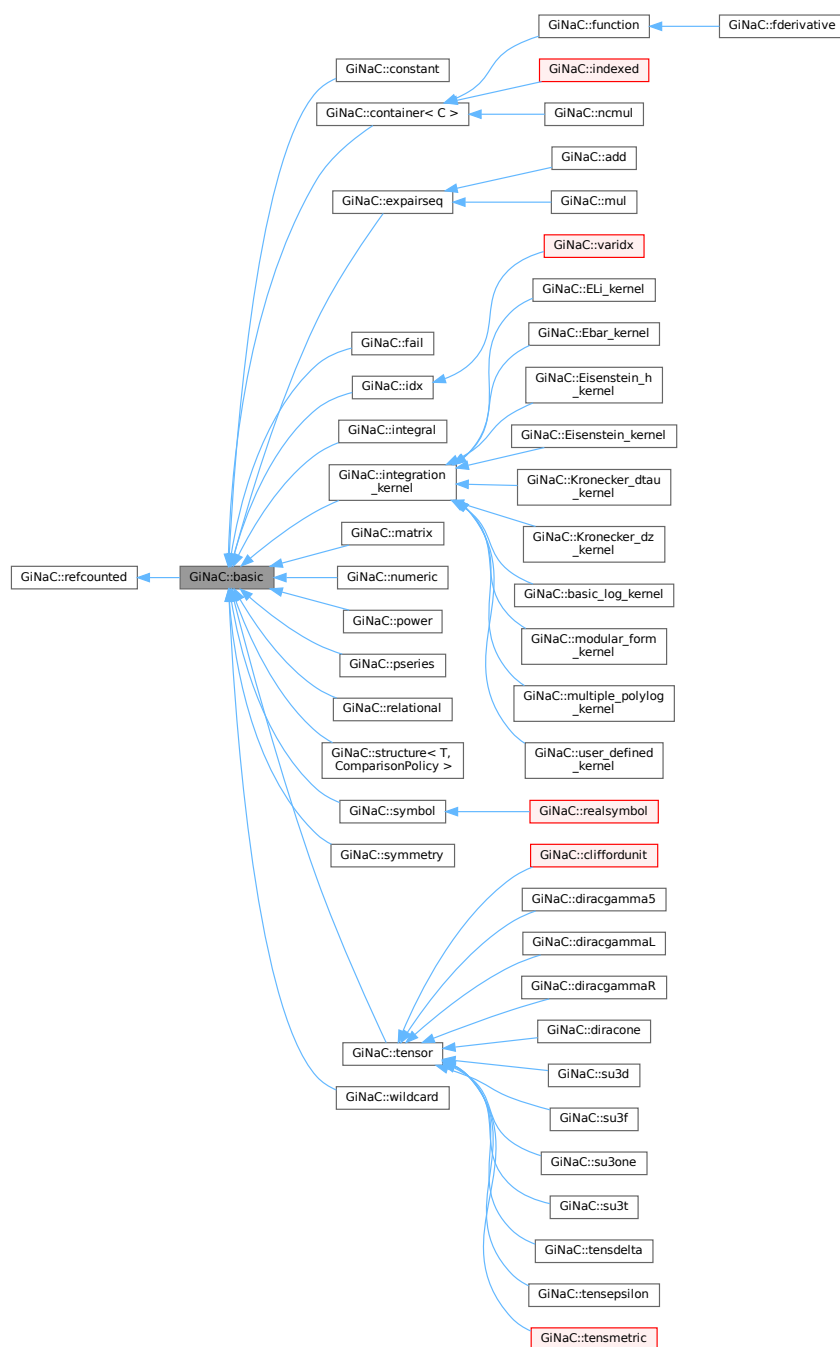
- [archive.h](#)

## 6.8 GiNaC::basic Class Reference

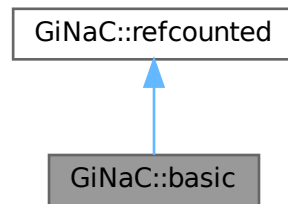
This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

```
#include <basic.h>
```

Inheritance diagram for GiNaC::basic:



Collaboration diagram for GiNaC::basic:



## Public Member Functions

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position `i`.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`

- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const

- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `virtual void archive (archive_node &n) const`  
*Save (serialize) the object into archive node.*
- `virtual void read_archive (const archive_node &n, lst &syms)`  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

## Protected Member Functions

- `basic ()`
- `virtual ex eval_ncmul (const exvector &v) const`
- `virtual bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative (const symbol &s) const`  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type (const basic &other) const`  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash () const`  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void ensure_if_modifiable () const`  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print (const print_context &c, unsigned level) const`  
*Default output to stream.*
- `void do_print_tree (const print_tree &c, unsigned level) const`  
*Tree output to stream.*
- `void do_print_python_repr (const print_python_repr &c, unsigned level) const`  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Friends

- class [ex](#)

## 6.8.1 Detailed Description

This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.

## 6.8.2 Constructor & Destructor Documentation

### 6.8.2.1 `basic()` [1/2]

```
GiNaC::basic::basic () [inline], [protected]
```

Referenced by [duplicate\(\)](#).

### 6.8.2.2 `~basic()`

```
virtual GiNaC::basic::~~basic () [inline], [virtual]
```

`basic` destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.

References [GiNaC::status\\_flags::dynallocated](#), [flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), and [GINAC\\_ASSERT](#).

### 6.8.2.3 `basic()` [2/2]

```
GiNaC::basic::basic (
 const basic & other)
```

## 6.8.3 Member Function Documentation

### 6.8.3.1 `operator=()`

```
const basic & GiNaC::basic::operator= (
 const basic & other)
```

`basic` assignment operator: the other object might be of a derived class.

References [flags](#), and [hashvalue](#).

### 6.8.3.2 duplicate()

```
virtual basic * GiNaC::basic::duplicate () const [inline], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented in [GiNaC::realsymbol](#), and [GiNaC::possymbol](#).

References [basic\(\)](#), [GiNaC::status\\_flags::dynallocated](#), and [setflag\(\)](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::spinidx::toggle\\_dot\(\)](#), [GiNaC::varidx::toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

### 6.8.3.3 eval()

```
ex GiNaC::basic::eval () const [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#).

### 6.8.3.4 evalf()

```
ex GiNaC::basic::evalf () const [virtual]
```

Evaluate object numerically.

Reimplemented in [GiNaC::constant](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::symbol](#).

References [GiNaC::nops\(\)](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.8.3.5 evalm()

```
ex GiNaC::basic::evalm () const [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented in [GiNaC::add](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy>](#).

References [GiNaC::map\\_evalm](#), and [GiNaC::nops\(\)](#).

### 6.8.3.6 eval\_integ()

```
ex GiNaC::basic::eval_integ () const [virtual]
```

Evaluate integrals, if result is known.

Reimplemented in [GiNaC::integral](#), and [GiNaC::pseries](#).

References [GiNaC::map\\_eval\\_integ](#), and [GiNaC::nops\(\)](#).

### 6.8.3.7 eval\_ncmul()

```
ex GiNaC::basic::eval_ncmul (
 const exvector & v) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::hold\\_ncmul\(\)](#).

### 6.8.3.8 eval\_indexed()

```
ex GiNaC::basic::eval_indexed (
 const basic & i) const [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented in [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::matrix](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).

### 6.8.3.9 print()

```
void GiNaC::basic::print (
 const print_context & c,
 unsigned level = 0) const [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

#### Parameters

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| <i>c</i>     | print context object that describes the output formatting                                                 |
| <i>level</i> | value that is used to identify the precedence or indentation level for placing parentheses and formatting |

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [c](#).

Referenced by [GiNaC::fderivative::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), and [GiNaC::pseries::print\\_series\(\)](#).

### 6.8.3.10 dbgprint()

```
void GiNaC::basic::dbgprint () const [virtual]
```

Little wrapper around print to be called within a debugger.

This is needed because you cannot call `foo.print(cout)` from within the debugger because it might not know what `cout` is. This method can be invoked with no argument and it will simply print to `stdout`.

See also

[basic::print](#)  
[basic::dbgprinttree](#)

### 6.8.3.11 dbgprinttree()

```
void GiNaC::basic::dbgprinttree () const [virtual]
```

Little wrapper around printtree to be called within a debugger.

See also

[basic::dbgprint](#)

### 6.8.3.12 precedence()

```
unsigned GiNaC::basic::precedence () const [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.13 info()

```
bool GiNaC::basic::info (
 unsigned inf) const [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).

Referenced by [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::function::info\(\)](#), and [GiNaC::matrix::solve\(\)](#).

### 6.8.3.14 nops()

```
size_t GiNaC::basic::nops () const [virtual]
```

Number of operands/members.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::H\\_eval\(\)](#), and [normal\(\)](#).

### 6.8.3.15 op()

```
ex GiNaC::basic::op (
 size_t i) const [virtual]
```

Return operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), and [GiNaC::tensepsilon::eval\\_indexed\(\)](#).

### 6.8.3.16 operator[]() [1/4]

```
ex GiNaC::basic::operator[] (
 const ex & index) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#), and [GiNaC::to\\_int\(\)](#).

### 6.8.3.17 operator[]() [2/4]

```
ex GiNaC::basic::operator[] (
 size_t i) const [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::op\(\)](#).

**6.8.3.18 let\_op()**

```
ex & GiNaC::basic::let_op (
 size_t i) [virtual]
```

Return modifiable operand/member at position i.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::integral](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::user\\_defined\\_kernel](#), [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

Referenced by [map\(\)](#), and [subs\(\)](#).

**6.8.3.19 operator[]() [3/4]**

```
ex & GiNaC::basic::operator[] (
 const ex & index) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::to\\_int\(\)](#).

**6.8.3.20 operator[]() [4/4]**

```
ex & GiNaC::basic::operator[] (
 size_t i) [virtual]
```

Reimplemented in [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.21 has()**

```
bool GiNaC::basic::has (
 const ex & pattern,
 unsigned options = 0) const [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented in [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::ex::has\(\)](#), [GiNaC::match\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [options](#).

Referenced by [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), and [series\(\)](#).

### 6.8.3.22 match()

```
bool GiNaC::basic::match (
 const ex & pattern,
 exmap & repl_lst) const [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::wildcard](#).

References [GiNaC::ex::match\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::ex::op\(\)](#).

### 6.8.3.23 match\_same\_type()

```
bool GiNaC::basic::match_same_type (
 const basic & other) const [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::matrix](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.24 subs()

```
ex GiNaC::basic::subs (
 const exmap & m,
 unsigned options = 0) const [virtual]
```

Substitute a set of objects by arbitrary expressions.

The *ex* returned will already be evaluated.

Reimplemented in [GiNaC::clifford](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), [m](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), [options](#), [GiNaC::ex::subs\(\)](#), and [subs\\_one\\_level\(\)](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**6.8.3.25 map()**

```
ex GiNaC::basic::map (
 map_function & f) const [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented in [GiNaC::expairseq](#), [GiNaC::idx](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [clearflag\(\)](#), [let\\_op\(\)](#), [n](#), [GiNaC::nops\(\)](#), and [GiNaC::op\(\)](#).

Referenced by [normal\(\)](#).

**6.8.3.26 accept()**

```
virtual void GiNaC::basic::accept (
 GiNaC::visitor & v) const [inline], [virtual]
```

**6.8.3.27 is\_polynomial()**

```
bool GiNaC::basic::is_polynomial (
 const ex & var) const [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), and [GiNaC::symbol](#).

References [GiNaC::has\(\)](#).

**6.8.3.28 degree()**

```
int GiNaC::basic::degree (
 const ex & s) const [virtual]
```

Return degree of highest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

**6.8.3.29 ldegree()**

```
int GiNaC::basic::ldegree (
 const ex & s) const [virtual]
```

Return degree of lowest power in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

### 6.8.3.30 `coeff()`

```
ex GiNaC::basic::coeff (
 const ex & s,
 int n = 1) const [virtual]
```

Return coefficient of degree n in object s.

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and [n](#).

Referenced by [GiNaC::expairseq::read\\_archive\(\)](#), [series\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

### 6.8.3.31 `expand()`

```
ex GiNaC::basic::expand (
 unsigned options = 0) const [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented in [GiNaC::add](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::nops\(\)](#), and [options](#).

Referenced by [GiNaC::power::expand\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

### 6.8.3.32 `collect()`

```
ex GiNaC::basic::collect (
 const ex & s,
 bool distributed = false) const [virtual]
```

Sort expanded expression in terms of powers of some object(s).

#### Parameters

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <i>s</i>           | object(s) to sort in                                       |
| <i>distributed</i> | recursive or distributed form (only used when s is a list) |

Reimplemented in [GiNaC::pseries](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::collect\(\)](#), [GiNaC::ex::degree\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::ldegree\(\)](#), [n](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::pow\(\)](#), and [x](#).

### 6.8.3.33 `derivative()`

```
ex GiNaC::basic::derivative (
```

```
const symbol & s) const [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), and [GiNaC::nops\(\)](#).

#### 6.8.3.34 series()

```
ex GiNaC::basic::series (
 const relational & r,
 int order,
 unsigned options = 0) const [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented in [GiNaC::add](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::integral](#), [GiNaC::integration\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), [GiNaC::mul](#), [GiNaC::power](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::ex::expand\(\)](#), [has\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [order](#), [r](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::lgamma\\_series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), and [GiNaC::power::series\(\)](#).

#### 6.8.3.35 normal()

```
ex GiNaC::basic::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::structure< T, ComparisonPolicy>](#), and [GiNaC::symbol](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::info\(\)](#), [map\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [nops\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), and [GiNaC::matrix::solve\(\)](#).

### 6.8.3.36 to\_rational()

```
ex GiNaC::basic::to_rational (
 exmap & repl) const [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

### 6.8.3.37 to\_polynomial()

```
ex GiNaC::basic::to_polynomial (
 exmap & repl) const [virtual]
```

Reimplemented in [GiNaC::expairseq](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

References [GiNaC::replace\\_with\\_symbol\(\)](#).

### 6.8.3.38 integer\_content()

```
numeric GiNaC::basic::integer_content () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

### 6.8.3.39 smod()

```
ex GiNaC::basic::smod (
 const numeric & xi) const [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

#### Parameters

|      |         |
|------|---------|
| $xi$ | modulus |
|------|---------|

#### Returns

mapped polynomial

See also

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

#### 6.8.3.40 max\_coefficient()

```
numeric GiNaC::basic::max_coefficient () const [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented in [GiNaC::add](#), [GiNaC::mul](#), [GiNaC::numeric](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::\\_num1\\_p](#).

#### 6.8.3.41 get\_free\_indices()

```
exvector GiNaC::basic::get_free_indices () const [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented in [GiNaC::add](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

#### 6.8.3.42 add\_indexed()

```
ex GiNaC::basic::add_indexed (
 const ex & self,
 const ex & other) const [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class `indexed` (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

Parameters

|              |                                                                 |
|--------------|-----------------------------------------------------------------|
| <i>self</i>  | First indexed expression; its base object is <code>*this</code> |
| <i>other</i> | Second indexed expression                                       |

Returns

sum of `self` and `other`

See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

#### 6.8.3.43 scalar\_mul\_indexed()

```
ex GiNaC::basic::scalar_mul_indexed (
 const ex & self,
 const numeric & other) const [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

Parameters

|              |                                              |
|--------------|----------------------------------------------|
| <i>self</i>  | Indexed expression; its base object is *this |
| <i>other</i> | Numeric value                                |

Returns

product of self and other

See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::matrix](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

#### 6.8.3.44 contract\_with()

```
bool GiNaC::basic::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

Parameters

|              |                                                               |
|--------------|---------------------------------------------------------------|
| <i>self</i>  | Pointer to first indexed expression; its base object is *this |
| <i>other</i> | Pointer to second indexed expression                          |
| <i>v</i>     | The complete vector of factors                                |

**Returns**

true if the contraction was successful, false otherwise

**See also**

[ex::simplify\\_indexed\(\)](#)

Reimplemented in [GiNaC::cliffordunit](#), [GiNaC::diracgamma](#), [GiNaC::su3t](#), [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::matrix](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::spinmetric](#), and [GiNaC::tensepsilon](#).

**6.8.3.45 return\_type()**

```
unsigned GiNaC::basic::return_type () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::su3f](#), [GiNaC::su3d](#), [GiNaC::expairseq](#), [GiNaC::fail](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::tensor](#), [GiNaC::tensdelta](#), [GiNaC::tensmetric](#), [GiNaC::minkmetric](#), and [GiNaC::tensepsilon](#).

**6.8.3.46 return\_type\_tinfo()**

```
return_type_t GiNaC::basic::return_type_tinfo () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::power](#), [GiNaC::relational](#), and [GiNaC::structure< T, ComparisonPolicy >](#).

References [GiNaC::return\\_type\\_t::rl](#), and [GiNaC::return\\_type\\_t::tinfo](#).

**6.8.3.47 conjugate()**

```
ex GiNaC::basic::conjugate () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::diracgamma5](#), [GiNaC::diracgammaL](#), [GiNaC::diracgammaR](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::spinidx](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

**6.8.3.48 real\_part()**

```
ex GiNaC::basic::real_part () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

Referenced by [GiNaC::function::real\\_part\(\)](#), and [GiNaC::ncmul::real\\_part\(\)](#).

### 6.8.3.49 `imag_part()`

```
ex GiNaC::basic::imag_part () const [virtual]
```

Reimplemented in [GiNaC::add](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::function](#), [GiNaC::indexed](#), [GiNaC::matrix](#), [GiNaC::mul](#), [GiNaC::ncmul](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::symbol](#), and [GiNaC::realsymbol](#).

Referenced by [GiNaC::function::imag\\_part\(\)](#), and [GiNaC::ncmul::imag\\_part\(\)](#).

### 6.8.3.50 `compare_same_type()`

```
int GiNaC::basic::compare_same_type (
 const basic & other) const [protected], [virtual]
```

Returns order relation between two objects of same type.

This needs to be implemented by each class. It may never return anything else than 0, signalling equality, or +1 and -1 signalling inequality and determining the canonical ordering. (Perl hackers will wonder why C++ doesn't feature the spaceship operator `<=>` for denoting just this.)

References [GiNaC::compare\\_pointers\(\)](#).

Referenced by [GiNaC::symbol::derivative\(\)](#).

### 6.8.3.51 `is_equal_same_type()`

```
bool GiNaC::basic::is_equal_same_type (
 const basic & other) const [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented in [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::numeric](#), [GiNaC::structure< T, ComparisonPolicy >](#), and [GiNaC::symbol](#).

### 6.8.3.52 `calchash()`

```
unsigned GiNaC::basic::calchash () const [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented in [GiNaC::constant](#), [GiNaC::expairseq](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::numeric](#), [GiNaC::relational](#), [GiNaC::structure< T, ComparisonPolicy >](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), and [GiNaC::wildcard](#).

References [GiNaC::ex::gethash\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::rotate\\_left\(\)](#).

Referenced by [gethash\(\)](#).

**6.8.3.53 print\_dispatch()** [1/2]

```
template<class T >
void GiNaC::basic::print_dispatch (
 const print_context & c,
 unsigned level) const [inline]
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [c](#), and [print\\_dispatch\(\)](#).

Referenced by [print\\_dispatch\(\)](#).

**6.8.3.54 print\_dispatch()** [2/2]

```
void GiNaC::basic::print_dispatch (
 const registered_class_info & ri,
 const print_context & c,
 unsigned level) const
```

Like [print\(\)](#), but dispatch to the specified class.

Can be used by implementations of print methods to dispatch to the method of the superclass.

See also

[basic::print](#)

References [c](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), and [GiNaC::class\\_info< OPT >::options](#).

**6.8.3.55 archive()**

```
void GiNaC::basic::archive (
 archive_node & n) const [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::constant](#), [GiNaC::container< C >](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::minkmetric](#), [GiNaC::tensepsilon](#), and [GiNaC::wildcard](#).

References [n](#).

### 6.8.3.56 read\_archive()

```
void GiNaC::basic::read_archive (
 const archive_node & n,
 lst & syms) [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented in [GiNaC::clifford](#), [GiNaC::color](#), [GiNaC::container< C >](#), [GiNaC::constant](#), [GiNaC::expairseq](#), [GiNaC::fderivative](#), [GiNaC::function](#), [GiNaC::idx](#), [GiNaC::varidx](#), [GiNaC::spinidx](#), [GiNaC::indexed](#), [GiNaC::integral](#), [GiNaC::matrix](#), [GiNaC::numeric](#), [GiNaC::power](#), [GiNaC::pseries](#), [GiNaC::relational](#), [GiNaC::symbol](#), [GiNaC::symmetry](#), [GiNaC::minkmetric](#), [GiNaC::tensepsilon](#), and [GiNaC::wildcard](#).

### 6.8.3.57 subs\_one\_level()

```
ex GiNaC::basic::subs_one_level (
 const exmap & m,
 unsigned options) const
```

Helper function for [subs\(\)](#).

Does not recurse into subexpressions.

References [GiNaC::ex::find\(\)](#), [m](#), [GiNaC::match\(\)](#), [options](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::relational::subs\(\)](#), and [GiNaC::symbol::subs\(\)](#).

### 6.8.3.58 diff()

```
ex GiNaC::basic::diff (
 const symbol & s,
 unsigned nth = 1) const
```

Default interface of nth derivative [ex::diff\(s, n\)](#).

It should be called instead of [::derivative\(s\)](#) for first derivatives and for nth derivatives it just recurses down.

#### Parameters

|            |                            |
|------------|----------------------------|
| <i>s</i>   | symbol to differentiate in |
| <i>nth</i> | order of differentiation   |

See also

[ex::diff](#)

References [GiNaC::ex::diff\(\)](#), and [GiNaC::ex::is\\_zero\(\)](#).

Referenced by [GiNaC::mul::derivative\(\)](#).

### 6.8.3.59 compare()

```
int GiNaC::basic::compare (
 const basic & other) const
```

Compare objects syntactically to establish canonical ordering.

All compare functions return: -1 for \*this less than other, 0 equal, 1 greater.

References [gethash\(\)](#).

### 6.8.3.60 is\_equal()

```
bool GiNaC::basic::is_equal (
 const basic & other) const
```

Test for syntactic equality.

This is only a quick test, meaning objects should be in the same domain. You might have to [.expand\(\)](#), [.normal\(\)](#) objects first, depending on the domain of your computation, to get a more reliable answer.

See also

[is\\_equal\\_same\\_type](#)

References [gethash\(\)](#).

Referenced by [GiNaC::ncmul::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), and [GiNaC::wildcard::match\(\)](#).

### 6.8.3.61 hold()

```
const basic & GiNaC::basic::hold () const
```

Stop further evaluation.



**6.8.3.64 clearflag()**

```
const basic & GiNaC::basic::clearflag (
 unsigned f) const [inline]
```

Clear some [status\\_flags](#).

References [flags](#).

Referenced by [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::function::function\(\)](#), [GiNaC::expairseq::info\(\)](#), [GiNaC::power::info\(\)](#), [map\(\)](#), [GiNaC::idx::map\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::spinidx::toggle\\_dot\(\)](#), [GiNaC::varidx::toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

**6.8.3.65 ensure\_if\_modifiable()**

```
void GiNaC::basic::ensure_if_modifiable () const [protected]
```

Ensure the object may be modified without hurting others, throws if this is not the case.

Referenced by [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::clifford::let\\_op\(\)](#), [GiNaC::integral::let\\_op\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::let\\_op\(\)](#), [GiNaC::ELi\\_kernel::let\\_op\(\)](#), [GiNaC::Ebar\\_kernel::let\\_op\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::let\\_op\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_kernel::let\\_op\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::let\\_op\(\)](#), [GiNaC::modular\\_form\\_kernel::let\\_op\(\)](#), [GiNaC::user\\_defined\\_kernel::let\\_op\(\)](#), [GiNaC::matrix::let\\_op\(\)](#), [GiNaC::matrix::operator\(\)](#), and [GiNaC::matrix::pivot\(\)](#).

**6.8.3.66 do\_print()**

```
void GiNaC::basic::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

Default output to stream.

References [c](#).

**6.8.3.67 do\_print\_tree()**

```
void GiNaC::basic::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

Tree output to stream.

References [c](#), [GiNaC::nops\(\)](#), [GiNaC::op\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.8.3.68 do\_print\_python\_repr()**

```
void GiNaC::basic::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

Python parsable output to stream.

References [c](#).

## 6.8.4 Friends And Related Symbol Documentation

### 6.8.4.1 ex

```
friend class ex [friend]
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::scalar\\_mul\\_indexed\(\)](#).

## 6.8.5 Member Data Documentation

### 6.8.5.1 flags

```
unsigned GiNaC::basic::flags [mutable], [protected]
```

of type [status\\_flags](#)

Referenced by [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [clearflag\(\)](#), [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::expairseq::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [gethash\(\)](#), [GiNaC::expairseq::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [operator=\(\)](#), [GiNaC::function::print\(\)](#), [setflag\(\)](#), and [~basic\(\)](#).

### 6.8.5.2 hashvalue

```
unsigned GiNaC::basic::hashvalue [mutable], [protected]
```

hash value

Referenced by [GiNaC::constant::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), [GiNaC::numeric::calchash\(\)](#), [GiNaC::relational::calchash\(\)](#), [GiNaC::symbol::calchash\(\)](#), [GiNaC::symmetry::calchash\(\)](#), [GiNaC::wildcard::calchash\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [gethash\(\)](#), [operator=\(\)](#), and [GiNaC::function::print\(\)](#).

The documentation for this class was generated from the following files:

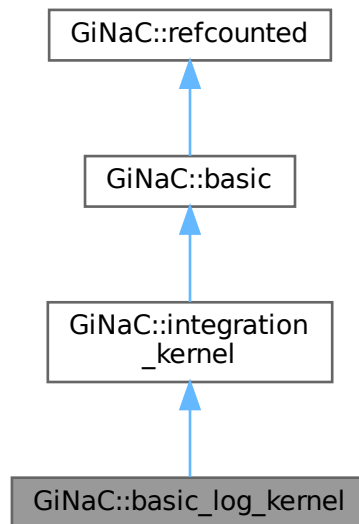
- [basic.h](#)
- [basic.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.9 GiNaC::basic\_log\_kernel Class Reference

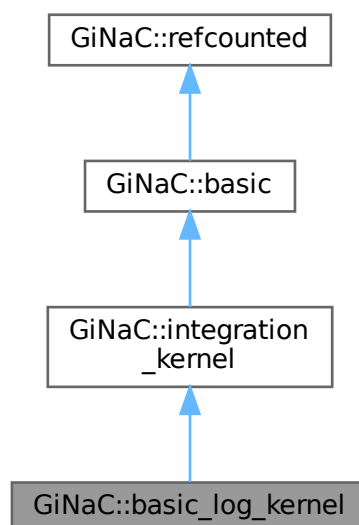
The basic integration kernel with a logarithmic singularity at the origin.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::basic\_log\_kernel:



Collaboration diagram for GiNaC::basic\_log\_kernel:



### Protected Member Functions

- `cl::cl_N_series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool `is_numeric` (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual `ex Laurent_series` (const `ex` &x, int order) const

- Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N `series_coeff` (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)

- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const

- *Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &[syms](#))
- *Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const
- *Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const
- *Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const
- *Test for syntactic equality.*
- const [basic](#) & [hold](#) () const
- *Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const
- *Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const
- *Clear some [status\\_flags](#).*

### Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.9.1 Detailed Description

The basic integration kernel with a logarithmic singularity at the origin.

This class represents the differential one-form

$$L_0 = \frac{d\lambda}{\lambda}$$

## 6.9.2 Member Function Documentation

### 6.9.2.1 series\_coeff\_impl()

```
cln::cl_N GiNaC::basic_log_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.9.2.2 do\_print()

```
void GiNaC::basic_log_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#).

The documentation for this class was generated from the following files:

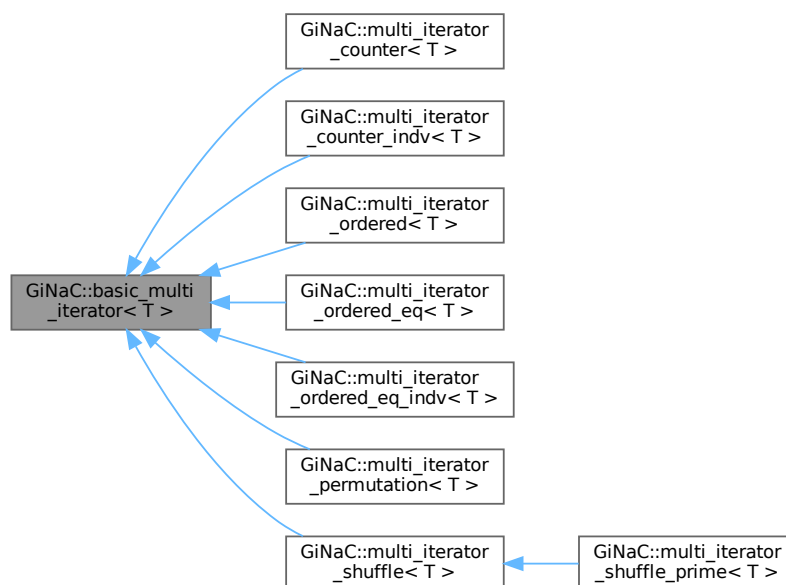
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.10 GiNaC::basic\_multi\_iterator< T > Class Template Reference

[basic\\_multi\\_iterator](#) is a base class.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::basic\_multi\_iterator< T >:



## Public Member Functions

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit N, lower limit B and size k.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*
- virtual [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- virtual [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*No effect for basic\_multi\_iterator.*

## Protected Attributes

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [basic\\_multi\\_iterator](#)< TT > &v)

### 6.10.1 Detailed Description

```
template<class T>
class GiNaC::basic_multi_iterator< T >
```

[basic\\_multi\\_iterator](#) is a base class.

The base class itself does not do anything useful. A typical use of a class derived from [basic\\_multi\\_iterator](#) is

```
multi_iterator_ordered<int> k(0,4,2);

for(k.init(); !k.overflow(); k++) { std::cout << k << std::endl; }
```

which prints out

```
multi_iterator_ordered(0,1) multi_iterator_ordered(0,2) multi_iterator_ordered(0,3) multi_iterator_ordered(1,2)
multi_iterator_ordered(1,3) multi_iterator_ordered(2,3)
```

Individual components of k can be accessed with k[i] or k(i).

All classes derived from [basic\\_multi\\_iterator](#) follow the same syntax.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 basic\_multi\_iterator() [1/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
 void) [inline]
```

Default constructor.

#### 6.10.2.2 basic\_multi\_iterator() [2/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
 T B,
 T N,
 size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N, lower limit B and size k .

#### 6.10.2.3 basic\_multi\_iterator() [3/3]

```
template<class T >
GiNaC::basic_multi_iterator< T >::basic_multi_iterator (
 T B,
 T N,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

#### 6.10.2.4 ~basic\_multi\_iterator()

```
template<class T >
GiNaC::basic_multi_iterator< T >::~~basic_multi_iterator () [inline], [virtual]
```

Destructor.

### 6.10.3 Member Function Documentation

#### 6.10.3.1 size()

```
template<class T >
size_t GiNaC::basic_multi_iterator< T >::size (
 void) const [inline]
```

Returns the size of a multi\_iterator.

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::operator++\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), and [GiNaC::operator<<\(\)](#).

#### 6.10.3.2 overflow()

```
template<class T >
bool GiNaC::basic_multi_iterator< T >::overflow (
 void) const [inline]
```

Return the overflow flag.

#### 6.10.3.3 get\_vector()

```
template<class T >
const std::vector< T > & GiNaC::basic_multi_iterator< T >::get_vector (
 void) const [inline]
```

Returns a reference to the vector v.

#### 6.10.3.4 operator[]() [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator[] (
 size_t i) const [inline]
```

Subscription via [].

**6.10.3.5 operator[]()** [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator[] (
 size_t i) [inline]
```

Subscription via [].

**6.10.3.6 operator()()** [1/2]

```
template<class T >
T GiNaC::basic_multi_iterator< T >::operator() (
 size_t i) const [inline]
```

Subscription via ()

**6.10.3.7 operator()()** [2/2]

```
template<class T >
T & GiNaC::basic_multi_iterator< T >::operator() (
 size_t i) [inline]
```

Subscription via ()

**6.10.3.8 init()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented in [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), [GiNaC::multi\\_iterator\\_shuffle< T >](#), and [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

**6.10.3.9 operator++()**

```
template<class T >
basic_multi_iterator< T > & GiNaC::basic_multi_iterator< T >::operator++ (
 int) [inline], [virtual]
```

No effect for [basic\\_multi\\_iterator](#).

Reimplemented in [GiNaC::multi\\_iterator\\_ordered< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#), [GiNaC::multi\\_iterator\\_counter< T >](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#), [GiNaC::multi\\_iterator\\_permutation< T >](#), and [GiNaC::multi\\_iterator\\_shuffle< T >](#).

## 6.10.4 Friends And Related Symbol Documentation

### 6.10.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const basic_multi_iterator< TT > & v) [friend]
```

## 6.10.5 Member Data Documentation

### 6.10.5.1 N

```
template<class T >
T GiNaC::basic_multi_iterator< T >::N [protected]
```

### 6.10.5.2 B

```
template<class T >
T GiNaC::basic_multi_iterator< T >::B [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::operator<<\(\)](#), and [GiNaC::operator<<\(\)](#).

### 6.10.5.3 v

```
template<class T >
std::vector<T> GiNaC::basic_multi_iterator< T >::v [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

### 6.10.5.4 flag\_overflow

```
template<class T >
bool GiNaC::basic_multi_iterator< T >::flag_overflow [protected]
```

The documentation for this class was generated from the following file:

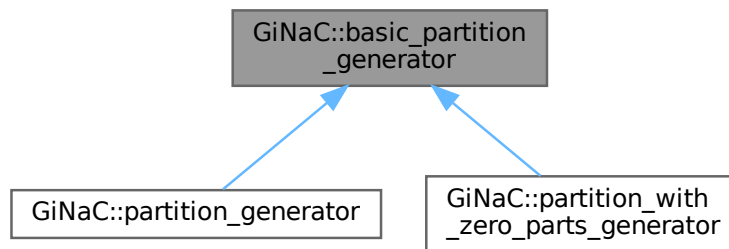
- [utils\\_multi\\_iterator.h](#)

## 6.11 GiNaC::basic\_partition\_generator Class Reference

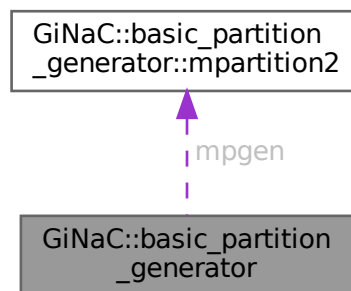
Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::basic\_partition\_generator:



Collaboration diagram for GiNaC::basic\_partition\_generator:



### Classes

- struct [mpartition2](#)

### Protected Member Functions

- [basic\\_partition\\_generator](#) (unsigned  $n$ , unsigned  $m$ )

**Protected Attributes**

- [mpartition2 mpgen](#)

**6.11.1 Detailed Description**

Base class for generating all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts in non-decreasing order.

**6.11.2 Constructor & Destructor Documentation****6.11.2.1 basic\_partition\_generator()**

```
GiNaC::basic_partition_generator::basic_partition_generator (
 unsigned n_,
 unsigned m_) [inline], [protected]
```

**6.11.3 Member Data Documentation****6.11.3.1 mpngen**

```
mpartition2 GiNaC::basic_partition_generator::mpngen [protected]
```

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#) and [GiNaC::partition\\_generator::next\(\)](#).

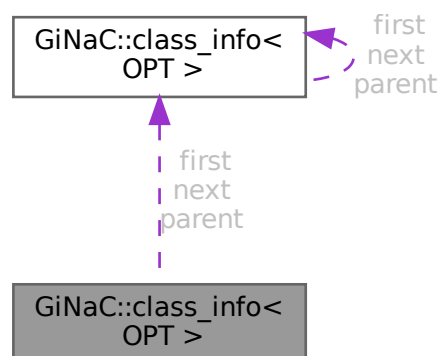
The documentation for this class was generated from the following file:

- [utils.h](#)

**6.12 GiNaC::class\_info< OPT > Class Template Reference**

```
#include <class_info.h>
```

Collaboration diagram for GiNaC::class\_info< OPT >:



## Classes

- struct [tree\\_node](#)

## Public Member Functions

- [class\\_info](#) (const OPT &o)
- [class\\_info](#) \* [get\\_parent](#) () const  
*Get pointer to [class\\_info](#) of parent class (or nullptr).*

## Static Public Member Functions

- static const [class\\_info](#) \* [find](#) (const std::string &class\_name)  
*Find [class\\_info](#) by name.*
- static void [dump\\_hierarchy](#) (bool verbose=false)  
*Dump class hierarchy to std::cout.*

## Public Attributes

- OPT [options](#)

## Static Private Member Functions

- static void [dump\\_tree](#) ([tree\\_node](#) \*n, const std::string &prefix, bool verbose)
- static void [identify\\_parents](#) ()

## Private Attributes

- [class\\_info](#) \* [next](#)
- [class\\_info](#) \* [parent](#)

## Static Private Attributes

- static [class\\_info](#) \* [first](#) = nullptr
- static bool [parents\\_identified](#) = false

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 [class\\_info](#)()

```
template<class OPT >
GiNaC::class_info< OPT >::class_info (
 const OPT & o) [inline]
```

References [GiNaC::class\\_info< OPT >::first](#), and [GiNaC::class\\_info< OPT >::parents\\_identified](#).

## 6.12.2 Member Function Documentation

### 6.12.2.1 get\_parent()

```
template<class OPT >
class_info * GiNaC::class_info< OPT >::get_parent () const [inline]
```

Get pointer to [class\\_info](#) of parent class (or nullptr).

References [GiNaC::class\\_info< OPT >::identify\\_parents\(\)](#), and [GiNaC::class\\_info< OPT >::parent](#).

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#), [GiNaC::function::print\(\)](#), and [GiNaC::basic::print\\_dispatch\(\)](#).

### 6.12.2.2 find()

```
template<class OPT >
const class_info< OPT > * GiNaC::class_info< OPT >::find (
 const std::string & class_name) [static]
```

Find [class\\_info](#) by name.

References [GiNaC::class\\_info< OPT >::find\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::class\\_info< OPT >::options](#).

Referenced by [GiNaC::class\\_info< OPT >::find\(\)](#).

### 6.12.2.3 dump\_hierarchy()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_hierarchy (
 bool verbose = false) [static]
```

Dump class hierarchy to std::cout.

References [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GiNaC::class\\_info< OPT >::next](#), and [GiNaC::tree\(\)](#).

### 6.12.2.4 dump\_tree()

```
template<class OPT >
void GiNaC::class_info< OPT >::dump_tree (
 tree_node * n,
 const std::string & prefix,
 bool verbose) [static], [private]
```

References [n](#), and [GiNaC::class\\_info< OPT >::options](#).

### 6.12.2.5 identify\_parents()

```
template<class OPT >
void GiNaC::class_info< OPT >::identify_parents () [static], [private]
```

References [GiNaC::class\\_info< OPT >::next](#).

Referenced by [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#).

## 6.12.3 Member Data Documentation

### 6.12.3.1 options

```
template<class OPT >
OPT GiNaC::class_info< OPT >::options
```

Referenced by [GiNaC::class\\_info< OPT >::dump\\_tree\(\)](#), [GiNaC::class\\_info< OPT >::find\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), and [GiNaC::set\\_print\\_func\(\)](#).

### 6.12.3.2 first

```
template<class OPT >
class_info< OPT > * GiNaC::class_info< OPT >::first = nullptr [static], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::class\\_info\(\)](#).

### 6.12.3.3 next

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::next [private]
```

Referenced by [GiNaC::class\\_info< OPT >::dump\\_hierarchy\(\)](#), [GiNaC::class\\_info< OPT >::find\(\)](#), and [GiNaC::class\\_info< OPT >::i](#).

### 6.12.3.4 parent

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::parent [mutable], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#).

### 6.12.3.5 parents\_identified

```
template<class OPT >
bool GiNaC::class_info< OPT >::parents_identified = false [static], [private]
```

Referenced by [GiNaC::class\\_info< OPT >::class\\_info\(\)](#).

The documentation for this class was generated from the following file:

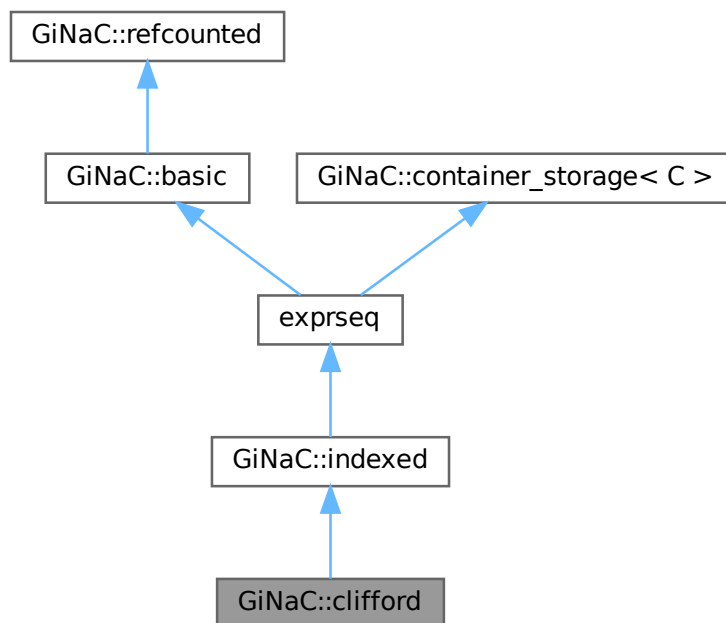
- [class\\_info.h](#)

## 6.13 GiNaC::clifford Class Reference

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::clifford:





## Public Member Functions inherited from GiNaC::indexed

- [indexed](#) (const [ex](#) &b)  
*Construct indexed object with no index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1)  
*Construct indexed object with one index.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices.*
- [indexed](#) (const [ex](#) &b, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2)  
*Construct indexed object with two indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Construct indexed object with three indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4)  
*Construct indexed object with four indices and a specified symmetry.*
- [indexed](#) (const [ex](#) &b, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices.*
- [indexed](#) (const [ex](#) &b, const [symmetry](#) &symm, const [exvector](#) &iv)  
*Construct indexed object with a specified vector of indices and symmetry.*
- [indexed](#) (const [symmetry](#) &symm, const [exprseq](#) &es)
- [indexed](#) (const [symmetry](#) &symm, const [exvector](#) &v)
- [indexed](#) (const [symmetry](#) &symm, [exvector](#) &&v)
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [exvector get\\_free\\_indices](#) () const override  
*Return a vector containing the free indices of an expression.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- bool [all\\_index\\_values\\_are](#) (unsigned inf) const  
*Check whether all index values have a certain property.*
- [exvector get\\_indices](#) () const  
*Return a vector containing the object's indices.*
- [exvector get\\_dummy\\_indices](#) () const  
*Return a vector containing the dummy indices of the object, if any.*
- [exvector get\\_dummy\\_indices](#) (const [indexed](#) &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- bool [has\\_dummy\\_index\\_for](#) (const [ex](#) &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- [ex get\\_symmetry](#) () const  
*Return symmetry properties.*

## Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (exvector::const\_iterator b, exvector::const\_iterator e)
- `container` (std::initializer\_list< ex > il)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend (const ex &b)`  
*Add element at front.*
- `container & append (const ex &b)`  
*Add element at back.*
- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin ()` const
- `const_iterator end ()` const
- `const_reverse_iterator rbegin ()` const
- `const_reverse_iterator rend ()` const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate ()` const  
*Create a clone of this object on the heap.*
- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const

- *Output to stream.*  
virtual void [dbgprint](#) () const
- *Little wrapper around print to be called within a debugger.*  
virtual void [dbgprinttree](#) () const
- *Little wrapper around printtree to be called within a debugger.*  
virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int [n](#)=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool [contract\\_with](#) ([exvector::iterator](#) self, [exvector::iterator](#) other, [exvector](#) &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*

- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override  
*Perform automatic simplification on noncommutative product of clifford objects.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- void `do_print_dflit` (const `print_dflit` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::indexed`

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t` `return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

### Protected Member Functions inherited from [GiNaC::container< C >](#)

- [ex conjugate](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- ~[container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

### Protected Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent spin lines.*
- [ex metric](#)  
*Metric of the space, all constructors make it an indexed object.*
- int [commutator\\_sign](#)  
*It is the sign in the definition  $e_{\sim i} e_{\sim j} \pm e_{\sim j} e_{\sim i} = B(i, j) + B(j, i)$*

## Protected Attributes inherited from [GiNaC::indexed](#)

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::container< C >](#)

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

## Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

## Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef [C< ex >](#) [STLT](#)

## Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, size\_t)

### 6.13.1 Detailed Description

This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).

These objects only carry Lorentz indices. Spinor indices are hidden. A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Clifford algebras (objects with different labels commute).

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 clifford() [1/4]

```
GiNaC::clifford::clifford (
 const ex & b,
 unsigned char rl = 0)
```

Construct object without any indices.

This constructor is for internal use only. Use the [dirac\\_ONE\(\)](#) function instead.

See also

[dirac\\_ONE](#)

#### 6.13.2.2 clifford() [2/4]

```
GiNaC::clifford::clifford (
 const ex & b,
 const ex & mu,
 const ex & metr,
 unsigned char rl = 0,
 int comm_sign = -1)
```

Construct object with one Lorentz index.

This constructor is for internal use only. Use the [clifford\\_unit\(\)](#) or [dirac\\_gamma\(\)](#) functions instead.

See also

[clifford\\_unit](#)

[dirac\\_gamma](#)

References [GINAC\\_ASSERT](#).

#### 6.13.2.3 clifford() [3/4]

```
GiNaC::clifford::clifford (
 unsigned char rl,
 const ex & metr,
 int comm_sign,
 const exvector & v)
```

#### 6.13.2.4 clifford() [4/4]

```
GiNaC::clifford::clifford (
 unsigned char rl,
 const ex & metr,
 int comm_sign,
 exvector && v)
```

### 6.13.3 Member Function Documentation

#### 6.13.3.1 precedence()

```
unsigned GiNaC::clifford::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_dflt\(\)](#), and [do\\_print\\_latex\(\)](#).

#### 6.13.3.2 archive()

```
void GiNaC::clifford::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [metric](#), [n](#), and [representation\\_label](#).

#### 6.13.3.3 read\_archive()

```
void GiNaC::clifford::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [metric](#), [n](#), and [representation\\_label](#).

#### 6.13.3.4 eval\_ncmul()

```
ex GiNaC::clifford::eval_ncmul (
 const exvector & v) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of clifford objects.

This removes superfluous ONEs, permutes gamma5/L/R's to the front and removes squares of gamma objects.

Reimplemented from [GiNaC::basic](#).

#### 6.13.3.5 match\_same\_type()

```
bool GiNaC::clifford::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [commutator\\_sign](#), [get\\_commutator\\_sign\(\)](#), [GINAC\\_ASSERT](#), [representation\\_label](#), and [same\\_metric\(\)](#).

#### 6.13.3.6 thiscontainer() [1/2]

```
ex GiNaC::clifford::thiscontainer (
 const exvector & v) const [override], [protected]
```

#### 6.13.3.7 thiscontainer() [2/2]

```
ex GiNaC::clifford::thiscontainer (
 exvector && v) const [override], [protected]
```

#### 6.13.3.8 return\_type()

```
unsigned GiNaC::clifford::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

**6.13.3.9 return\_type\_tinfo()**

```
return_type_t GiNaC::clifford::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

**6.13.3.10 get\_representation\_label()**

```
unsigned char GiNaC::clifford::get_representation_label () const [inline]
```

References [representation\\_label](#).

**6.13.3.11 get\_metric() [1/2]**

```
ex GiNaC::clifford::get_metric () const [inline]
```

References [metric](#).

Referenced by [same\\_metric\(\)](#).

**6.13.3.12 get\_metric() [2/2]**

```
ex GiNaC::clifford::get_metric (
 const ex & i,
 const ex & j,
 bool symmetrised = false) const [virtual]
```

References [GiNaC::\\_ex1\\_2](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::indexed::get\\_symmetry\(\)](#), [GiNaC::indexed::indexed\(\)](#), [metric](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::op\(\)](#), [GiNaC::indexed::simplify\\_indexed](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::symmetric2\(\)](#).

**6.13.3.13 same\_metric()**

```
bool GiNaC::clifford::same_metric (
 const ex & other) const
```

References [GiNaC::ex::get\\_free\\_indices\(\)](#), [get\\_metric\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), and [GiNaC::indexed::simplify\\_indexed](#).

Referenced by [match\\_same\\_type\(\)](#).

**6.13.3.14 get\_commutator\_sign()**

```
int GiNaC::clifford::get_commutator_sign () const [inline]
```

References [commutator\\_sign](#).

Referenced by [match\\_same\\_type\(\)](#).

**6.13.3.15 nops()**

```
size_t GiNaC::clifford::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

Referenced by [let\\_op\(\)](#), and [op\(\)](#).

**6.13.3.16 op()**

```
ex GiNaC::clifford::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

Referenced by [same\\_metric\(\)](#).

**6.13.3.17 let\_op()**

```
ex & GiNaC::clifford::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [nops\(\)](#), and [representation\\_label](#).

**6.13.3.18 subs()**

```
ex GiNaC::clifford::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [c](#), [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

### 6.13.3.19 do\_print\_dflt()

```
void GiNaC::clifford::do_print_dflt (
 const print_dflt & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [GiNaC::indexed::printindices\(\)](#), [representation\\_label](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.13.3.20 do\_print\_latex()

```
void GiNaC::clifford::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::is\\_dirac\\_slash\(\)](#), [precedence\(\)](#), [representation\\_label](#), and [GiNaC::container\\_storage< C >::seq](#).

### 6.13.3.21 do\_print\_tree()

```
void GiNaC::clifford::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [metric](#), [GiNaC::ex::print\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::indexed::symtree](#).

## 6.13.4 Member Data Documentation

### 6.13.4.1 representation\_label

```
unsigned char GiNaC::clifford::representation_label [protected]
```

Representation label to distinguish independent spin lines.

Referenced by [archive\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [get\\_representation\\_label\(\)](#), [let\\_op\(\)](#), [match\\_same\\_type\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [return\\_type\\_tinfo\(\)](#).

### 6.13.4.2 metric

```
ex GiNaC::clifford::metric [protected]
```

Metric of the space, all constructors make it an indexed object.

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_metric\(\)](#), [get\\_metric\(\)](#), and [read\\_archive\(\)](#).

### 6.13.4.3 commutator\_sign

```
int GiNaC::clifford::commutator_sign [protected]
```

It is the sign in the definition  $e_i e_j \pm e_j e_i = B(i, j) + B(j, i)$

Referenced by [archive\(\)](#), [get\\_commutator\\_sign\(\)](#), [match\\_same\\_type\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

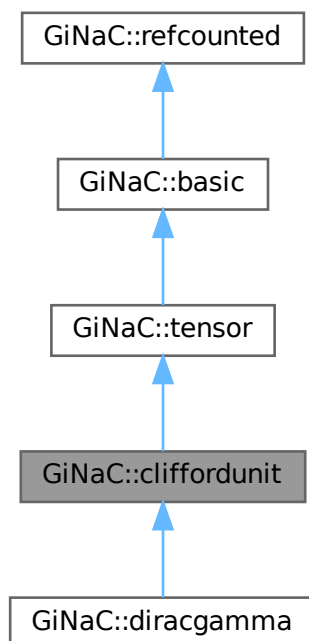
- [clifford.h](#)
- [clifford.cpp](#)

## 6.14 GiNaC::cliffordunit Class Reference

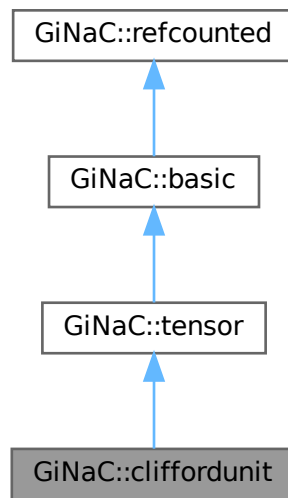
This class represents the Clifford algebra generators (units).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::cliffordunit:



Collaboration diagram for `GiNaC::cliffordunit`:



### Public Member Functions

- `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const` override  
*Contraction of a Clifford unit with something else.*

### Public Member Functions inherited from [GiNaC::tensor](#)

- `bool replace_contr_index (exvector::iterator self, exvector::iterator other) const`  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- `virtual ~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf () const`  
*Evaluate object numerically.*
- `virtual ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

**Protected Member Functions**

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

**Protected Member Functions inherited from `GiNaC::tensor`**

- unsigned `return_type` () const override

**Protected Member Functions inherited from `GiNaC::basic`**

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Additional Inherited Members****Protected Attributes inherited from `GiNaC::basic`**

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

**6.14.1 Detailed Description**

This class represents the Clifford algebra generators (units).

## 6.14.2 Member Function Documentation

### 6.14.2.1 `contract_with()`

```
bool GiNaC::cliffordunit::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of a Clifford unit with something else.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::diracgamma](#).

### 6.14.2.2 `do_print()`

```
void GiNaC::cliffordunit::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.14.2.3 `do_print_latex()`

```
void GiNaC::cliffordunit::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

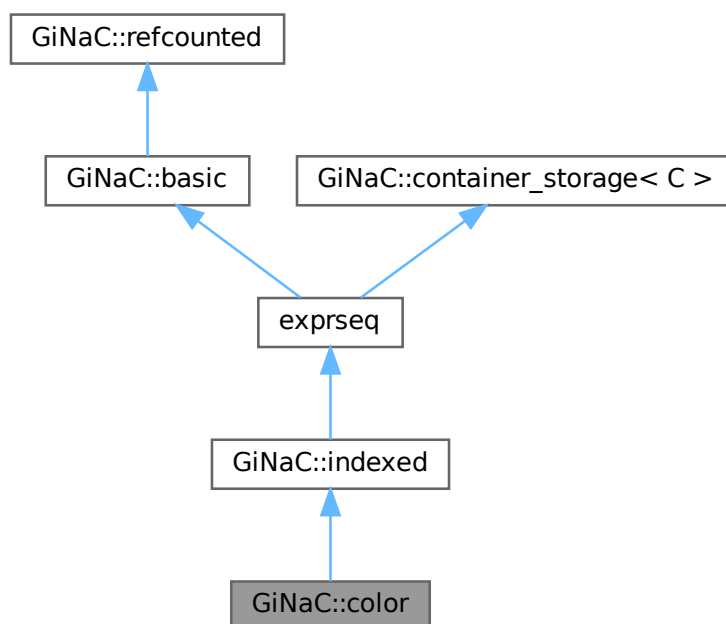
- [clifford.h](#)
- [clifford.cpp](#)

## 6.15 GiNaC::color Class Reference

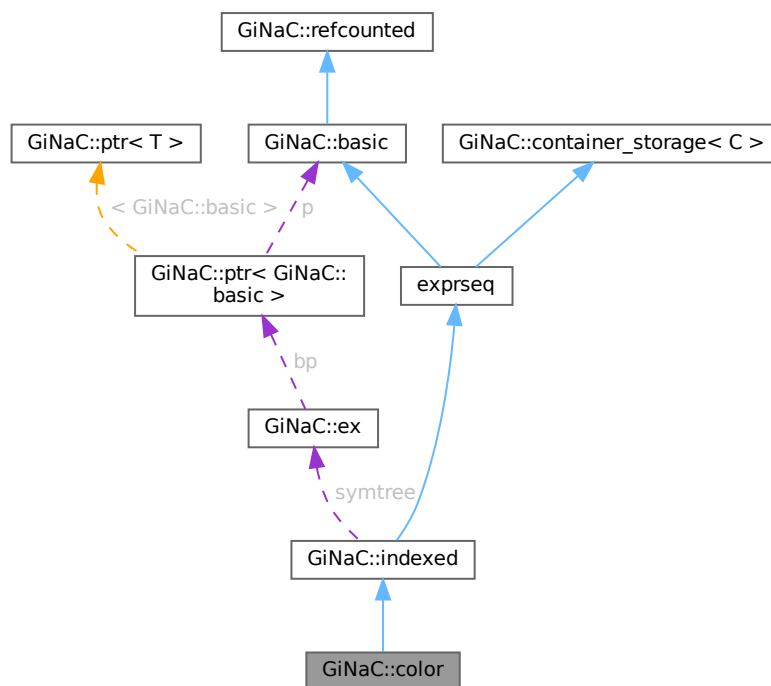
This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

```
#include <color.h>
```

Inheritance diagram for GiNaC::color:



Collaboration diagram for `GiNaC::color`:



## Public Member Functions

- `color` (const `ex` &b, unsigned char rl=0)  
*Construct object without any color index.*
- `color` (const `ex` &b, const `ex` &i1, unsigned char rl=0)  
*Construct object with one color index.*
- `color` (unsigned char rl, const `exvector` &v)
- `color` (unsigned char rl, `exvector` &&v)
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- unsigned char `get_representation_label` () const

## Public Member Functions inherited from `GiNaC::indexed`

- `indexed` (const `ex` &b)  
*Construct indexed object with no index.*
- `indexed` (const `ex` &b, const `ex` &i1)  
*Construct indexed object with one index.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
*Construct indexed object with two indices.*
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)

- Construct indexed object with three indices.*
  - `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)
  - Construct indexed object with four indices.*
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)
  - Construct indexed object with two indices and a specified symmetry.*
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)
  - Construct indexed object with three indices and a specified symmetry.*
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)
  - Construct indexed object with four indices and a specified symmetry.*
  - `indexed` (const `ex` &b, const `exvector` &iv)
  - Construct indexed object with a specified vector of indices.*
  - `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)
  - Construct indexed object with a specified vector of indices and symmetry.*
  - `indexed` (const `symmetry` &symm, const `exprseq` &es)
  - `indexed` (const `symmetry` &symm, const `exvector` &v)
  - `indexed` (const `symmetry` &symm, `exvector` &&v)
  - unsigned `precedence` () const override
  - Return relative operator precedence (for parenthezing output).*
  - bool `info` (unsigned inf) const override
  - Information about the object.*
  - `ex eval` () const override
  - Perform automatic non-interruptive term rewriting rules.*
  - `ex real_part` () const override
  - `ex imag_part` () const override
  - `exvector get_free_indices` () const override
  - Return a vector containing the free indices of an expression.*
  - void `archive` (`archive_node` &n) const override
  - Save (a.k.a.*
  - void `read_archive` (const `archive_node` &n, `lst` &symbols) override
  - Read (a.k.a.*
  - bool `all_index_values_are` (unsigned inf) const
  - Check whether all index values have a certain property.*
  - `exvector get_indices` () const
  - Return a vector containing the object's indices.*
  - `exvector get_dummy_indices` () const
  - Return a vector containing the dummy indices of the object, if any.*
  - `exvector get_dummy_indices` (const `indexed` &other) const
  - Return a vector containing the dummy indices in the contraction with another indexed object.*
  - bool `has_dummy_index_for` (const `ex` &i) const
  - Check whether the object has an index that forms a dummy index pair with a given index.*
  - `ex get_symmetry` () const
  - Return symmetry properties.*

## Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (`exvector::const_iterator` b, `exvector::const_iterator` e)
- `container` (`std::initializer_list< ex >` il)
- `size_t nops` () const override
- Number of operands/members.*

- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend` (const `ex` &b)  
*Add element at front.*
- `container & append` (const `ex` &b)  
*Add element at back.*
- `container & remove_first` ()  
*Remove first element.*
- `container & remove_last` ()  
*Remove last element.*
- `container & remove_all` ()  
*Remove all elements.*
- `container & sort` ()  
*Sort elements.*
- `container & unique` ()  
*Remove adjacent duplicate elements.*
- `const_iterator begin` () const
- `const_iterator end` () const
- `const_reverse_iterator rbegin` () const
- `const_reverse_iterator rend` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const

- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic & hold` () const

*Stop further evaluation.*

- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*

- const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &v) const override  
*Perform automatic simplification on noncommutative product of color objects.*
- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override

## Protected Member Functions inherited from `GiNaC::indexed`

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

## Protected Member Functions inherited from [GiNaC::container< C >](#)

- [ex conjugate](#) () const override
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Private Attributes

- unsigned char [representation\\_label](#)  
*Representation label to distinguish independent color matrices coming from separated fermion lines.*

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >](#)::STLT STLT

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > STLT

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

### Protected Attributes inherited from [GiNaC::indexed](#)

- [ex symtree](#)  
*Index symmetry (tree of symmetry objects)*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- STLT seq

### 6.15.1 Detailed Description

This class holds a generator  $T_a$  or the unity element of the Lie algebra of  $SU(3)$ , as used for calculations in quantum chromodynamics.

A representation label (an unsigned 8-bit integer) is used to distinguish elements from different Lie algebras (objects with different labels commute). These objects implement an abstract representation of the group, not a specific matrix representation. The indices used for color objects should not have a variance.

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 `color()` [1/4]

```
GiNaC::color::color (
 const ex & b,
 unsigned char rl = 0)
```

Construct object without any color index.

This constructor is for internal use only. Use the `color_ONE()` function instead.

See also

[color\\_ONE](#)

Referenced by [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 6.15.2.2 `color()` [2/4]

```
GiNaC::color::color (
 const ex & b,
 const ex & il,
 unsigned char rl = 0)
```

Construct object with one color index.

This constructor is for internal use only. Use the `color_T()` function instead.

See also

[color\\_T](#)

#### 6.15.2.3 `color()` [3/4]

```
GiNaC::color::color (
 unsigned char rl,
 const exvector & v)
```

#### 6.15.2.4 color() [4/4]

```
GiNaC::color::color (
 unsigned char rl,
 exvector && v)
```

### 6.15.3 Member Function Documentation

#### 6.15.3.1 archive()

```
void GiNaC::color::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), and [representation\\_label](#).

#### 6.15.3.2 read\_archive()

```
void GiNaC::color::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [representation\\_label](#).

#### 6.15.3.3 eval\_ncmul()

```
ex GiNaC::color::eval_ncmul (
 const exvector & v) const [override], [protected], [virtual]
```

Perform automatic simplification on noncommutative product of color objects.

This removes superfluous ONES.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::hold\\_ncmul\(\)](#).

### 6.15.3.4 match\_same\_type()

```
bool GiNaC::color::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [representation\\_label](#).

### 6.15.3.5 thiscontainer() [1/2]

```
ex GiNaC::color::thiscontainer (
 const exvector & v) const [override], [protected]
```

References [color\(\)](#), and [representation\\_label](#).

### 6.15.3.6 thiscontainer() [2/2]

```
ex GiNaC::color::thiscontainer (
 exvector && v) const [override], [protected]
```

References [color\(\)](#), and [representation\\_label](#).

### 6.15.3.7 return\_type()

```
unsigned GiNaC::color::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

### 6.15.3.8 return\_type\_tinfo()

```
return_type_t GiNaC::color::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [representation\\_label](#).

### 6.15.3.9 `get_representation_label()`

```
unsigned char GiNaC::color::get_representation_label () const [inline]
```

References [representation\\_label](#).

## 6.15.4 Member Data Documentation

### 6.15.4.1 `representation_label`

```
unsigned char GiNaC::color::representation_label [private]
```

Representation label to distinguish independent color matrices coming from separated fermion lines.

Referenced by [archive\(\)](#), [get\\_representation\\_label\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [return\\_type\\_tinfo\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.16 `GiNaC::compare_all_equal< T >` Class Template Reference

Comparison policy: all structures of one type are equal.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_all\\_equal\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.16.1 Detailed Description

```
template<class T>
class GiNaC::compare_all_equal< T >
```

Comparison policy: all structures of one type are equal.

## 6.16.2 Constructor & Destructor Documentation

### 6.16.2.1 ~compare\_all\_equal()

```
template<class T >
GiNaC::compare_all_equal< T >::~~compare_all_equal () [inline], [protected]
```

## 6.16.3 Member Function Documentation

### 6.16.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_all_equal< T >::struct_is_equal (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

### 6.16.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_all_equal< T >::struct_compare (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.17 GiNaC::compare\_bitwise< T > Class Template Reference

Comparison policy: use bit-wise comparison to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_bitwise](#) ()

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.17.1 Detailed Description

```
template<class T>
class GiNaC::compare_bitwise< T >
```

Comparison policy: use bit-wise comparison to compare structures.

## 6.17.2 Constructor & Destructor Documentation

### 6.17.2.1 `~compare_bitwise()`

```
template<class T >
GiNaC::compare_bitwise< T >::~~compare_bitwise () [inline], [protected]
```

## 6.17.3 Member Function Documentation

### 6.17.3.1 `struct_is_equal()`

```
template<class T >
static bool GiNaC::compare_bitwise< T >::struct_is_equal (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

### 6.17.3.2 `struct_compare()`

```
template<class T >
static int GiNaC::compare_bitwise< T >::struct_compare (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

- [structure.h](#)

## 6.18 GiNaC::compare\_std\_less< T > Class Template Reference

Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.

```
#include <structure.h>
```

### Protected Member Functions

- [~compare\\_std\\_less\(\)](#)

### Static Protected Member Functions

- static bool [struct\\_is\\_equal](#) (const T \*t1, const T \*t2)
- static int [struct\\_compare](#) (const T \*t1, const T \*t2)

### 6.18.1 Detailed Description

```
template<class T>
class GiNaC::compare_std_less< T >
```

Comparison policy: use `std::equal_to`/`std::less` (defaults to operators `==` and `<`) to compare structures.

## 6.18.2 Constructor & Destructor Documentation

### 6.18.2.1 ~compare\_std\_less()

```
template<class T >
GiNaC::compare_std_less< T >::~~compare_std_less () [inline], [protected]
```

## 6.18.3 Member Function Documentation

### 6.18.3.1 struct\_is\_equal()

```
template<class T >
static bool GiNaC::compare_std_less< T >::struct_is_equal (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

### 6.18.3.2 struct\_compare()

```
template<class T >
static int GiNaC::compare_std_less< T >::struct_compare (
 const T * t1,
 const T * t2) [inline], [static], [protected]
```

The documentation for this class was generated from the following file:

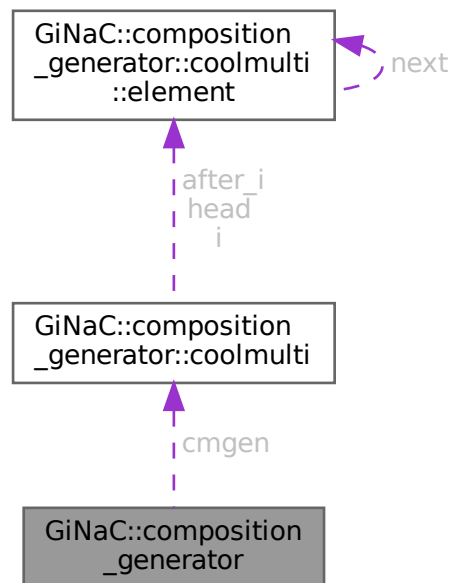
- [structure.h](#)

## 6.19 GiNaC::composition\_generator Class Reference

Generate all compositions of a partition of an integer n, starting with the compositions which has non-decreasing order.

```
#include <utils.h>
```

Collaboration diagram for `GiNaC::composition_generator`:



## Classes

- struct `coolmulti`

## Public Member Functions

- `composition_generator` (const std::vector< unsigned > &partition)
- const std::vector< unsigned > & `get` () const
- bool `next` ()

## Private Attributes

- struct `GiNaC::composition_generator::coolmulti` `cmgen`
- bool `atend`
- bool `trivial`
- std::vector< unsigned > `composition`
- bool `current_updated`

### 6.19.1 Detailed Description

Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.

## 6.19.2 Constructor & Destructor Documentation

### 6.19.2.1 composition\_generator()

```
GiNaC::composition_generator::composition_generator (
 const std::vector< unsigned > & partition) [inline], [explicit]
```

References [trivial](#).

## 6.19.3 Member Function Documentation

### 6.19.3.1 get()

```
const std::vector< unsigned > & GiNaC::composition_generator::get () const [inline]
```

References [cmgen](#), [composition](#), [current\\_updated](#), [GiNaC::composition\\_generator::coolmulti::head](#), [GiNaC::composition\\_generator::coolmulti::element::value](#), and [GiNaC::composition\\_generator::coolmulti::element::value](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

### 6.19.3.2 next()

```
bool GiNaC::composition_generator::next () [inline]
```

References [attend](#), [cmgen](#), [current\\_updated](#), [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::coolmulti::element::value](#), and [trivial](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

## 6.19.4 Member Data Documentation

### 6.19.4.1 cmgen

```
struct GiNaC::composition_generator::coolmulti GiNaC::composition_generator::cmgen [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

### 6.19.4.2 attend

```
bool GiNaC::composition_generator::attend [private]
```

Referenced by [next\(\)](#).

### 6.19.4.3 trivial

```
bool GiNaC::composition_generator::trivial [private]
```

Referenced by [composition\\_generator\(\)](#), and [next\(\)](#).

#### 6.19.4.4 composition

```
std::vector<unsigned> GiNaC::composition_generator::composition [mutable], [private]
```

Referenced by [get\(\)](#).

#### 6.19.4.5 current\_updated

```
bool GiNaC::composition_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

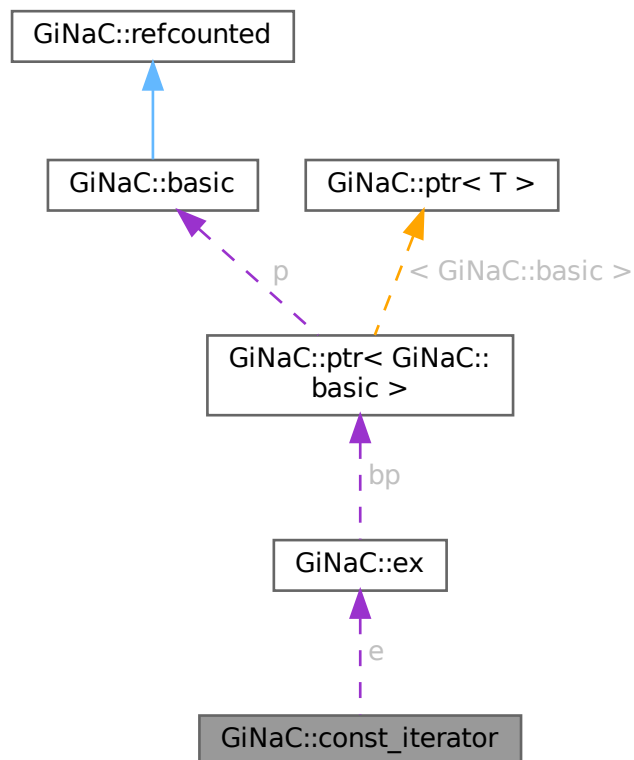
The documentation for this class was generated from the following file:

- [utils.h](#)

## 6.20 GiNaC::const\_iterator Class Reference

```
#include <ex.h>
```

Collaboration diagram for GiNaC::const\_iterator:



## Public Types

- using `iterator_category` = `std::random_access_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

## Public Member Functions

- `const_iterator` () noexcept
- `ex operator*` () const
- `std::unique_ptr< ex > operator->` () const
- `ex operator[]` (`difference_type n`) const
- `const_iterator & operator++` () noexcept
- `const_iterator operator++` (`int`) noexcept
- `const_iterator & operator+=` (`difference_type n`) noexcept
- `const_iterator operator+` (`difference_type n`) const noexcept
- `const_iterator & operator--` () noexcept
- `const_iterator operator--` (`int`) noexcept
- `const_iterator & operator-=` (`difference_type n`) noexcept
- `const_iterator operator-` (`difference_type n`) const noexcept
- `bool operator==` (`const const_iterator &other`) const noexcept
- `bool operator!=` (`const const_iterator &other`) const noexcept
- `bool operator<` (`const const_iterator &other`) const noexcept
- `bool operator>` (`const const_iterator &other`) const noexcept
- `bool operator<=` (`const const_iterator &other`) const noexcept
- `bool operator>=` (`const const_iterator &other`) const noexcept

## Protected Attributes

- `ex e`
- `size_t i`

## Private Member Functions

- `const_iterator` (`const ex &e`, `size_t i`) noexcept

## Friends

- class `ex`
- class `const_preorder_iterator`
- class `const_postorder_iterator`
- `const_iterator operator+` (`difference_type n`, `const const_iterator &it`) noexcept
- `difference_type operator-` (`const const_iterator &lhs`, `const const_iterator &rhs`) noexcept

## 6.20.1 Member Typedef Documentation

### 6.20.1.1 iterator\_category

```
using GiNaC::const_iterator::iterator_category = std::random_access_iterator_tag
```

### 6.20.1.2 value\_type

```
using GiNaC::const_iterator::value_type = ex
```

### 6.20.1.3 difference\_type

```
using GiNaC::const_iterator::difference_type = ptrdiff_t
```

### 6.20.1.4 pointer

```
using GiNaC::const_iterator::pointer = const ex *
```

### 6.20.1.5 reference

```
using GiNaC::const_iterator::reference = const ex &
```

## 6.20.2 Constructor & Destructor Documentation

### 6.20.2.1 const\_iterator() [1/2]

```
GiNaC::const_iterator::const_iterator () [inline], [noexcept]
```

Referenced by [operator+\(\)](#), and [operator-\(\)](#).

### 6.20.2.2 const\_iterator() [2/2]

```
GiNaC::const_iterator::const_iterator (
 const ex & e_,
 size_t i_) [inline], [private], [noexcept]
```

## 6.20.3 Member Function Documentation

### 6.20.3.1 operator\*()

```
ex GiNaC::const_iterator::operator* () const [inline]
```

References [e](#), [i](#), and [GiNaC::ex::op\(\)](#).

### 6.20.3.2 operator->()

```
std::unique_ptr< ex > GiNaC::const_iterator::operator-> () const [inline]
```

References [ex](#).

### 6.20.3.3 operator[]()

```
ex GiNaC::const_iterator::operator[] (
 difference_type n) const [inline]
```

References [e](#), [i](#), [n](#), and [GiNaC::ex::op\(\)](#).

### 6.20.3.4 operator++() [1/2]

```
const_iterator & GiNaC::const_iterator::operator++ () [inline], [noexcept]
```

References [i](#).

### 6.20.3.5 operator++() [2/2]

```
const_iterator GiNaC::const_iterator::operator++ (
 int) [inline], [noexcept]
```

References [i](#).

### 6.20.3.6 operator+=()

```
const_iterator & GiNaC::const_iterator::operator+= (
 difference_type n) [inline], [noexcept]
```

References [i](#), and [n](#).

### 6.20.3.7 operator+()

```
const_iterator GiNaC::const_iterator::operator+ (
 difference_type n) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), [i](#), and [n](#).

### 6.20.3.8 operator--() [1/2]

```
const_iterator & GiNaC::const_iterator::operator-- () [inline], [noexcept]
```

References [i](#).

### 6.20.3.9 operator--() [2/2]

```
const_iterator GiNaC::const_iterator::operator-- (
 int) [inline], [noexcept]
```

References [i](#).

#### 6.20.3.10 operator-=( )

```
const_iterator & GiNaC::const_iterator::operator-= (
 difference_type n) [inline], [noexcept]
```

References [i](#), and [n](#).

#### 6.20.3.11 operator-( )

```
const_iterator GiNaC::const_iterator::operator- (
 difference_type n) const [inline], [noexcept]
```

References [const\\_iterator\(\)](#), [e](#), [i](#), and [n](#).

#### 6.20.3.12 operator==( )

```
bool GiNaC::const_iterator::operator== (
 const const_iterator & other) const [inline], [noexcept]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [e](#), and [i](#).

#### 6.20.3.13 operator!=( )

```
bool GiNaC::const_iterator::operator!= (
 const const_iterator & other) const [inline], [noexcept]
```

#### 6.20.3.14 operator<( )

```
bool GiNaC::const_iterator::operator< (
 const const_iterator & other) const [inline], [noexcept]
```

References [i](#).

#### 6.20.3.15 operator>( )

```
bool GiNaC::const_iterator::operator> (
 const const_iterator & other) const [inline], [noexcept]
```

#### 6.20.3.16 operator<=( )

```
bool GiNaC::const_iterator::operator<= (
 const const_iterator & other) const [inline], [noexcept]
```

#### 6.20.3.17 operator>=( )

```
bool GiNaC::const_iterator::operator>= (
 const const_iterator & other) const [inline], [noexcept]
```

## 6.20.4 Friends And Related Symbol Documentation

### 6.20.4.1 ex

```
friend class ex [friend]
```

Referenced by [operator->\(\)](#).

### 6.20.4.2 const\_preorder\_iterator

```
friend class const_preorder_iterator [friend]
```

### 6.20.4.3 const\_postorder\_iterator

```
friend class const_postorder_iterator [friend]
```

### 6.20.4.4 operator+

```
const_iterator operator+ (
 difference_type n,
 const const_iterator & it) [friend]
```

### 6.20.4.5 operator-

```
difference_type operator- (
 const const_iterator & lhs,
 const const_iterator & rhs) [friend]
```

## 6.20.5 Member Data Documentation

### 6.20.5.1 e

```
ex GiNaC::const_iterator::e [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator-\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

### 6.20.5.2 i

```
size_t GiNaC::const_iterator::i [protected]
```

Referenced by [operator\\*\(\)](#), [operator+\(\)](#), [operator++\(\)](#), [operator++\(\)](#), [operator+=\(\)](#), [operator-\(\)](#), [operator--\(\)](#), [operator--\(\)](#), [operator-=\(\)](#), [operator<\(\)](#), [operator==\(\)](#), and [operator\[\]\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.21 GiNaC::const\_postorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using `iterator_category` = `std::forward_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

### Public Member Functions

- `const_postorder_iterator` () noexcept
- `const_postorder_iterator` (const `ex` &`e`, `size_t` `n`)
- `reference operator*` () const
- `pointer operator->` () const
- `const_postorder_iterator` & `operator++` ()
- `const_postorder_iterator` `operator++` (int)
- bool `operator==` (const `const_postorder_iterator` &`other`) const noexcept
- bool `operator!=` (const `const_postorder_iterator` &`other`) const noexcept

### Private Member Functions

- void `descend` ()
- void `increment` ()

### Private Attributes

- `std::stack< internal::_iter_rep, std::vector< internal::_iter_rep > >` `s`

## 6.21.1 Member Typedef Documentation

### 6.21.1.1 iterator\_category

```
using GiNaC::const_postorder_iterator::iterator_category = std::forward_iterator_tag
```

### 6.21.1.2 value\_type

```
using GiNaC::const_postorder_iterator::value_type = ex
```

### 6.21.1.3 difference\_type

```
using GiNaC::const_postorder_iterator::difference_type = ptrdiff_t
```

#### 6.21.1.4 pointer

using [GiNaC::const\\_postorder\\_iterator::pointer](#) = const [ex](#) \*

#### 6.21.1.5 reference

using [GiNaC::const\\_postorder\\_iterator::reference](#) = const [ex](#) &

### 6.21.2 Constructor & Destructor Documentation

#### 6.21.2.1 const\_postorder\_iterator() [1/2]

[GiNaC::const\\_postorder\\_iterator::const\\_postorder\\_iterator](#) ( ) [inline], [noexcept]

#### 6.21.2.2 const\_postorder\_iterator() [2/2]

[GiNaC::const\\_postorder\\_iterator::const\\_postorder\\_iterator](#) (  
    const [ex](#) & e,  
    size\_t n ) [inline]

References [descend\(\)](#), [n](#), and [s](#).

### 6.21.3 Member Function Documentation

#### 6.21.3.1 operator\*()

[reference](#) [GiNaC::const\\_postorder\\_iterator::operator\\*](#) ( ) const [inline]

References [s](#).

#### 6.21.3.2 operator->()

[pointer](#) [GiNaC::const\\_postorder\\_iterator::operator->](#) ( ) const [inline]

References [s](#).

#### 6.21.3.3 operator++() [1/2]

[const\\_postorder\\_iterator](#) & [GiNaC::const\\_postorder\\_iterator::operator++](#) ( ) [inline]

References [increment\(\)](#).

#### 6.21.3.4 `operator++()` [2/2]

```
const_postorder_iterator GiNaC::const_postorder_iterator::operator++ (
 int) [inline]
```

References [increment\(\)](#).

#### 6.21.3.5 `operator==()`

```
bool GiNaC::const_postorder_iterator::operator== (
 const const_postorder_iterator & other) const [inline], [noexcept]
```

References [s](#).

#### 6.21.3.6 `operator!=()`

```
bool GiNaC::const_postorder_iterator::operator!= (
 const const_postorder_iterator & other) const [inline], [noexcept]
```

#### 6.21.3.7 `descend()`

```
void GiNaC::const_postorder_iterator::descend () [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [const\\_postorder\\_iterator\(\)](#), and [increment\(\)](#).

#### 6.21.3.8 `increment()`

```
void GiNaC::const_postorder_iterator::increment () [inline], [private]
```

References [descend\(\)](#), and [s](#).

Referenced by [operator++\(\)](#), and [operator++\(\)](#).

### 6.21.4 Member Data Documentation

#### 6.21.4.1 `s`

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_postorder_↵
iterator::s [private]
```

Referenced by [const\\_postorder\\_iterator\(\)](#), [descend\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following file:

- [ex.h](#)

## 6.22 GiNaC::const\_preorder\_iterator Class Reference

```
#include <ex.h>
```

### Public Types

- using `iterator_category` = `std::forward_iterator_tag`
- using `value_type` = `ex`
- using `difference_type` = `ptrdiff_t`
- using `pointer` = `const ex *`
- using `reference` = `const ex &`

### Public Member Functions

- `const_preorder_iterator` () noexcept
- `const_preorder_iterator` (const `ex` &`e`, size\_t `n`)
- `reference operator*` () const
- `pointer operator->` () const
- `const_preorder_iterator & operator++` ()
- `const_preorder_iterator operator++` (int)
- bool `operator==` (const `const_preorder_iterator` &`other`) const noexcept
- bool `operator!=` (const `const_preorder_iterator` &`other`) const noexcept

### Private Member Functions

- void `increment` ()

### Private Attributes

- `std::stack< internal::_iter_rep, std::vector< internal::_iter_rep > >` `s`

## 6.22.1 Member Typedef Documentation

### 6.22.1.1 iterator\_category

```
using GiNaC::const_preorder_iterator::iterator_category = std::forward_iterator_tag
```

### 6.22.1.2 value\_type

```
using GiNaC::const_preorder_iterator::value_type = ex
```

### 6.22.1.3 difference\_type

```
using GiNaC::const_preorder_iterator::difference_type = ptrdiff_t
```

#### 6.22.1.4 pointer

using [GiNaC::const\\_preorder\\_iterator::pointer](#) = const [ex](#) \*

#### 6.22.1.5 reference

using [GiNaC::const\\_preorder\\_iterator::reference](#) = const [ex](#) &

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 const\_preorder\_iterator() [1/2]

[GiNaC::const\\_preorder\\_iterator::const\\_preorder\\_iterator](#) ( ) [inline], [noexcept]

#### 6.22.2.2 const\_preorder\_iterator() [2/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator (
 const ex & e,
 size_t n) [inline]
```

References [n](#), and [s](#).

### 6.22.3 Member Function Documentation

#### 6.22.3.1 operator\*()

[reference](#) [GiNaC::const\\_preorder\\_iterator::operator\\*](#) ( ) const [inline]

References [s](#).

#### 6.22.3.2 operator->()

[pointer](#) [GiNaC::const\\_preorder\\_iterator::operator->](#) ( ) const [inline]

References [s](#).

#### 6.22.3.3 operator++() [1/2]

[const\\_preorder\\_iterator](#) & [GiNaC::const\\_preorder\\_iterator::operator++](#) ( ) [inline]

References [increment\(\)](#).

### 6.22.3.4 operator++() [2/2]

```
const_preorder_iterator GiNaC::const_preorder_iterator::operator++ (
 int) [inline]
```

References [increment\(\)](#).

### 6.22.3.5 operator==( )

```
bool GiNaC::const_preorder_iterator::operator== (
 const const_preorder_iterator & other) const [inline], [noexcept]
```

References [s](#).

### 6.22.3.6 operator!=( )

```
bool GiNaC::const_preorder_iterator::operator!= (
 const const_preorder_iterator & other) const [inline], [noexcept]
```

### 6.22.3.7 increment()

```
void GiNaC::const_preorder_iterator::increment () [inline], [private]
```

References [GiNaC::internal::\\_iter\\_rep::e](#), [GiNaC::internal::\\_iter\\_rep::i](#), [GiNaC::internal::\\_iter\\_rep::i\\_end](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [s](#).

Referenced by [operator++\(\)](#), and [operator++\(\)](#).

## 6.22.4 Member Data Documentation

### 6.22.4.1 s

```
std::stack<internal::_iter_rep, std::vector<internal::_iter_rep> > GiNaC::const_preorder_iterator::s [private]
```

Referenced by [const\\_preorder\\_iterator\(\)](#), [increment\(\)](#), [operator\\*\(\)](#), [operator->\(\)](#), and [operator==\( \)](#).

The documentation for this class was generated from the following file:

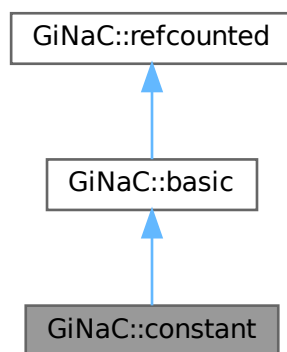
- [ex.h](#)

## 6.23 GiNaC::constant Class Reference

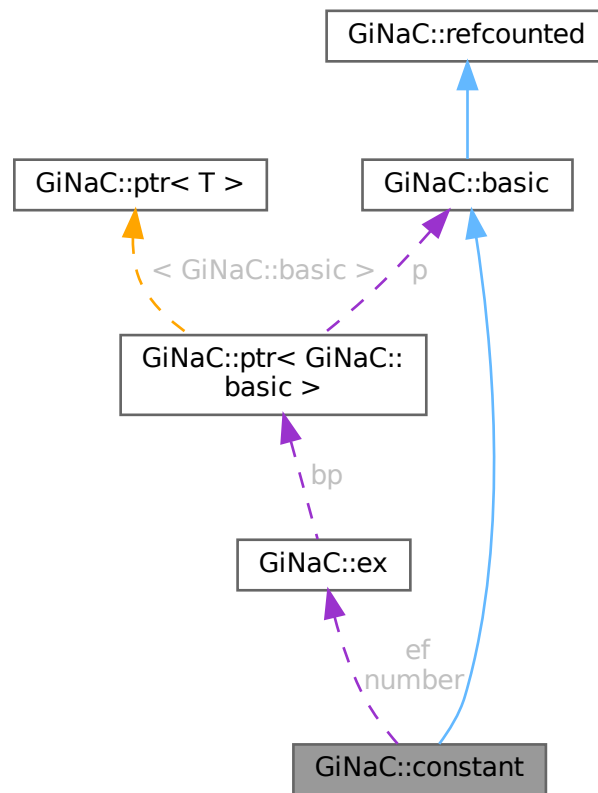
This class holds constants, symbols with specific numerical value.

```
#include <constant.h>
```

Inheritance diagram for GiNaC::constant:



Collaboration diagram for GiNaC::constant:



## Public Member Functions

- `constant` (const std::string &initname, evalfunctype efun=nullptr, const std::string &texname=std::string(), unsigned domain=domain::complex)
- `constant` (const std::string &initname, const numeric &initnumber, const std::string &texname=std::string(), unsigned domain=domain::complex)
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (archive\_node &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const archive\_node &n, lst &syms) override  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position *i*.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position *i*.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object *s*.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object *s*.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree *n* in object *s*.*
- virtual `ex expand (unsigned options=0) const`

- *Expand expression, i.e.*  
virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const
- *Sort expanded expression in terms of powers of some object(s).*  
virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const
- *Default implementation of [ex::series\(\)](#).*  
virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const
- *Default implementation of [ex::normal\(\)](#).*  
virtual [ex to\\_rational](#) ([exmap](#) &repl) const
- *Default implementation of [ex::to\\_rational\(\)](#).*  
virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*  
virtual [numeric max\\_coefficient](#) () const
- *Implementation [ex::max\\_coefficient\(\)](#).*  
virtual [exvector get\\_free\\_indices](#) () const
- *Return a vector containing the free indices of an expression.*  
virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const
- *Add two indexed expressions.*  
virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const
- *Multiply an indexed expression with a scalar.*  
virtual bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const
- *Try to contract two indexed expressions that appear in the same product.*  
virtual unsigned [return\\_type](#) () const
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const
- *Like [print\(\)](#), but dispatch to the specified class.*  
void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const
- *Like [print\(\)](#), but dispatch to the specified class.*  
[ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const
- *Helper function for [subs\(\)](#).*  
[ex diff](#) (const [symbol](#) &s, unsigned nth=1) const
- *Default interface of nth derivative [ex::diff\(s, n\)](#).*  
int [compare](#) (const [basic](#) &other) const
- *Compare objects syntactically to establish canonical ordering.*  
bool [is\\_equal](#) (const [basic](#) &other) const
- *Test for syntactic equality.*  
const [basic](#) & [hold](#) () const
- *Stop further evaluation.*  
unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const
- *Set some [status\\_flags](#).*  
const [basic](#) & [clearflag](#) (unsigned f) const
- *Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a constant always returns 0.*
- `bool is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual `bool match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- `std::string name`  
*printrname of this constant*
- `std::string TeX_name`  
*LaTeX name.*
- `evalffunc type ef`
- `ex number`  
*numerical value this constant `evalf()`s to*
- `unsigned serial`  
*unique serial number for comparison*
- `unsigned domain`  
*numerical value this constant `evalf()`s to*

## Static Private Attributes

- static `unsigned next_serial` = 0

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.23.1 Detailed Description

This class holds constants, symbols with specific numerical value.

Each object of this class must either provide their own function to evaluate it to class numeric or provide the constant as a numeric (if it's an exact number).

## 6.23.2 Constructor & Destructor Documentation

### 6.23.2.1 `constant()` [1/2]

```
GiNaC::constant::constant (
 const std::string & initname,
 evalfunctype efun = nullptr,
 const std::string & texname = std::string(),
 unsigned domain = domain::complex)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

### 6.23.2.2 `constant()` [2/2]

```
GiNaC::constant::constant (
 const std::string & initname,
 const numeric & initnumber,
 const std::string & texname = std::string(),
 unsigned domain = domain::complex)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [name](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

## 6.23.3 Member Function Documentation

### 6.23.3.1 `info()`

```
bool GiNaC::constant::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::domain::real](#), and [GiNaC::info\\_flags::real](#).

### 6.23.3.2 evalf()

```
ex GiNaC::constant::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [ef](#), [GiNaC::ex::evalf\(\)](#), and [number](#).

Referenced by [GiNaC::EllipticE\\_eval\(\)](#), and [GiNaC::EllipticK\\_eval\(\)](#).

### 6.23.3.3 is\_polynomial()

```
bool GiNaC::constant::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

### 6.23.3.4 conjugate()

```
ex GiNaC::constant::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.5 real\_part()

```
ex GiNaC::constant::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.6 imag\_part()

```
ex GiNaC::constant::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::domain::positive](#), and [GiNaC::domain::real](#).

### 6.23.3.7 archive()

```
void GiNaC::constant::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), and [name](#).

### 6.23.3.8 read\_archive()

```
void GiNaC::constant::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::Catalan](#), [GiNaC::Euler](#), [n](#), [name](#), and [GiNaC::Pi](#).

### 6.23.3.9 derivative()

```
ex GiNaC::constant::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a constant always returns 0.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

#### 6.23.3.10 `is_equal_same_type()`

```
bool GiNaC::constant::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

#### 6.23.3.11 `calchash()`

```
unsigned GiNaC::constant::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.23.3.12 `do_print()`

```
void GiNaC::constant::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [name](#).

#### 6.23.3.13 `do_print_tree()`

```
void GiNaC::constant::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [name](#).

#### 6.23.3.14 `do_print_latex()`

```
void GiNaC::constant::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [TeX\\_name](#).

### 6.23.3.15 do\_print\_python\_repr()

```
void GiNaC::constant::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [name](#), and [TeX\\_name](#).

## 6.23.4 Member Data Documentation

### 6.23.4.1 name

```
std::string GiNaC::constant::name [private]
```

printname of this constant

Referenced by [archive\(\)](#), [constant\(\)](#), [constant\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), and [read\\_archive\(\)](#).

### 6.23.4.2 TeX\_name

```
std::string GiNaC::constant::TeX_name [private]
```

LaTeX name.

Referenced by [constant\(\)](#), [constant\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

### 6.23.4.3 ef

```
evalffunctype GiNaC::constant::ef [private]
```

Referenced by [evalf\(\)](#).

### 6.23.4.4 number

```
ex GiNaC::constant::number [private]
```

numerical value this constant [evalf\(\)](#)s to

Referenced by [evalf\(\)](#).

### 6.23.4.5 serial

```
unsigned GiNaC::constant::serial [private]
```

unique serial number for comparison

Referenced by [calchash\(\)](#), and [is\\_equal\\_same\\_type\(\)](#).

#### 6.23.4.6 next\_serial

```
unsigned GiNaC::constant::next_serial = 0 [static], [private]
```

#### 6.23.4.7 domain

```
unsigned GiNaC::constant::domain [private]
```

numerical value this constant [evalf\(\)](#)s to

The documentation for this class was generated from the following files:

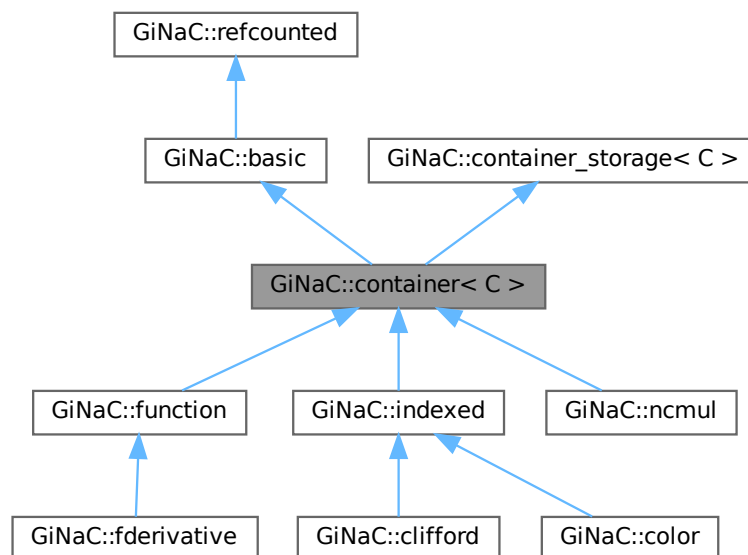
- [constant.h](#)
- [constant.cpp](#)

## 6.24 GiNaC::container< C > Class Template Reference

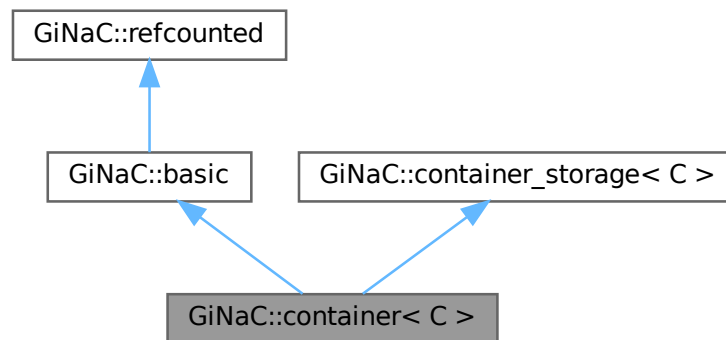
Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include <container.h>
```

Inheritance diagram for GiNaC::container< C >:



Collaboration diagram for GiNaC::container< C >:



### Public Types

- typedef `STLT::const_iterator` `const_iterator`
- typedef `STLT::const_reverse_iterator` `const_reverse_iterator`

### Public Member Functions

- `container` (`STLT const &s`)
- `container` (`STLT &&v`)
- `container` (`exvector::const_iterator b, exvector::const_iterator e`)
- `container` (`std::initializer_list< ex > il`)
- `bool info` (`unsigned inf`) `const` override  
*Information about the object.*
- `unsigned precedence` () `const` override  
*Return relative operator precedence (for parenthezing output).*
- `size_t nops` () `const` override  
*Number of operands/members.*
- `ex op` (`size_t i`) `const` override  
*Return operand/member at position i.*
- `ex & let_op` (`size_t i`) `override`  
*Return modifiable operand/member at position i.*
- `ex subs` (`const exmap &m, unsigned options=0`) `const` override  
*Substitute a set of objects by arbitrary expressions.*
- `void read_archive` (`const archive_node &n, lst &sym_lst`) `override`  
*Load (deserialize) the object from an archive node.*
- `void archive` (`archive_node &n`) `const` override  
*Archive the object.*
- `container & prepend` (`const ex &b`)  
*Add element at front.*
- `container & append` (`const ex &b`)  
*Add element at back.*
- `container & remove_first` ()

- Remove first element.*
- `container & remove_last ()`
- Remove last element.*
- `container & remove_all ()`
- Remove all elements.*
- `container & sort ()`
- Sort elements.*
- `container & unique ()`
- Remove adjacent duplicate elements.*
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`

- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &`s`) const

*Return degree of highest power in object `s`.*
- virtual int `ldegree` (const `ex` &`s`) const

*Return degree of lowest power in object `s`.*
- virtual `ex` `coeff` (const `ex` &`s`, int `n`=1) const

*Return coefficient of degree `n` in object `s`.*
- virtual `ex` `expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex` `collect` (const `ex` &`s`, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex` `series` (const `relational` &`r`, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex` `normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const

*Default implementation of `ex::normal()`.*
- virtual `ex` `to_rational` (`exmap` &`repl`) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex` `to_polynomial` (`exmap` &`repl`) const
- virtual `numeric` `integer_content` () const
- virtual `ex` `smod` (const `numeric` &`xi`) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric` `max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector` `get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex` `add_indexed` (const `ex` &`self`, const `ex` &`other`) const

*Add two indexed expressions.*
- virtual `ex` `scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- template<class `T` >
  - void `print_dispatch` (const `print_context` &`c`, unsigned `level`) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned `level`) const

*Like `print()`, but dispatch to the specified class.*
- `ex` `subs_one_level` (const `exmap` &`m`, unsigned `options`) const

*Helper function for `subs()`.*
- `ex` `diff` (const `symbol` &`s`, unsigned `nth`=1) const

*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &`other`) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &`other`) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const

*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

## Protected Types

- typedef `container_storage`< `C` >::STLT `STLT`

## Protected Types inherited from `GiNaC::container_storage< C >`

- typedef `C`< `ex` > `STLT`

## Protected Member Functions

- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- bool `is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer` (const `STLT` &`v`) const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer` (`STLT` &&`v`) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq` (const `print_context` &`c`, char `openbracket`, char `delim`, char `closebracket`, unsigned `this←_precedence`, unsigned `upper_precedence=0`) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &`c`, unsigned `level`) const
- void `do_print_tree` (const `print_tree` &`c`, unsigned `level`) const
- void `do_print_python` (const `print_python` &`c`, unsigned `level`) const
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const
- `STLT` `subchildren` (const `exmap` &`m`, unsigned `options=0`) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &`v`) const
- virtual bool `match_same_type` (const `basic` &`other`) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &`s`) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &`c`, unsigned `level`) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &`c`, unsigned `level`) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const  
*Python parsable output to stream.*

**Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

**Static Protected Member Functions**

- static unsigned [get\\_default\\_flags](#) ()  
  *Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
  *Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
  *Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

**Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)**

- static void [reserve](#) (STLT &, size\_t)

**Private Member Functions**

- void [sort\\_](#) (std::random\_access\_iterator\_tag)
- void [sort\\_](#) (std::input\_iterator\_tag)
- void [unique\\_](#) ()
- void [unique\\_](#) ()  
  *Specialization of [container::unique\\_\(\)](#) for [std::list](#).*

**Additional Inherited Members****Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
  *of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
  *hash value*

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT seq](#)

### 6.24.1 Detailed Description

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container< C >
```

Wrapper template for making [GiNaC](#) classes out of STL containers.

### 6.24.2 Member Typedef Documentation

#### 6.24.2.1 STLT

```
template<template< class T, class=std::allocator< T > > class C>
typedef container_storage<C>::STLT GiNaC::container< C >::STLT [protected]
```

#### 6.24.2.2 const\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STLT::const_iterator GiNaC::container< C >::const_iterator
```

#### 6.24.2.3 const\_reverse\_iterator

```
template<template< class T, class=std::allocator< T > > class C>
typedef STLT::const_reverse_iterator GiNaC::container< C >::const_reverse_iterator
```

### 6.24.3 Constructor & Destructor Documentation

#### 6.24.3.1 container() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
 STLT const & s) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [GiNaC::container< C >::thiscontainer\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.3.2 container() [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
 STLT && v) [inline], [explicit]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.3.3 container()** [3/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
 exvector::const_iterator b,
 exvector::const_iterator e) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.3.4 container()** [4/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container< C >::container (
 std::initializer_list< ex > il) [inline]
```

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.24.4 Member Function Documentation****6.24.4.1 get\_default\_flags()**

```
unsigned GiNaC::lst::get_default_flags () [inline], [static], [protected]
```

Specialization of [container::get\\_default\\_flags\(\)](#) for `lst`.

Referenced by [GiNaC::container< C >::container\(\)](#), [GiNaC::container< C >::container\(\)](#), [GiNaC::container< C >::container\(\)](#), [GiNaC::container< C >::container\(\)](#), and [GiNaC::container< C >::read\\_archive\(\)](#).

**6.24.4.2 get\_open\_delim()**

```
char GiNaC::lst::get_open_delim () [inline], [static], [protected]
```

Specialization of [container::get\\_open\\_delim\(\)](#) for `lst`.

**6.24.4.3 get\_close\_delim()**

```
char GiNaC::lst::get_close_delim () [inline], [static], [protected]
```

Specialization of [container::get\\_close\\_delim\(\)](#) for `lst`.

**6.24.4.4 info()**

```
template bool GiNaC::container< C >::info (
 unsigned inf) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

#### 6.24.4.5 precedence()

```
template<template< class T, class=std::allocator< T > > class C>
unsigned GiNaC::container< C >::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::function::print\(\)](#).

#### 6.24.4.6 nops()

```
template<template< class T, class=std::allocator< T > > class C>
size_t GiNaC::container< C >::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::calchash\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::diag\\_matrix\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::lst\\_to\\_matrix\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::ex::subs\(\)](#).

#### 6.24.4.7 op()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::nops\(\)](#).

Referenced by [GiNaC::function::calchash\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::indexed::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::indexed::real\\_part\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::indexed::return\\_type\(\)](#), [GiNaC::indexed::return\\_type\\_tinfo\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

#### 6.24.4.8 let\_op()

```
template<template< class T, class=std::allocator< T > > class C>
ex & GiNaC::container< C >::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::nops\(\)](#).

#### 6.24.4.9 subs()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::is\\_a\(\)](#), [m](#), and [options](#).

#### 6.24.4.10 read\_archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::read_archive (
 const archive_node & n,
 lst & syms) [inline], [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

##### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

References [GiNaC::container< C >::get\\_default\\_flags\(\)](#), [n](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.24.4.11 archive()

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::archive (
 archive_node & n) const [inline], [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), [GiNaC::function](#), and [GiNaC::indexed](#).

References [GiNaC::archive\\_node::add\\_ex\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.24.4.12 conjugate()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::conjugate () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), and [GiNaC::ncmul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::container< C >::thiscontainer\(\)](#), and [x](#).

Referenced by [GiNaC::ncmul::conjugate\(\)](#).

#### 6.24.4.13 real\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::real_part () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [cont](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

#### 6.24.4.14 imag\_part()

```
template<template< class T, class=std::allocator< T > > class C>
ex GiNaC::container< C >::imag_part () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::function](#), [GiNaC::indexed](#), and [GiNaC::ncmul](#).

References [GiNaC::container< C >::begin\(\)](#), [cont](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), and [GiNaC::container< C >::thiscontainer\(\)](#).

**6.24.4.15 is\_equal\_same\_type()**

```
template<template< class T, class=std::allocator< T > > class C>
bool GiNaC::container< C >::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::fderivative](#), and [GiNaC::function](#).

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::function::is\\_equal\\_same\\_type\(\)](#).

**6.24.4.16 thiscontainer() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
 const STLIT & v) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence.

Must be overridden by derived classes.

References [GiNaC::container< C >::container\(\)](#).

Referenced by [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), and [GiNaC::container< C >::real\\_part\(\)](#).

**6.24.4.17 thiscontainer() [2/2]**

```
template<template< class T, class=std::allocator< T > > class C>
virtual ex GiNaC::container< C >::thiscontainer (
 STLIT && v) const [inline], [protected], [virtual]
```

Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).

Must be overridden by derived classes.

References [GiNaC::container< C >::container\(\)](#).

**6.24.4.18 printseq()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::printseq (
 const print_context & c,
 char openbracket,
 char delim,
 char closebracket,
 unsigned this_precedence,
 unsigned upper_precedence = 0) const [protected], [virtual]
```

Print sequence of contained elements.

References [c](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [GiNaC::function::print\(\)](#).

**6.24.4.19 sort\_() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
 std::random_access_iterator_tag) [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.20 sort\_() [2/2]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::sort_ (
 std::input_iterator_tag) [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.21 unique\_() [1/2]**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::unique_ () [inline], [private]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.22 prepend()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::prepend (
 const ex & b)
```

Add element at front.

#### 6.24.4.23 append()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::append (
 const ex & b)
```

Add element at back.

Referenced by [GiNaC::H\\_evalf\(\)](#), [GiNaC::ifactor\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::symm\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

#### 6.24.4.24 remove\_first()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_first ()
```

Remove first element.

Referenced by [GiNaC::sqrfree\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

#### 6.24.4.25 remove\_last()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_last ()
```

Remove last element.

#### 6.24.4.26 remove\_all()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::remove_all ()
```

Remove all elements.

#### 6.24.4.27 sort()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::sort ()
```

Sort elements.

#### 6.24.4.28 unique()

```
template<template< class T, class=std::allocator< T > > class C>
container< C > & GiNaC::container< C >::unique ()
```

Remove adjacent duplicate elements.

**6.24.4.29 begin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::begin () const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::symmetrize\(\)](#), [GiNaC::ex::symmetrize\\_cyclic\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

**6.24.4.30 end()**

```
template<template< class T, class=std::allocator< T > > class C>
const_iterator GiNaC::container< C >::end () const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::ex::antisymmetrize\(\)](#), [GiNaC::fderivative::archive\(\)](#), [GiNaC::ncmul::conjugate\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::ex::symmetrize\(\)](#), and [GiNaC::ex::symmetrize\\_cyclic\(\)](#).

**6.24.4.31 rbegin()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rbegin () const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.32 rend()**

```
template<template< class T, class=std::allocator< T > > class C>
const_reverse_iterator GiNaC::container< C >::rend () const [inline]
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.24.4.33 do\_print()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#).

**6.24.4.34 do\_print\_tree()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), and [GiNaC::nops\(\)](#).

**6.24.4.35 do\_print\_python()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python (
 const print_python & c,
 unsigned level) const [protected]
```

References [c](#).

**6.24.4.36 do\_print\_python\_repr()**

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container< C >::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#).

**6.24.4.37 subschildren()**

```
template<template< class T, class=std::allocator< T > > class C>
container< C >::STLT GiNaC::container< C >::subschildren (
 const exmap & m,
 unsigned options = 0) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [m](#), [options](#), and [GiNaC::ex::subs\(\)](#).

**6.24.4.38 unique\_() [2/2]**

```
void GiNaC::container< std::list >::unique_ () [inline], [private]
```

Specialization of [container::unique\\_\(\)](#) for `std::list`.

The documentation for this class was generated from the following files:

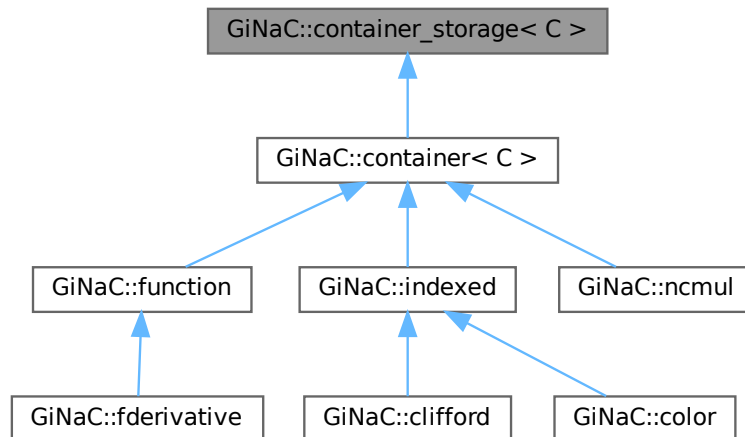
- [container.h](#)
- [exprseq.h](#)
- [lst.h](#)

## 6.25 GiNaC::container\_storage< C > Class Template Reference

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

```
#include <container.h>
```

Inheritance diagram for GiNaC::container\_storage< C >:



### Protected Types

- typedef C< [ex](#) > [STLT](#)

### Protected Member Functions

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

### Static Protected Member Functions

- static void [reserve](#) ([STLT](#) &, size\_t)

### Protected Attributes

- [STLT seq](#)

## 6.25.1 Detailed Description

```
template<template< class T, class=std::allocator< T > > class C>
class GiNaC::container_storage< C >
```

Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.

## 6.25.2 Member Typedef Documentation

### 6.25.2.1 STLT

```
template<template< class T, class=std::allocator< T > > class C>
typedef C<ex> GiNaC::container_storage< C >::STLT [protected]
```

## 6.25.3 Constructor & Destructor Documentation

### 6.25.3.1 container\_storage() [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage () [inline], [protected]
```

### 6.25.3.2 container\_storage() [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
 size_t n,
 const ex & e) [inline], [protected]
```

### 6.25.3.3 container\_storage() [3/4]

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::container_storage (
 std::initializer_list< ex > il) [inline], [protected]
```

### 6.25.3.4 container\_storage() [4/4]

```
template<template< class T, class=std::allocator< T > > class C>
template<class In >
GiNaC::container_storage< C >::container_storage (
 In b,
 In e) [inline], [protected]
```

### 6.25.3.5 ~container\_storage()

```
template<template< class T, class=std::allocator< T > > class C>
GiNaC::container_storage< C >::~~container_storage () [inline], [protected]
```

## 6.25.4 Member Function Documentation

### 6.25.4.1 `reserve()` [1/4]

```
template<template< class T, class=std::allocator< T > > class C>
void GiNaC::container_storage< C >::reserve (
 size_t) [inline], [protected]
```

Referenced by [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::container< C >::real\\_part\(\)](#), and [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#).

### 6.25.4.2 `reserve()` [2/4]

```
template<template< class T, class=std::allocator< T > > class C>
static void GiNaC::container_storage< C >::reserve (
 STL & ,
 size_t) [inline], [static], [protected]
```

### 6.25.4.3 `reserve()` [3/4]

```
void GiNaC::container_storage< std::vector >::reserve (
 size_t n) [inline], [protected]
```

References [n](#).

### 6.25.4.4 `reserve()` [4/4]

```
void GiNaC::container_storage< std::vector >::reserve (
 std::vector< ex > & v,
 size_t n) [inline], [protected]
```

References [n](#).

## 6.25.5 Member Data Documentation

### 6.25.5.1 `seq`

```
template<template< class T, class=std::allocator< T > > class C>
STLT GiNaC::container_storage< C >::seq [protected]
```

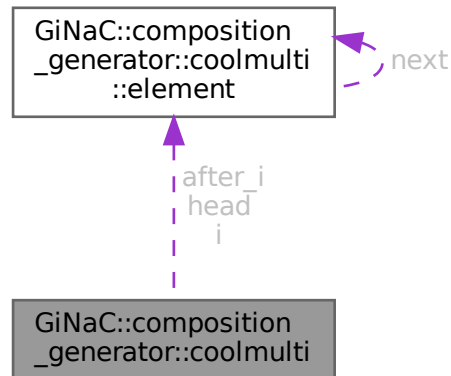
Referenced by [GiNaC::indexed::all\\_index\\_values\\_are\(\)](#), [GiNaC::container< C >::archive\(\)](#), [GiNaC::container< C >::begin\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::function::conjugate\(\)](#), [GiNaC::container< C >::container\(\)](#), [GiNaC::container< C >::container\(\)](#), [GiNaC::ncmul::degree\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::ncmul::derivative\(\)](#), [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::fderivative::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::function::eval\\_ncmul\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::indexed::get\\_dummy\\_indices\(\)](#), [GiNaC::ncmul::get\\_factors\(\)](#), [GiNaC::indexed::get\\_free\\_indices\(\)](#), [GiNaC::indexed::get\\_indices\(\)](#), [GiNaC::indexed::has\\_dummy\\_index\\_for\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::indexed::indexed\(\)](#), [GiNaC::function::info\(\)](#), [GiNaC::indexed::info\(\)](#), [GiNaC::remember\\_table\\_entry::is\\_equal\(\)](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::ncmul::ldegree\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::indexed::print\\_indexed\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::container< C >::rbegin\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::function::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::container< C >::rend\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::ncmul::return\\_type\\_tinfo\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::container< C >::sort\\_\(\)](#), [GiNaC::container< C >::sort\\_\(\)](#), [GiNaC::container< C >::unique\\_\(\)](#), and [GiNaC::indexed::validate\(\)](#).

The documentation for this class was generated from the following file:

- [container.h](#)

## 6.26 GiNaC::composition\_generator::coolmulti Struct Reference

Collaboration diagram for GiNaC::composition\_generator::coolmulti:



### Classes

- struct [element](#)

### Public Member Functions

- [coolmulti](#) (const std::vector< unsigned > &partition)
- [~coolmulti](#) ()
- void [next\\_permutation](#) ()
- bool [finished](#) () const

### Public Attributes

- [element](#) \* [head](#)
- [element](#) \* [i](#)
- [element](#) \* [after\\_i](#)

## 6.26.1 Constructor & Destructor Documentation

### 6.26.1.1 coolmulti()

```

GiNaC::composition_generator::coolmulti::coolmulti (
 const std::vector< unsigned > & partition) [inline], [explicit]

```

References [after\\_i](#), [head](#), [i](#), [n](#), and [GiNaC::composition\\_generator::coolmulti::element::next](#).

### 6.26.1.2 `~coolmulti()`

`GiNaC::composition_generator::coolmulti::~~coolmulti ( ) [inline]`

References [head](#).

## 6.26.2 Member Function Documentation

### 6.26.2.1 `next_permutation()`

`void GiNaC::composition_generator::coolmulti::next_permutation ( ) [inline]`

References [after\\_i](#), [head](#), [i](#), [k](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::next](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

### 6.26.2.2 `finished()`

`bool GiNaC::composition_generator::coolmulti::finished ( ) const [inline]`

References [after\\_i](#), [head](#), [GiNaC::composition\\_generator::coolmulti::element::next](#), and [GiNaC::composition\\_generator::coolmulti::element::next](#).

Referenced by [GiNaC::composition\\_generator::next\(\)](#).

## 6.26.3 Member Data Documentation

### 6.26.3.1 `head`

`element* GiNaC::composition_generator::coolmulti::head`

Referenced by [coolmulti\(\)](#), [finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [next\\_permutation\(\)](#), and [~coolmulti\(\)](#).

### 6.26.3.2 `i`

`element * GiNaC::composition_generator::coolmulti::i`

Referenced by [coolmulti\(\)](#), and [next\\_permutation\(\)](#).

### 6.26.3.3 `after_i`

`element * GiNaC::composition_generator::coolmulti::after_i`

Referenced by [coolmulti\(\)](#), [finished\(\)](#), and [next\\_permutation\(\)](#).

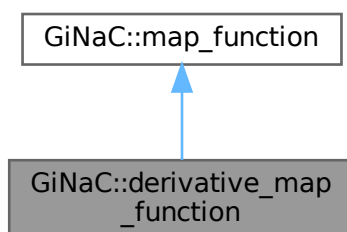
The documentation for this struct was generated from the following file:

- [utils.h](#)

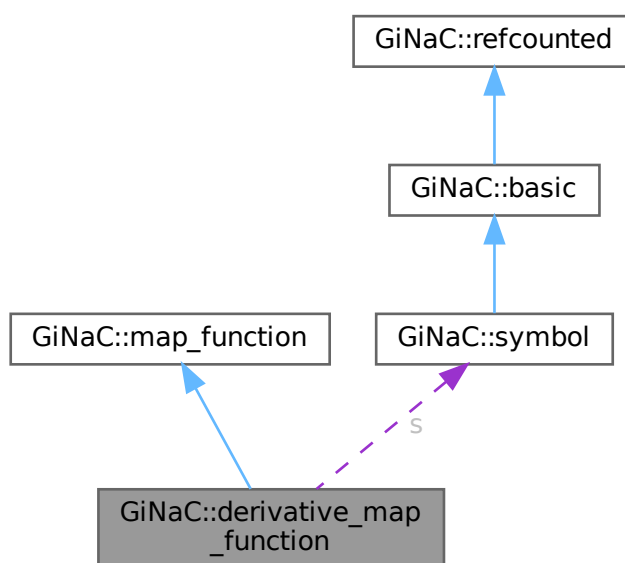
## 6.27 GiNaC::derivative\_map\_function Struct Reference

Function object to be applied by [basic::derivative\(\)](#).

Inheritance diagram for GiNaC::derivative\_map\_function:



Collaboration diagram for GiNaC::derivative\_map\_function:



### Public Member Functions

- [derivative\\_map\\_function](#) (const [symbol](#) &sym)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Public Attributes

- const [symbol](#) & [s](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.27.1 Detailed Description

Function object to be applied by [basic::derivative\(\)](#).

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 [derivative\\_map\\_function\(\)](#)

```
GiNaC::derivative_map_function::derivative_map_function (
 const symbol & sym) [inline]
```

### 6.27.3 Member Function Documentation

#### 6.27.3.1 [operator\(\)](#)

```
ex GiNaC::derivative_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::diff\(\)](#).

### 6.27.4 Member Data Documentation

#### 6.27.4.1 [s](#)

```
const symbol& GiNaC::derivative_map_function::s
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.28 GiNaC::determinant\_algo Class Reference

Switch to control algorithm for determinant computation.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#) , [gauss](#) , [divfree](#) , [laplace](#) ,  
[bareiss](#) }

### 6.28.1 Detailed Description

Switch to control algorithm for determinant computation.

### 6.28.2 Member Enumeration Documentation

#### 6.28.2.1 anonymous enum

anonymous enum

#### Enumerator

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| automatic | Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| gauss     | <p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>The determinant is then just the product of diagonal elements. Choose this algorithm only for purely numerical matrices.</p>                                                                                                                                                                                                                                                                                                                                                                                           |
| divfree   | <p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>The determinant can later be computed by inspecting the diagonal elements only. This algorithm is only there for the purpose of cross-checks. It is never fast.</p>                                                                                                                                                                                                                                      |
| laplace   | Laplace elimination. This is plain recursive elimination along minors although multiple minors are avoided by the algorithm. Although the algorithm is exponential in complexity it is frequently the fastest one when the matrix is populated by complicated symbolic expressions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| bareiss   | <p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. The determinant can then be read off from the lower right entry. This algorithm is rarely fast for</p> |

The documentation for this class was generated from the following file:

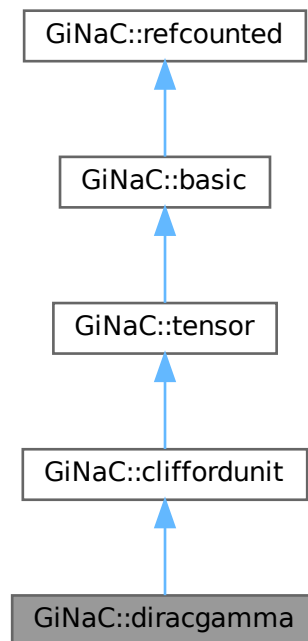
- [flags.h](#)

## 6.29 GiNaC::diracgamma Class Reference

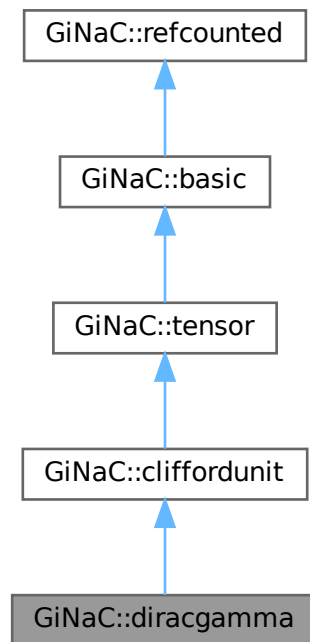
This class represents the Dirac gamma Lorentz vector.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma:



Collaboration diagram for GiNaC::diracgamma:



### Public Member Functions

- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of a gamma matrix with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*

- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex &self`, const `ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex &self`, const `numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info &ri`, const `print_context &c`, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node &n`) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node &n`, `lst &syms`)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap &m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol &s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic &other`) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic &other`) const  
*Test for syntactic equality.*
- const `basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::cliffordunit`

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.29.1 Detailed Description

This class represents the Dirac gamma Lorentz vector.

### 6.29.2 Member Function Documentation

#### 6.29.2.1 contract\_with()

```
bool GiNaC::diracgamma::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of a gamma matrix with something else.

Reimplemented from [GiNaC::cliffordunit](#).

#### 6.29.2.2 do\_print()

```
void GiNaC::diracgamma::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

#### 6.29.2.3 do\_print\_latex()

```
void GiNaC::diracgamma::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

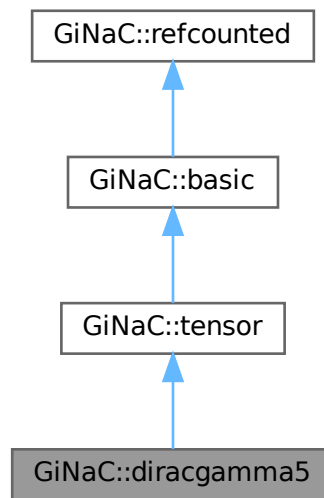
- [clifford.h](#)
- [clifford.cpp](#)

## 6.30 GiNaC::diracgamma5 Class Reference

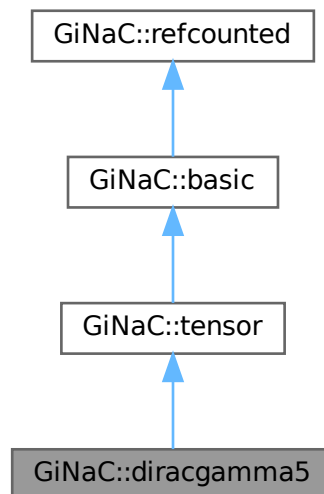
This class represents the Dirac gamma5 object which anticommutes with all other gammas.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgamma5:



Collaboration diagram for GiNaC::diracgamma5:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Member Functions

- [ex conjugate](#) () const override

## Additional Inherited Members

## Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) ([exvector::iterator](#) self, [exvector::iterator](#) other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.30.1 Detailed Description

This class represents the Dirac gamma5 object which anticommutes with all other gammas.

### 6.30.2 Member Function Documentation

#### 6.30.2.1 [conjugate\(\)](#)

```
ex GiNaC::diracgamma5::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

#### 6.30.2.2 [do\\_print\(\)](#)

```
void GiNaC::diracgamma5::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

#### 6.30.2.3 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracgamma5::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

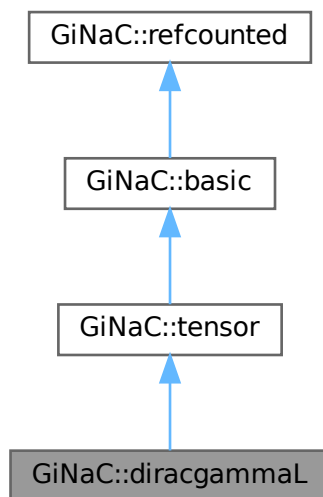
- [clifford.h](#)
- [clifford.cpp](#)

## 6.31 GiNaC::diracgammaL Class Reference

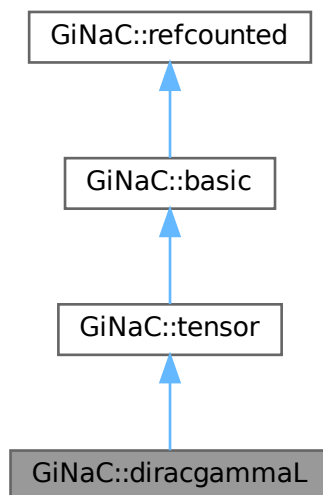
This class represents the Dirac gammaL object which behaves like  $1/2 (1-\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaL:



Collaboration diagram for GiNaC::diracgammaL:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Private Member Functions

- `ex conjugate` () const override

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
- void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*

  - const `basic` & `hold` () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.31.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2$  (1-gamma5).

### 6.31.2 Member Function Documentation

#### 6.31.2.1 [conjugate\(\)](#)

```
ex GiNaC::diracgammaL::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

#### 6.31.2.2 [do\\_print\(\)](#)

```
void GiNaC::diracgammaL::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

#### 6.31.2.3 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracgammaL::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

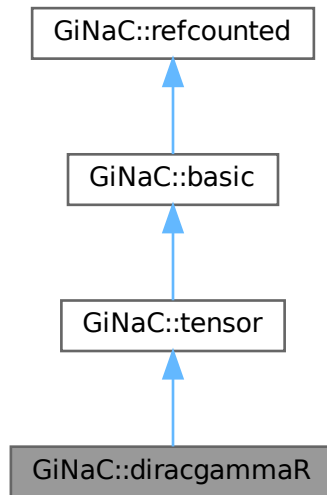
- [clifford.h](#)
- [clifford.cpp](#)

## 6.32 GiNaC::diracgammaR Class Reference

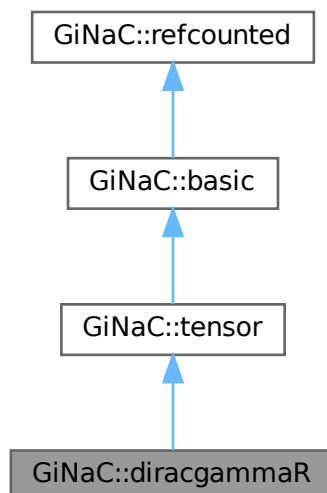
This class represents the Dirac gammaL object which behaves like  $\frac{1}{2}(1+\gamma_5)$ .

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracgammaR:



Collaboration diagram for GiNaC::diracgammaR:



**Protected Member Functions**

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

**Protected Member Functions inherited from `GiNaC::tensor`**

- unsigned `return_type` () const override

**Protected Member Functions inherited from `GiNaC::basic`**

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

**Private Member Functions**

- `ex conjugate` () const override

**Additional Inherited Members****Public Member Functions inherited from `GiNaC::tensor`**

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.32.1 Detailed Description

This class represents the Dirac gammaL object which behaves like  $1/2 (1+\gamma_5)$ .

### 6.32.2 Member Function Documentation

#### 6.32.2.1 [conjugate\(\)](#)

```
ex GiNaC::diracgammaR::conjugate () const [override], [private], [virtual]
```

Reimplemented from [GiNaC::basic](#).

#### 6.32.2.2 [do\\_print\(\)](#)

```
void GiNaC::diracgammaR::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

#### 6.32.2.3 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracgammaR::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

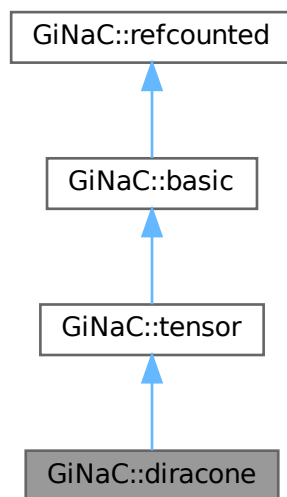
- [clifford.h](#)
- [clifford.cpp](#)

## 6.33 GiNaC::diracone Class Reference

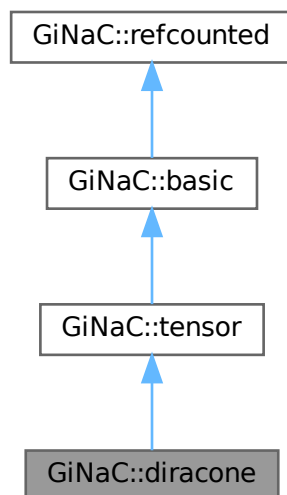
This class represents the Clifford algebra unity element.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracone:



Collaboration diagram for GiNaC::diracone:



### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (`exvector::iterator` self, `exvector::iterator` other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `bool info (unsigned inf) const`  
*Information about the object.*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`

- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)
- Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const

*Set some [status\\_flags](#).*

- const [basic](#) & [clearflag](#) (unsigned f) const

*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.33.1 Detailed Description

This class represents the Clifford algebra unity element.

### 6.33.2 Member Function Documentation

#### 6.33.2.1 [do\\_print\(\)](#)

```
void GiNaC::diracone::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

#### 6.33.2.2 [do\\_print\\_latex\(\)](#)

```
void GiNaC::diracone::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following file:

- [clifford.h](#)

## 6.34 GiNaC::do\_taylor Class Reference

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

```
#include <function.h>
```

### 6.34.1 Detailed Description

Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.

The documentation for this class was generated from the following file:

- [function.h](#)

## 6.35 GiNaC::domain Class Reference

Domain of an object.

```
#include <flags.h>
```

### Public Types

- enum { [complex](#) , [real](#) , [positive](#) }

### 6.35.1 Detailed Description

Domain of an object.

### 6.35.2 Member Enumeration Documentation

#### 6.35.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|          |  |
|----------|--|
| complex  |  |
| real     |  |
| positive |  |

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.36 GiNaC::dunno Class Reference

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

```
#include <utils.h>
```

### 6.36.1 Detailed Description

Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()

The documentation for this class was generated from the following file:

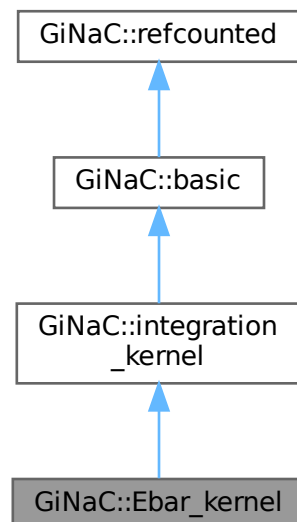
- [utils.h](#)

## 6.37 GiNaC::Ebar\_kernel Class Reference

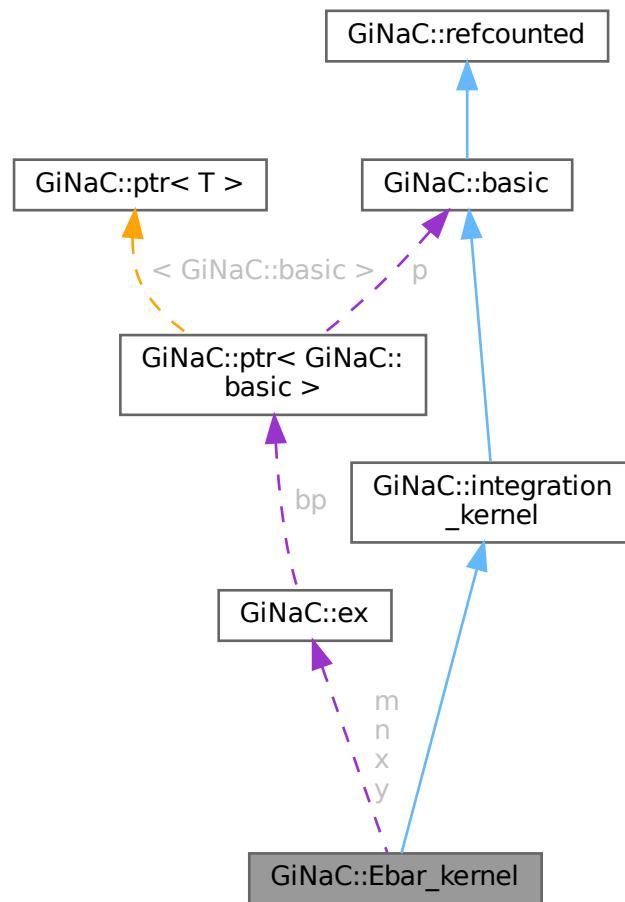
The Ebar-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Ebar\_kernel:



Collaboration diagram for GiNaC::Ebar\_kernel:



## Public Member Functions

- `Ebar_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$*

## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const
- *Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- `const basic & hold` () const
- *Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const
- *Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int `i`) const override
- *For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const
- *Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int `shift`, int `N_trunc`) const
- *The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.37.1 Detailed Description

The Ebar-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\bar{E}}(x; y) = \bar{E}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

## 6.37.2 Constructor & Destructor Documentation

### 6.37.2.1 Ebar\_kernel()

```
GiNaC::Ebar_kernel::Ebar_kernel (
 const ex & n,
 const ex & m,
 const ex & x,
 const ex & y)
```

## 6.37.3 Member Function Documentation

### 6.37.3.1 nops()

```
size_t GiNaC::Ebar_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.37.3.2 op()

```
ex GiNaC::Ebar_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.3 let\_op()

```
ex & GiNaC::Ebar_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.4 is\_numeric()

```
bool GiNaC::Ebar_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

### 6.37.3.5 `get_numerical_value()`

```
ex GiNaC::Ebar_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of  $Ebar_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

Referenced by [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

### 6.37.3.6 `series_coeff_impl()`

```
cln::cl_N GiNaC::Ebar_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [k](#), [m](#), [n](#), [x](#), and [y](#).

### 6.37.3.7 `do_print()`

```
void GiNaC::Ebar_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [m](#), [n](#), [GiNaC::ex::print\(\)](#), [x](#), and [y](#).

## 6.37.4 Member Data Documentation

### 6.37.4.1 `n`

```
ex GiNaC::Ebar_kernel::n [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.37.4.2 `m`

```
ex GiNaC::Ebar_kernel::m [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

**6.37.4.3 x**

```
ex GiNaC::Ebar_kernel::x [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

**6.37.4.4 y**

```
ex GiNaC::Ebar_kernel::y [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

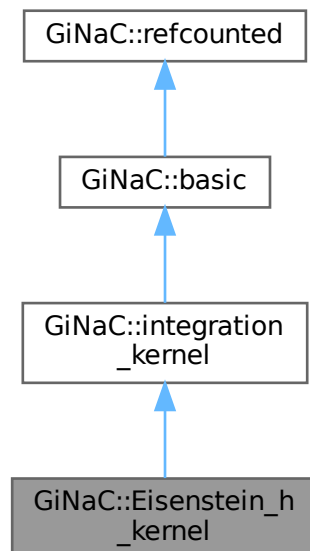
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

**6.38 GiNaC::Eisenstein\_h\_kernel Class Reference**

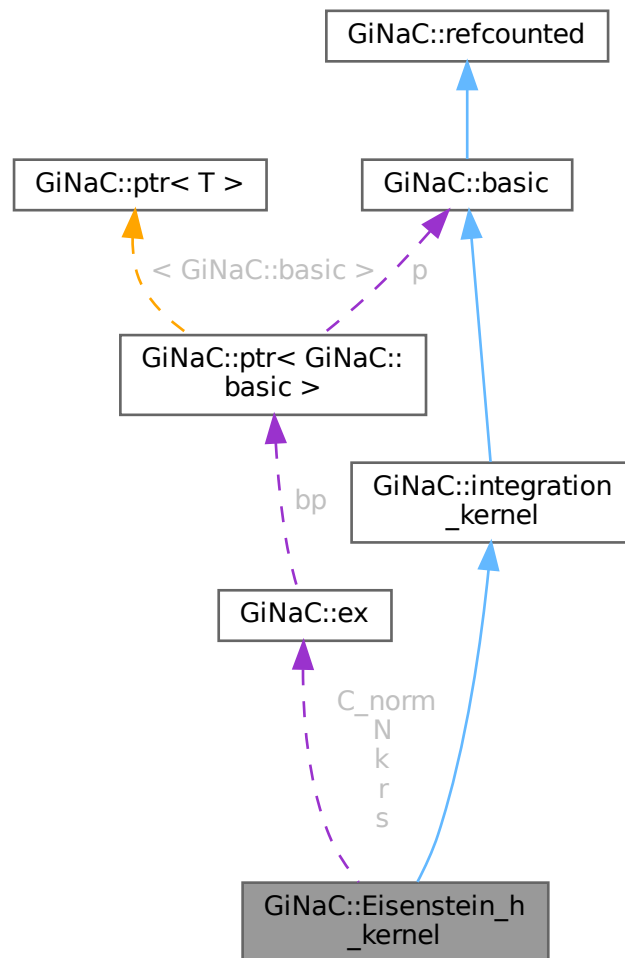
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_h\_kernel:



Collaboration diagram for `GiNaC::Eisenstein_h_kernel`:



## Public Member Functions

- `Eisenstein_h_kernel` (const `ex` &`k`, const `ex` &`N`, const `ex` &`r`, const `ex` &`s`, const `ex` &`C_norm=numeric(1)`)
- `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position  $i$ .*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position  $i$ .*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`x`, int `order`) const override

- Returns the Laurent series, starting possibly with the pole term.*

• `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override

*Returns the value of the modular form.*
- `ex coefficient_a0` (const `numeric` &`k`, const `numeric` &`r`, const `numeric` &`s`, const `numeric` &`N`) const

*The constant coefficient in the Fourier expansion.*
- `ex coefficient_an` (const `numeric` &`n`, const `numeric` &`k`, const `numeric` &`r`, const `numeric` &`s`, const `numeric` &`N`) const

*The higher coefficients in the Fourier expansion.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

## Public Member Functions inherited from GiNaC::integration\_kernel

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override

*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const

*This routine returns true, if the integration kernel has a trailing zero.*
- size\_t `get_cache_size` (void) const

*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const

*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const

*Wrapper around `series_coeff(i)`, converts `cl_N` to `numeric`.*
- `cln::cl_N series_coeff` (int `i`) const

*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic` ()

*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)

*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const

*Create a clone of this object on the heap.*
- virtual `ex eval` () const

*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const

*Evaluate object numerically.*
- virtual `ex evalm` () const

*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const

*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const

*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level`=0) const

*Output to stream.*
- virtual void `dbgprint` () const

*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const

- Little wrapper around printtree to be called within a debugger.*

  - virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*

  - virtual bool `info` (unsigned inf) const

*Information about the object.*

  - virtual `ex operator[]` (const `ex` &index) const
  - virtual `ex operator[]` (size\_t i) const
  - virtual `ex & operator[]` (const `ex` &index)
  - virtual `ex & operator[]` (size\_t i)
  - virtual bool `has` (const `ex` &other, unsigned `options`=0) const

*Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const

*Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const

- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex k](#)
- [ex N](#)
- [ex r](#)
- [ex s](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.38.1 Detailed Description

The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,r,s}^{\text{Eisenstein,h}} = C_k h_{k,N,r,s}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

## 6.38.2 Constructor & Destructor Documentation

### 6.38.2.1 Eisenstein\_h\_kernel()

```
GiNaC::Eisenstein_h_kernel::Eisenstein_h_kernel (
 const ex & k,
 const ex & N,
 const ex & r,
 const ex & s,
 const ex & C_norm = numeric(1))
```

## 6.38.3 Member Function Documentation

### 6.38.3.1 series()

```
ex GiNaC::Eisenstein_h_kernel::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::lhs\(\)](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), [GiNaC::ex::rhs\(\)](#), and [GiNaC::ex::series\(\)](#).

### 6.38.3.2 nops()

```
size_t GiNaC::Eisenstein_h_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.38.3.3 op()

```
ex GiNaC::Eisenstein_h_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), [N](#), [r](#), and [s](#).

#### 6.38.3.4 let\_op()

```
ex & GiNaC::Eisenstein_h_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), [N](#), [r](#), and [s](#).

#### 6.38.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_h_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [k](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [r](#), and [s](#).

#### 6.38.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_h_kernel::Laurent_series (
 const ex & x,
 int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

#### 6.38.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_h_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.38.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_h_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.38.3.9 coefficient\_a0()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_a0 (
 const numeric & k,
 const numeric & r,
 const numeric & s,
 const numeric & N) const
```

The constant coefficient in the Fourier expansion.

References [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::I](#), [GiNaC::irem\(\)](#), [k](#), [GiNaC::mod\(\)](#), [N](#), [GiNaC::Pi](#), [r](#), [s](#), and [GiNaC::sin\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.3.10 coefficient\_an()

```
ex GiNaC::Eisenstein_h_kernel::coefficient_an (
 const numeric & n,
 const numeric & k,
 const numeric & r,
 const numeric & s,
 const numeric & N) const
```

The higher coefficients in the Fourier expansion.

References [GiNaC::exp\(\)](#), [GiNaC::I](#), [GiNaC::irem\(\)](#), [k](#), [m](#), [GiNaC::mod\(\)](#), [n](#), [N](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [r](#), and [s](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.3.11 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_h_kernel::q_expansion_modular_form (
 const ex & q,
 int order) const
```

References [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [k](#), [N](#), [GiNaC::pow\(\)](#), [r](#), [s](#), and [GiNaC::ex::series\(\)](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 6.38.3.12 do\_print()

```
void GiNaC::Eisenstein_h_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [C\\_norm](#), [k](#), [N](#), [GiNaC::ex::print\(\)](#), [r](#), and [s](#).

## 6.38.4 Member Data Documentation

### 6.38.4.1 k

```
ex GiNaC::Eisenstein_h_kernel::k [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.4.2 N

```
ex GiNaC::Eisenstein_h_kernel::N [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.4.3 r

```
ex GiNaC::Eisenstein_h_kernel::r [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [q\\_expansion\\_modular\\_form\(\)](#), and [series\(\)](#).

### 6.38.4.4 s

```
ex GiNaC::Eisenstein_h_kernel::s [protected]
```

Referenced by [coefficient\\_a0\(\)](#), [coefficient\\_an\(\)](#), [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

### 6.38.4.5 C\_norm

```
ex GiNaC::Eisenstein_h_kernel::C_norm [protected]
```

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

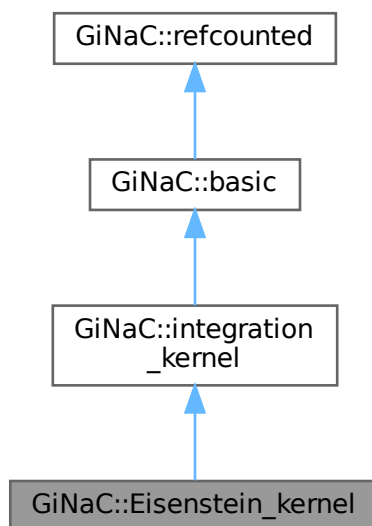
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.39 GiNaC::Eisenstein\_kernel Class Reference

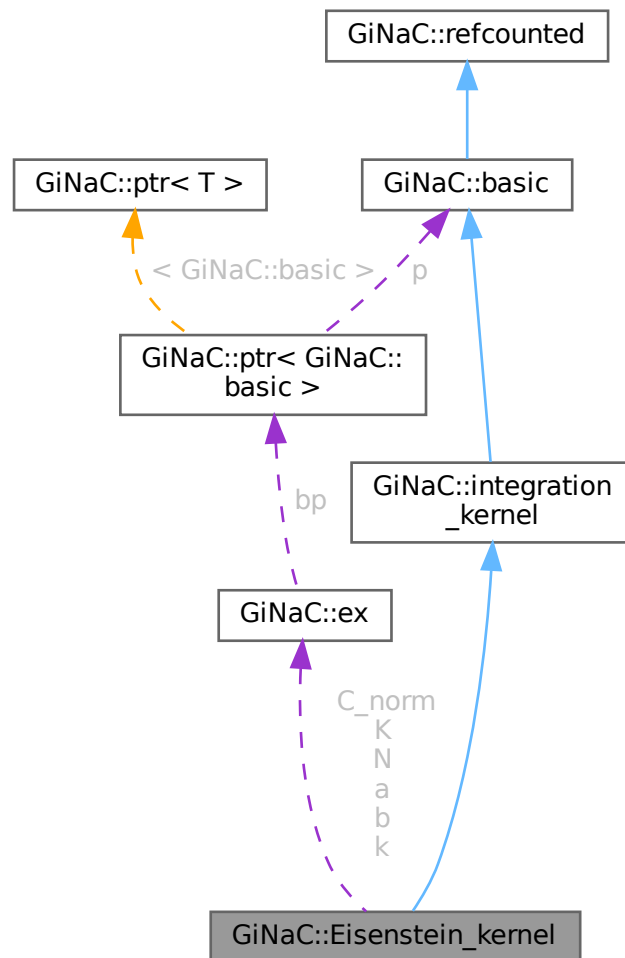
The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Eisenstein\_kernel:



Collaboration diagram for GiNaC::Eisenstein\_kernel:



## Public Member Functions

- `Eisenstein_kernel` (const `ex` &`k`, const `ex` &`N`, const `ex` &`a`, const `ex` &`b`, const `ex` &`K`, const `ex` &`C_norm`=numeric(1))
- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

- `ex Laurent_series` (const `ex` &`x`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc`=0) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- `size_t get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- `cln::cl_N series_coeff` (int `i`) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level`=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned `inf`) const

*Information about the object.*

- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const

*Test for occurrence of a pattern.*

- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*

- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*

- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

- virtual `ex coeff` (const `ex` &s, int `n`=1) const

*Return coefficient of degree n in object s.*

- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*

- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*

- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const

- template<class T >

void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex k](#)
- [ex N](#)
- [ex a](#)
- [ex b](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.39.1 Detailed Description

The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .

This class represents the differential one-form

$$\omega_{k,N,a,b,K}^{\text{Eisenstein}} = C_k E_{k,N,a,b,K}(\tau) \frac{d\bar{q}_N}{\bar{q}_N}$$

The integers a and b are either one or the discriminant of a quadratic number field. This class represents Eisenstein series, which can be defined by primitive Dirichlet characters from the Kronecker symbol. This implies that the characters take the values -1,0,1, i.e. no higher roots of unity occur. The

$$\bar{q}$$

-expansion has then rational coefficients.

Ref.: W. Stein, Modular Forms: A Computational Approach, Chapter 5

### 6.39.2 Constructor & Destructor Documentation

#### 6.39.2.1 Eisenstein\_kernel()

```
GiNaC::Eisenstein_kernel::Eisenstein_kernel (
 const ex & k,
 const ex & N,
 const ex & a,
 const ex & b,
 const ex & K,
 const ex & C_norm = numeric(1))
```

### 6.39.3 Member Function Documentation

#### 6.39.3.1 series()

```
ex GiNaC::Eisenstein_kernel::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.

This allows for easy use in the class [modular\\_form\\_kernel](#).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::lhs\(\)](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [r](#), and [GiNaC::ex::series\(\)](#).

### 6.39.3.2 nops()

```
size_t GiNaC::Eisenstein_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.39.3.3 op()

```
ex GiNaC::Eisenstein_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [k](#), [K](#), and [N](#).

### 6.39.3.4 let\_op()

```
ex & GiNaC::Eisenstein_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), [K](#), and [N](#).

### 6.39.3.5 is\_numeric()

```
bool GiNaC::Eisenstein_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [a](#), [b](#), [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [k](#), [K](#), [N](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), and [GiNaC::info\\_flags::posint](#).

### 6.39.3.6 Laurent\_series()

```
ex GiNaC::Eisenstein_kernel::Laurent_series (
 const ex & x,
 int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

### 6.39.3.7 get\_numerical\_value()

```
ex GiNaC::Eisenstein_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.39.3.8 uses\_Laurent\_series()

```
bool GiNaC::Eisenstein_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.39.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::Eisenstein_kernel::q_expansion_modular_form (
 const ex & q,
 int order) const
```

References [a](#), [b](#), [k](#), [K](#), [N](#), and [order](#).

Referenced by [Laurent\\_series\(\)](#), and [series\(\)](#).

### 6.39.3.10 do\_print()

```
void GiNaC::Eisenstein_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [a](#), [b](#), [c](#), [C\\_norm](#), [k](#), [K](#), [N](#), and [GiNaC::ex::print\(\)](#).

## 6.39.4 Member Data Documentation

### 6.39.4.1 k

```
ex GiNaC::Eisenstein_kernel::k [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**6.39.4.2 N**

`ex GiNaC::Eisenstein_kernel::N [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**6.39.4.3 a**

`ex GiNaC::Eisenstein_kernel::a [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**6.39.4.4 b**

`ex GiNaC::Eisenstein_kernel::b [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**6.39.4.5 K**

`ex GiNaC::Eisenstein_kernel::K [protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [q\\_expansion\\_modular\\_form\(\)](#).

**6.39.4.6 C\_norm**

`ex GiNaC::Eisenstein_kernel::C_norm [protected]`

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

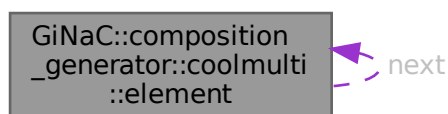
The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.40 GiNaC::composition\_generator::coolmulti::element Struct Reference

```
#include <utils.h>
```

Collaboration diagram for GiNaC::composition\_generator::coolmulti::element:



## Public Member Functions

- [element](#) (unsigned val, [element](#) \*n)
- [~element](#) ()

## Public Attributes

- unsigned [value](#)
- [element](#) \* [next](#)

## 6.40.1 Constructor & Destructor Documentation

### 6.40.1.1 [element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::element (
 unsigned val,
 element * n) [inline]
```

### 6.40.1.2 [~element\(\)](#)

```
GiNaC::composition_generator::coolmulti::element::~~element () [inline]
```

References [next](#).

## 6.40.2 Member Data Documentation

### 6.40.2.1 [value](#)

```
unsigned GiNaC::composition_generator::coolmulti::element::value
```

Referenced by [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), and [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#).

### 6.40.2.2 [next](#)

```
element* GiNaC::composition_generator::coolmulti::element::next
```

Referenced by [GiNaC::composition\\_generator::coolmulti::coolmulti\(\)](#), [GiNaC::composition\\_generator::coolmulti::finished\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GiNaC::composition\\_generator::coolmulti::next\\_permutation\(\)](#), and [~element\(\)](#).

The documentation for this struct was generated from the following file:

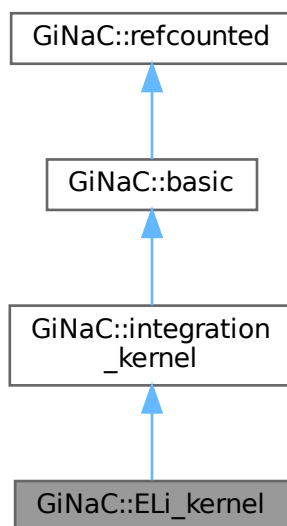
- [utils.h](#)

## 6.41 GiNaC::ELi\_kernel Class Reference

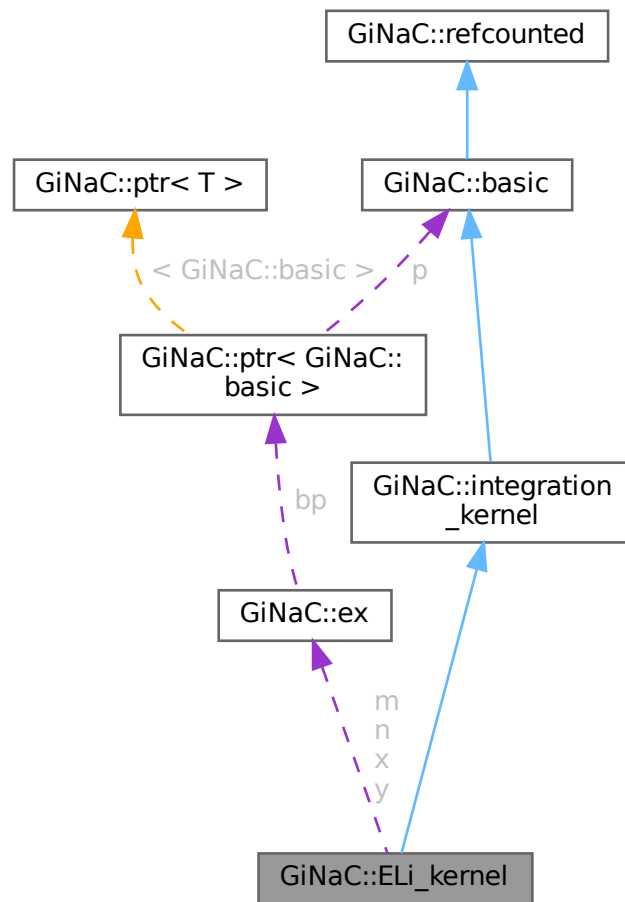
The ELi-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::ELi\_kernel:



Collaboration diagram for GiNaC::ELi\_kernel:



### Public Member Functions

- `ELi_kernel` (const `ex` &`n`, const `ex` &`m`, const `ex` &`x`, const `ex` &`y`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of  $ELi_{\{n,m\}}(x,y,qbar)$*

## Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const relational &r, int order, unsigned options=0) const override  
*Default implementation of `ex::series()`.*
- virtual bool `has_trailing_zero` (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual `ex Laurent_series` (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t `get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int i) const  
*Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.*
- cln::cl\_N `series_coeff` (int i) const  
*Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const basic &other)
- const `basic & operator=` (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual `ex operator[]` (const ex &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & operator[]` (const ex &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (GiNaC::visitor &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `cl::cl_N series_coeff_impl` (int `i`) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int `shift`, int `N_trunc`) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from GiNaC::basic

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex m](#)
- [ex x](#)
- [ex y](#)

## Protected Attributes inherited from GiNaC::integration\_kernel

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from GiNaC::basic

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.41.1 Detailed Description

The ELi-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\text{ELi}}(x; y) = \text{ELi}_{n;m}(x; y; \bar{q}) \frac{d\bar{q}}{\bar{q}}$$

## 6.41.2 Constructor & Destructor Documentation

### 6.41.2.1 ELi\_kernel()

```
GiNaC::ELi_kernel::ELi_kernel (
 const ex & n,
 const ex & m,
 const ex & x,
 const ex & y)
```

## 6.41.3 Member Function Documentation

### 6.41.3.1 nops()

```
size_t GiNaC::ELi_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.41.3.2 op()

```
ex GiNaC::ELi_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [m](#), [n](#), [x](#), and [y](#).

### 6.41.3.3 let\_op()

```
ex & GiNaC::ELi_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), [n](#), [x](#), and [y](#).

### 6.41.3.4 is\_numeric()

```
bool GiNaC::ELi_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [x](#), and [y](#).

### 6.41.3.5 get\_numerical\_value()

```
ex GiNaC::ELi_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of  $\text{ELi}_{\{n,m\}}(x,y,qbar)$

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.41.3.6 series\_coeff\_impl()

```
cln::cl_N GiNaC::ELi_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [k](#), [m](#), [n](#), [x](#), and [y](#).

### 6.41.3.7 do\_print()

```
void GiNaC::ELi_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [m](#), [n](#), [GiNaC::ex::print\(\)](#), [x](#), and [y](#).

## 6.41.4 Member Data Documentation

### 6.41.4.1 n

```
ex GiNaC::ELi_kernel::n [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.41.4.2 m

```
ex GiNaC::ELi_kernel::m [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 6.41.4.3 x

`ex` `GiNaC::ELi_kernel::x` `[protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 6.41.4.4 y

`ex` `GiNaC::ELi_kernel::y` `[protected]`

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.42 std::equal\_to< GiNaC::ex > Struct Reference

Specialization of `std::equal_to()` for `ex` objects.

```
#include <ex.h>
```

### Public Member Functions

- `bool` [operator\(\)](#) (const [GiNaC::ex](#) &`e1`, const [GiNaC::ex](#) &`e2`) const noexcept

### 6.42.1 Detailed Description

Specialization of `std::equal_to()` for `ex` objects.

### 6.42.2 Member Function Documentation

#### 6.42.2.1 operator()

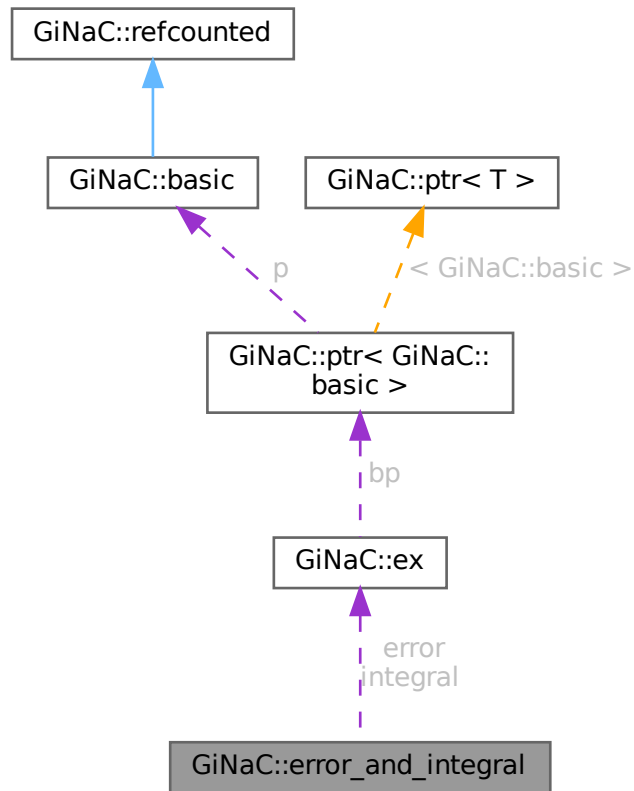
```
bool std::equal_to< GiNaC::ex >::operator() (
 const GiNaC::ex & e1,
 const GiNaC::ex & e2) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.43 GiNaC::error\_and\_integral Struct Reference

Collaboration diagram for GiNaC::error\_and\_integral:



### Public Member Functions

- [error\\_and\\_integral](#) (const [ex](#) &err, const [ex](#) &integ)

### Public Attributes

- [ex error](#)
- [ex integral](#)

### 6.43.1 Constructor & Destructor Documentation

#### 6.43.1.1 error\_and\_integral()

```

GiNaC::error_and_integral::error_and_integral (
 const ex & err,
 const ex & integ) [inline]

```

## 6.43.2 Member Data Documentation

### 6.43.2.1 error

`ex` `GiNaC::error_and_integral::error`

Referenced by [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)\(\)](#).

### 6.43.2.2 integral

`ex` `GiNaC::error_and_integral::integral`

Referenced by [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)\(\)](#).

The documentation for this struct was generated from the following file:

- [integral.cpp](#)

## 6.44 GiNaC::error\_and\_integral\_is\_less Struct Reference

### Public Member Functions

- `bool operator() (const error\_and\_integral &e1, const error\_and\_integral &e2) const`

### 6.44.1 Member Function Documentation

#### 6.44.1.1 operator()()

```
bool GiNaC::error_and_integral_is_less::operator() (
 const error_and_integral & e1,
 const error_and_integral & e2) const [inline]
```

References [c](#), [GiNaC::ex::compare\(\)](#), [GiNaC::error\\_and\\_integral::error](#), and [GiNaC::error\\_and\\_integral::integral](#).

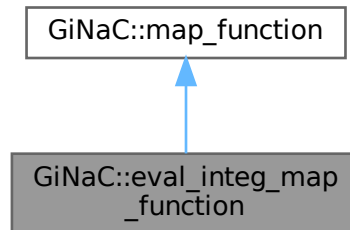
The documentation for this struct was generated from the following file:

- [integral.cpp](#)

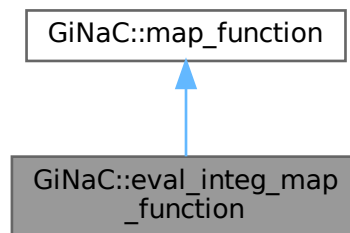
## 6.45 GiNaC::eval\_integ\_map\_function Struct Reference

Function object to be applied by [basic::eval\\_integ\(\)](#).

Inheritance diagram for GiNaC::eval\_integ\_map\_function:



Collaboration diagram for GiNaC::eval\_integ\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.45.1 Detailed Description

Function object to be applied by [basic::eval\\_integ\(\)](#).

### 6.45.2 Member Function Documentation

#### 6.45.2.1 operator()()

```
ex GiNaC::eval_integ_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::eval\\_integ\(\)](#).

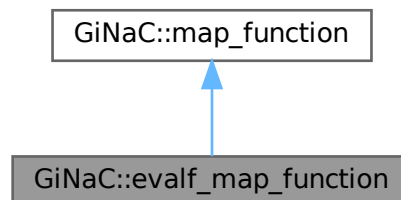
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

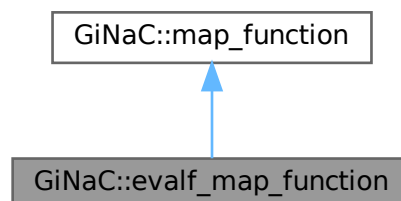
## 6.46 GiNaC::evalf\_map\_function Struct Reference

Function object to be applied by [basic::evalf\(\)](#).

Inheritance diagram for [GiNaC::evalf\\_map\\_function](#):



Collaboration diagram for [GiNaC::evalf\\_map\\_function](#):



## Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &*e*) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.46.1 Detailed Description

Function object to be applied by [basic::evalf\(\)](#).

### 6.46.2 Member Function Documentation

#### 6.46.2.1 [operator\(\)](#)

```
ex GiNaC::evalf_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalf\(\)](#).

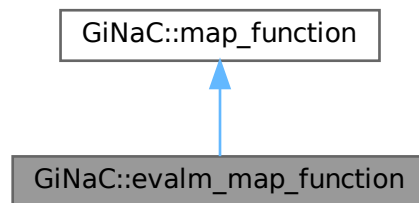
The documentation for this struct was generated from the following file:

- [basic.cpp](#)

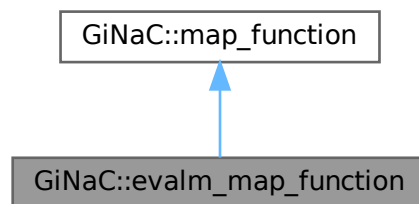
## 6.47 GiNaC::evalm\_map\_function Struct Reference

Function object to be applied by [basic::evalm\(\)](#).

Inheritance diagram for GiNaC::evalm\_map\_function:



Collaboration diagram for GiNaC::evalm\_map\_function:



### Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &[e](#)) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.47.1 Detailed Description

Function object to be applied by [basic::evalm\(\)](#).

### 6.47.2 Member Function Documentation

#### 6.47.2.1 operator>()

```
ex GiNaC::evalm_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::evalm\(\)](#).

The documentation for this struct was generated from the following file:

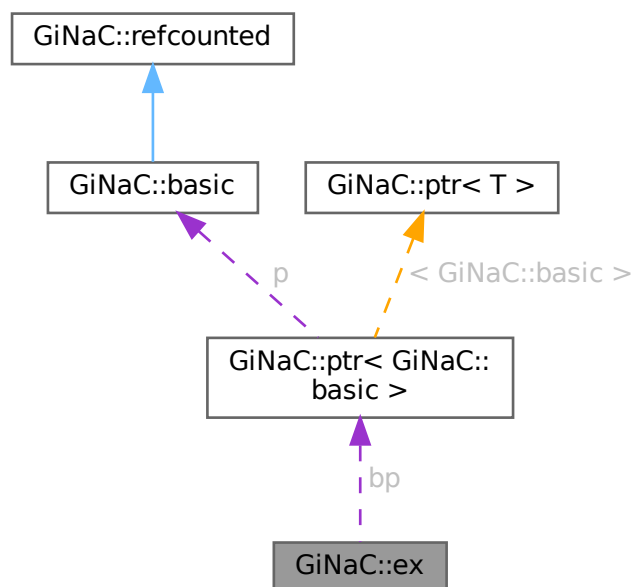
- [basic.cpp](#)

## 6.48 GiNaC::ex Class Reference

Lightweight wrapper for [GiNaC](#)'s symbolic objects.

```
#include <ex.h>
```

Collaboration diagram for GiNaC::ex:



## Public Member Functions

- `ex ()` noexcept
- `ex (const basic &other)`
- `ex (int i)`
- `ex (unsigned int i)`
- `ex (long i)`
- `ex (unsigned long i)`
- `ex (long long i)`
- `ex (unsigned long long i)`
- `ex (double const d)`
- `ex (const std::string &s, const ex &l)`  
*Construct ex from string and a list of symbols.*
- `void swap (ex &other)` noexcept  
*Efficiently swap the contents of two expressions.*
- `const_iterator begin ()` const noexcept
- `const_iterator end ()` const noexcept
- `const_preorder_iterator preorder_begin ()` const
- `const_preorder_iterator preorder_end ()` const noexcept
- `const_postorder_iterator postorder_begin ()` const
- `const_postorder_iterator postorder_end ()` const noexcept
- `ex eval ()` const
- `ex evalf ()` const
- `ex evalm ()` const
- `ex eval_ncmul (const exvector &v)` const
- `ex eval_integ ()` const
- `void print (const print_context &c, unsigned level=0)` const  
*Print expression to stream.*
- `void dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- `void dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- `bool info (unsigned inf)` const
- `size_t nops ()` const
- `ex op (size_t i)` const
- `ex operator[] (const ex &index)` const
- `ex operator[] (size_t i)` const
- `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `ex & operator[] (const ex &index)`
- `ex & operator[] (size_t i)`
- `ex lhs ()` const  
*Left hand side of relational expression.*
- `ex rhs ()` const  
*Right hand side of relational expression.*
- `ex conjugate ()` const
- `ex real_part ()` const
- `ex imag_part ()` const
- `bool has (const ex &pattern, unsigned options=0)` const
- `bool find (const ex &pattern, exset &found)` const  
*Find all occurrences of a pattern.*
- `bool match (const ex &pattern)` const  
*Check whether expression matches a specified pattern.*

- `bool match` (const `ex` &pattern, `exmap` &repls) const
- `ex subs` (const `exmap` &m, unsigned `options`=0) const
- `ex subs` (const `lst` &ls, const `lst` &lr, unsigned `options`=0) const  
*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex subs` (const `ex` &e, unsigned `options`=0) const  
*Substitute objects in an expression (syntactic substitution) and return the result as a new expression.*
- `ex map` (`map_function` &f) const
- `ex map` (`ex`(\*f)(const `ex` &e)) const
- `void accept` (`visitor` &v) const
- `void traverse_preorder` (`visitor` &v) const  
*Traverse expression tree with given visitor, preorder traversal.*
- `void traverse_postorder` (`visitor` &v) const  
*Traverse expression tree with given visitor, postorder traversal.*
- `void traverse` (`visitor` &v) const
- `bool is_polynomial` (const `ex` &vars) const  
*Check whether expression is a polynomial.*
- `int degree` (const `ex` &s) const
- `int ldegree` (const `ex` &s) const
- `ex coeff` (const `ex` &s, int `n`=1) const
- `ex lcoeff` (const `ex` &s) const
- `ex tcoeff` (const `ex` &s) const
- `ex expand` (unsigned `options`=0) const  
*Expand an expression.*
- `ex collect` (const `ex` &s, bool `distributed`=false) const
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Compute partial derivative of an expression.*
- `ex series` (const `ex` &r, int `order`, unsigned `options`=0) const  
*Compute the truncated series expansion of an expression.*
- `ex normal` () const  
*Normalization of rational functions.*
- `ex to_rational` (`exmap` &repl) const  
*Rationalization of non-rational functions.*
- `ex to_polynomial` (`exmap` &repl) const
- `ex numer` () const  
*Get numerator of an expression.*
- `ex denom` () const  
*Get denominator of an expression.*
- `ex numer_denom` () const  
*Get numerator and denominator of an expression.*
- `ex unit` (const `ex` &x) const  
*Compute unit part (= sign of leading coefficient) of a multivariate polynomial in  $Q[x]$ .*
- `ex content` (const `ex` &x) const  
*Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $Q[x]$ .*
- `numeric_integer_content` () const  
*Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.*
- `ex primpart` (const `ex` &x) const  
*Compute primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex primpart` (const `ex` &x, const `ex` &cont) const  
*Compute primitive part of a multivariate polynomial in  $Q[x]$  when the content part is already known.*
- `void unitcontprim` (const `ex` &x, `ex` &u, `ex` &c, `ex` &p) const  
*Compute unit part, content part, and primitive part of a multivariate polynomial in  $Q[x]$ .*
- `ex smod` (const `numeric` &xi) const

- `numeric max_coefficient () const`  
*Return maximum (absolute value) coefficient of a polynomial.*
- `exvector get_free_indices () const`
- `ex simplify_indexed (unsigned options=0) const`  
*Simplify/canonicalize expression containing indexed objects.*
- `ex simplify_indexed (const scalar_products &sp, unsigned options=0) const`  
*Simplify/canonicalize expression containing indexed objects.*
- `int compare (const ex &other) const`
- `bool is_equal (const ex &other) const`
- `bool is_zero () const`
- `bool is_zero_matrix () const`  
*Check whether expression is zero or zero matrix.*
- `ex symmetrize () const`  
*Symmetrize expression over its free indices.*
- `ex symmetrize (const lst &l) const`  
*Symmetrize expression over a list of objects (symbols, indices).*
- `ex antisymmetrize () const`  
*Antisymmetrize expression over its free indices.*
- `ex antisymmetrize (const lst &l) const`  
*Antisymmetrize expression over a list of objects (symbols, indices).*
- `ex symmetrize_cyclic () const`  
*Symmetrize expression by cyclic permutation over its free indices.*
- `ex symmetrize_cyclic (const lst &l) const`  
*Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).*
- `unsigned return_type () const`
- `return_type_t return_type_tinfo () const`
- `unsigned gethash () const`

### Private Member Functions

- `void makewritable ()`  
*Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.*
- `void share (const ex &other) const`  
*Share equal objects between expressions.*

### Static Private Member Functions

- `static ptr< basic > construct_from_basic (const basic &other)`  
*Helper function for the ex-from-basic constructor.*
- `static basic & construct_from_int (int i)`
- `static basic & construct_from_uint (unsigned int i)`
- `static basic & construct_from_long (long i)`
- `static basic & construct_from_ulong (unsigned long i)`
- `static basic & construct_from_longlong (long long i)`
- `static basic & construct_from_ulonglong (unsigned long long i)`
- `static basic & construct_from_double (double d)`
- `static ptr< basic > construct_from_string_and_lst (const std::string &s, const ex &l)`

## Private Attributes

- [ptr< basic > bp](#)  
*pointer to basic object managed by this*

## Friends

- class [archive\\_node](#)
- bool [are\\_ex\\_trivially\\_equal](#) (const [ex](#) &, const [ex](#) &)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- template<class T >  
const T & [ex\\_to](#) (const [ex](#) &)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- template<class T >  
bool [is\\_a](#) (const [ex](#) &)  
*Check if ex is a handle to a T, including base classes.*
- template<class T >  
bool [is\\_exactly\\_a](#) (const [ex](#) &)  
*Check if ex is a handle to a T, not including base classes.*

## 6.48.1 Detailed Description

Lightweight wrapper for [GiNaC](#)'s symbolic objects.

It holds a pointer to the other object in order to do garbage collection by the method of reference counting. I.e., it is a smart pointer. Also, the constructor [ex::ex\(const basic & other\)](#) calls the methods that do automatic evaluation. E.g., x-x turns automatically into 0.

## 6.48.2 Constructor & Destructor Documentation

### 6.48.2.1 [ex\(\)](#) [1/10]

```
GiNaC::ex::ex () [inline], [noexcept]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 6.48.2.2 [ex\(\)](#) [2/10]

```
GiNaC::ex::ex (
 const basic & other) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 6.48.2.3 [ex\(\)](#) [3/10]

```
GiNaC::ex::ex (
 int i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.4 ex()** [4/10]

```
GiNaC::ex::ex (
 unsigned int i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.5 ex()** [5/10]

```
GiNaC::ex::ex (
 long i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.6 ex()** [6/10]

```
GiNaC::ex::ex (
 unsigned long i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.7 ex()** [7/10]

```
GiNaC::ex::ex (
 long long i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.8 ex()** [8/10]

```
GiNaC::ex::ex (
 unsigned long long i) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

**6.48.2.9 ex()** [9/10]

```
GiNaC::ex::ex (
 double const d) [inline]
```

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

### 6.48.2.10 ex() [10/10]

```
GiNaC::ex::ex (
 const std::string & s,
 const ex & l) [inline]
```

Construct ex from string and a list of symbols.

The input grammar is similar to the [GiNaC](#) output format. All symbols and indices to be used in the expression must be specified in a list in the second argument. Undefined symbols and other parser errors will throw an exception.

References [bp](#), [GiNaC::status\\_flags::dynamallocated](#), and [GINAC\\_ASSERT](#).

## 6.48.3 Member Function Documentation

### 6.48.3.1 swap()

```
void GiNaC::ex::swap (
 ex & other) [inline], [noexcept]
```

Efficiently swap the contents of two expressions.

References [bp](#), [GiNaC::status\\_flags::dynamallocated](#), and [GINAC\\_ASSERT](#).

Referenced by [GiNaC::ncmul::derivative\(\)](#), [GiNaC::ex\\_swap::operator\(\)\(\)](#), [GiNaC::swap\(\)](#), [GiNaC::expair::swap\(\)](#), and [std::swap\(\)](#).

### 6.48.3.2 begin()

```
const_iterator GiNaC::ex::begin () const [inline], [noexcept]
```

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::antisymmetrize\(\)](#), [GiNaC::function::eval\\_ncmul\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [GiNaC::archive::printraw\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::symmetrize\(\)](#), [GiNaC::symmetrize\\_cyclic\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

### 6.48.3.3 end()

```
const_iterator GiNaC::ex::end () const [inline], [noexcept]
```

References [nops\(\)](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::log\\_expand\(\)](#), and [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#).

### 6.48.3.4 preorder\_begin()

```
const_preorder_iterator GiNaC::ex::preorder_begin () const [inline]
```

References [nops\(\)](#).

**6.48.3.5 preorder\_end()**

```
const_preorder_iterator GiNaC::ex::preorder_end () const [inline], [noexcept]
```

**6.48.3.6 postorder\_begin()**

```
const_postorder_iterator GiNaC::ex::postorder_begin () const [inline]
```

References [nops\(\)](#).

**6.48.3.7 postorder\_end()**

```
const_postorder_iterator GiNaC::ex::postorder_end () const [inline], [noexcept]
```

**6.48.3.8 eval()**

```
ex GiNaC::ex::eval () const [inline]
```

References [bp](#), and [eval\(\)](#).

Referenced by [eval\(\)](#), and [GiNaC::eval\(\)](#).

**6.48.3.9 evalf()**

```
ex GiNaC::ex::evalf () const [inline]
```

References [bp](#), and [evalf\(\)](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [evalf\(\)](#), [GiNaC::constant::evalf\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::power::evalf\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::is\\_numeric\(\)](#), [GiNaC::ELi\\_kernel::is\\_numeric\(\)](#), [GiNaC::Ebar\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::ELi\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Ebar\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::subsvalue\(\)](#).

**6.48.3.10 evalm()**

```
ex GiNaC::ex::evalm () const [inline]
```

References [bp](#), and [evalm\(\)](#).

Referenced by [evalm\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::power::evalm\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::evalm\(\)](#), and [is\\_zero\\_matrix\(\)](#).

**6.48.3.11 eval\_ncmul()**

```
ex GiNaC::ex::eval_ncmul (
 const exvector & v) const [inline]
```

References [bp](#), and [eval\\_ncmul\(\)](#).

Referenced by [eval\\_ncmul\(\)](#), [GiNaC::integral::eval\\_ncmul\(\)](#), and [GiNaC::relational::eval\\_ncmul\(\)](#).

**6.48.3.12 eval\_integ()**

```
ex GiNaC::ex::eval_integ () const [inline]
```

References [bp](#), and [eval\\_integ\(\)](#).

Referenced by [eval\\_integ\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), and [GiNaC::eval\\_integ\(\)](#).

**6.48.3.13 print()**

```
void GiNaC::ex::print (
 const print_context & c,
 unsigned level = 0) const
```

Print expression to stream.

The formatting of the output is determined by the kind of [print\\_context](#) object that is passed. Possible formattings include ginsh-parsable output (the default), tree-like output for debugging, and C++ source.

See also

[print\\_context](#)

References [bp](#), and [c](#).

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::operator<<\(\)](#), [GiNaC::expair::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), and [GiNaC::zeta1\\_print\\_latex\(\)](#).

### 6.48.3.14 dbgprint()

```
void GiNaC::ex::dbgprint () const
```

Little wrapper around print to be called within a debugger.

References [bp](#).

### 6.48.3.15 dbgprinttree()

```
void GiNaC::ex::dbgprinttree () const
```

Little wrapper around printtree to be called within a debugger.

References [bp](#).

### 6.48.3.16 info()

```
bool GiNaC::ex::info (
 unsigned int) const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::abs\\_info\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_info\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan2\\_info\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_info\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [content\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::csgn\\_series\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_info\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::func\\_arg\\_info\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::heur\\_gcd\(\)](#), [GiNaC::idx::idx\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::power::info\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::is\\_numeric\(\)](#), [GiNaC::ELi\\_kernel::is\\_numeric\(\)](#), [GiNaC::Ebar\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::is\\_numeric\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_kernel::is\\_numeric\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::power::is\\_polynomial\(\)](#), [GiNaC::iterated\\_integral2\\_eval\(\)](#), [GiNaC::iterated\\_integral3\\_eval\(\)](#), [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::lgamma\\_eval\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_imag\\_part\(\)](#), [GiNaC::log\\_info\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Order\\_imag\\_part\(\)](#), [GiNaC::Order\\_power\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::step\\_series\(\)](#), [subs\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::tgamma\\_series\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), [GiNaC::power::to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), [GiNaC::power::to\\_rational\(\)](#), [GiNaC::matrix::trace\(\)](#), [GiNaC::trig\\_info\(\)](#), [GiNaC::tryfactsubs\(\)](#), [unit\(\)](#), [unitcontprim\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), and [GiNaC::zeta2\\_eval\(\)](#).

### 6.48.3.17 nops()

```
size_t GiNaC::ex::nops () const [inline]
```

References [bp](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::divide\(\)](#), [end\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::nops\(\)](#), [postorder\\_begin\(\)](#), [preorder\\_begin\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::step\\_eval\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), and [GiNaC::zeta2\\_evalf\(\)](#).

### 6.48.3.18 op()

```
ex GiNaC::ex::op (
 size_t i) const [inline]
```

References [bp](#), and [op\(\)](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::mul::can\\_be\\_further\\_expanded\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::collect\\_symbols\(\)](#), [GiNaC::color\\_trace\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::decomp\\_rational\(\)](#), [denom\(\)](#), [GiNaC::const\\_postorder\\_iterator::descend\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [find\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::const\\_preorder\\_iterator::increment\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [GiNaC::op\(\)](#), [op\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)](#), [GiNaC::ex\\_base\\_is\\_less::operator\(\)](#), [GiNaC::const\\_iterator::operator\\*](#)(), [GiNaC::const\\_iterator::operator\[\]\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::spmapkey::spmapkey\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::step\\_eval\(\)](#), [subs\(\)](#), [GiNaC::symminfo::symminfo\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [traverse\\_postorder\(\)](#), [traverse\\_preorder\(\)](#), [GiNaC::tryfactsubs\(\)](#), and [GiNaC::zeta2\\_deriv\(\)](#).

### 6.48.3.19 operator[]() [1/4]

```
ex GiNaC::ex::operator[] (
 const ex & index) const [inline]
```

References [bp](#).

**6.48.3.20 operator[]() [2/4]**

```
ex GiNaC::ex::operator[] (
 size_t i) const [inline]
```

References [bp](#).

**6.48.3.21 let\_op()**

```
ex & GiNaC::ex::let_op (
 size_t i)
```

Return modifiable operand/member at position i.

References [bp](#), and [makewriteable\(\)](#).

Referenced by [GiNaC::H\\_evalf\(\)](#).

**6.48.3.22 operator[]() [3/4]**

```
ex & GiNaC::ex::operator[] (
 const ex & index)
```

References [bp](#), and [makewriteable\(\)](#).

**6.48.3.23 operator[]() [4/4]**

```
ex & GiNaC::ex::operator[] (
 size_t i)
```

References [bp](#), and [makewriteable\(\)](#).

**6.48.3.24 lhs()**

```
ex GiNaC::ex::lhs () const
```

Left hand side of relational expression.

References [bp](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::lhs\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::mul::series\(\)](#), and [GiNaC::power::series\(\)](#).

**6.48.3.25 rhs()**

```
ex GiNaC::ex::rhs () const
```

Right hand side of relational expression.

References [bp](#).

Referenced by [GiNaC::fsolve\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::rhs\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::symbol::series\(\)](#).

### 6.48.3.26 conjugate()

```
ex GiNaC::ex::conjugate () const [inline]
```

References [bp](#), and [conjugate\(\)](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [conjugate\(\)](#), [GiNaC::expair::conjugate\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::power::conjugate\(\)](#), [GiNaC::pseries::conjugate\(\)](#), [GiNaC::conjugate\(\)](#), [GiNaC::conjugate\\_eval\(\)](#), [GiNaC::conjugate\\_evalf\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::cos\\_conjugate\(\)](#), [GiNaC::cosh\\_conjugate\(\)](#), [GiNaC::exp\\_conjugate\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::sin\\_conjugate\(\)](#), [GiNaC::sinh\\_conjugate\(\)](#), [GiNaC::tan\\_conjugate\(\)](#), [GiNaC::tanh\\_conjugate\(\)](#), and [GiNaC::tgamma\\_conjugate\(\)](#).

### 6.48.3.27 real\_part()

```
ex GiNaC::ex::real_part () const [inline]
```

References [bp](#), and [real\\_part\(\)](#).

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::conjugate\\_real\\_part\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::pseries::imag\\_part\(\)](#), [real\\_part\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::pseries::real\\_part\(\)](#), [GiNaC::real\\_part\(\)](#), and [GiNaC::real\\_part\\_eval\(\)](#).

### 6.48.3.28 imag\_part()

```
ex GiNaC::ex::imag_part () const [inline]
```

References [bp](#), and [imag\\_part\(\)](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::conjugate\\_imag\\_part\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [imag\\_part\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::imag\\_part\(\)](#), [GiNaC::imag\\_part\\_eval\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::log\\_conjugate\(\)](#), and [GiNaC::power::real\\_part\(\)](#).

### 6.48.3.29 has()

```
bool GiNaC::ex::has (
 const ex & pattern,
 unsigned options = 0) const [inline]
```

References [bp](#), and [options](#).

Referenced by [GiNaC::power::degree\(\)](#), [GiNaC::integral::eval\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::has\(\)](#), [GiNaC::power::is\\_polynomial\(\)](#), and [GiNaC::power::ldegree\(\)](#).

### 6.48.3.30 find()

```
bool GiNaC::ex::find (
 const ex & pattern,
 exset & found) const
```

Find all occurrences of a pattern.

The found matches are appended to the "found" list. If the expression itself matches the pattern, the children are not further examined. This function returns true when any matches were found.

References [find\(\)](#), [match\(\)](#), [nops\(\)](#), and [op\(\)](#).

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::scalar\\_products::evaluate\(\)](#), [find\(\)](#), [GiNaC::find\(\)](#), [GiNaC::scalar\\_products::is\\_defined\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::idx::subs\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 6.48.3.31 match() [1/2]

```
bool GiNaC::ex::match (
 const ex & pattern) const
```

Check whether expression matches a specified pattern.

References [bp](#), and [match\(\)](#).

Referenced by [find\(\)](#), [GiNaC::power::has\(\)](#), [match\(\)](#), [GiNaC::basic::match\(\)](#), [GiNaC::match\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

### 6.48.3.32 match() [2/2]

```
bool GiNaC::ex::match (
 const ex & pattern,
 exmap & repls) const [inline]
```

References [bp](#).

### 6.48.3.33 subs() [1/3]

```
ex GiNaC::ex::subs (
 const exmap & m,
 unsigned options = 0) const [inline]
```

References [bp](#), [m](#), [options](#), and [subs\(\)](#).

Referenced by [GiNaC::adaptivesimpson\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::integral::conjugate\(\)](#), [GiNaC::csgn\\_series\(\)](#), [denom\(\)](#), [GiNaC::integral::derivative\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::integral::eval\\_integ\(\)](#), [GiNaC::integral::evalf\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::is\\_numeric\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::power::normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::tensor::replace\\_contr\\_index\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::step\\_series\(\)](#), [subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::subsvalue\(\)](#), [GiNaC::symm\(\)](#), [GiNaC::symmetrize\\_cyclic\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

**6.48.3.34 subs()** [2/3]

```
ex GiNaC::ex::subs (
 const lst & ls,
 const lst & lr,
 unsigned options = 0) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

References [GiNaC::container< C >::begin\(\)](#), [bp](#), [GiNaC::container< C >::end\(\)](#), [GINAC\\_ASSERT](#), [lr](#), [m](#), [GiNaC::container< C >::nops\(\)](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), and [GiNaC::subs\\_options::pattern\\_is\\_product](#).

**6.48.3.35 subs()** [3/3]

```
ex GiNaC::ex::subs (
 const ex & e,
 unsigned options = 0) const
```

Substitute objects in an expression (syntactic substitution) and return the result as a new expression.

There are two valid types of replacement arguments: 1) a relational like `object==ex` and 2) a list of relationals `lst{object1==ex1,object2==ex2,...}`.

References [bp](#), [GINAC\\_ASSERT](#), [info\(\)](#), [GiNaC::info\\_flags::list](#), [m](#), [op\(\)](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_product](#), [r](#), [GiNaC::info\\_flags::relation\\_equal](#), and [subs\(\)](#).

**6.48.3.36 map()** [1/2]

```
ex GiNaC::ex::map (
 map_function & f) const [inline]
```

References [bp](#), and [map\(\)](#).

Referenced by [GiNaC::color\\_trace\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), and [map\(\)](#).

**6.48.3.37 map()** [2/2]

```
ex GiNaC::ex::map (
 ex(*) (const ex &e) f) const
```

**6.48.3.38 accept()**

```
void GiNaC::ex::accept (
 visitor & v) const [inline]
```

References [bp](#).

Referenced by [traverse\\_postorder\(\)](#), and [traverse\\_preorder\(\)](#).

**6.48.3.39 traverse\_preorder()**

```
void GiNaC::ex::traverse_preorder (
 visitor & v) const
```

Traverse expression tree with given visitor, preorder traversal.

References [accept\(\)](#), [n](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_preorder\(\)](#).

Referenced by [traverse\(\)](#), and [traverse\\_preorder\(\)](#).

**6.48.3.40 traverse\_postorder()**

```
void GiNaC::ex::traverse_postorder (
 visitor & v) const
```

Traverse expression tree with given visitor, postorder traversal.

References [accept\(\)](#), [n](#), [nops\(\)](#), [op\(\)](#), and [traverse\\_postorder\(\)](#).

Referenced by [traverse\\_postorder\(\)](#).

**6.48.3.41 traverse()**

```
void GiNaC::ex::traverse (
 visitor & v) const [inline]
```

References [traverse\\_preorder\(\)](#).

**6.48.3.42 is\_polynomial()**

```
bool GiNaC::ex::is_polynomial (
 const ex & vars) const
```

Check whether expression is a polynomial.

References [bp](#).

Referenced by [GiNaC::is\\_polynomial\(\)](#), and [GiNaC::power::is\\_polynomial\(\)](#).

**6.48.3.43 degree()**

```
int GiNaC::ex::degree (
 const ex & s) const [inline]
```

References [bp](#).

Referenced by [GiNaC::basic::collect\(\)](#), [content\(\)](#), [GiNaC::integral::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::degree\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [lcoeff\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::sprem\(\)](#), and [GiNaC::sr\\_gcd\(\)](#).

### 6.48.3.44 ldegree()

```
int GiNaC::ex::ldegree (
 const ex & s) const [inline]
```

References [bp](#).

Referenced by [content\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::mul::ldegree\(\)](#), [GiNaC::power::ldegree\(\)](#), [GiNaC::ldegree\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [tcoeff\(\)](#).

### 6.48.3.45 coeff()

```
ex GiNaC::ex::coeff (
 const ex & s,
 int n = 1) const [inline]
```

References [bp](#), [coeff\(\)](#), and [n](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [coeff\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::basic::collect\(\)](#), [content\(\)](#), [GiNaC::pseries::convert\\_to\\_poly\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::eval\\_integ\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [lcoeff\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::integration\\_kernel::series\\_coeff\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqfree\\_parfrac\(\)](#), [GiNaC::expairseq::subchildren\(\)](#), and [tcoeff\(\)](#).

### 6.48.3.46 lcoeff()

```
ex GiNaC::ex::lcoeff (
 const ex & s) const [inline]
```

References [coeff\(\)](#), and [degree\(\)](#).

Referenced by [content\(\)](#), [GiNaC::get\\_symbol\\_stats\(\)](#), [GiNaC::sr\\_gcd\(\)](#), and [unit\(\)](#).

### 6.48.3.47 tcoeff()

```
ex GiNaC::ex::tcoeff (
 const ex & s) const [inline]
```

References [coeff\(\)](#), and [ldegree\(\)](#).

### 6.48.3.48 expand()

```
ex GiNaC::ex::expand (
 unsigned options = 0) const
```

Expand an expression.

## Parameters

|                |                                           |
|----------------|-------------------------------------------|
| <i>options</i> | see <a href="#">GiNaC::expand_options</a> |
|----------------|-------------------------------------------|

References [bp](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), and [options](#).

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::color\\_trace\(\)](#), [content\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::expand\(\)](#), [expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::expand\\_dummy\\_sum\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expand\\_map\\_function::operator\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::matrix::trace\(\)](#), [unit\(\)](#), and [unitcontprim\(\)](#).

**6.48.3.49 collect()**

```
ex GiNaC::ex::collect (
 const ex & s,
 bool distributed = false) const [inline]
```

References [bp](#), and [collect\(\)](#).

Referenced by [GiNaC::matrix::charpoly\(\)](#), [collect\(\)](#), and [GiNaC::collect\(\)](#).

**6.48.3.50 diff()**

```
ex GiNaC::ex::diff (
 const symbol & s,
 unsigned nth = 1) const
```

Compute partial derivative of an expression.

## Parameters

|            |                                           |
|------------|-------------------------------------------|
| <i>s</i>   | symbol by which the expression is derived |
| <i>nth</i> | order of derivative (default 1)           |

## Returns

partial derivative as a new expression

References [bp](#), and [diff\(\)](#).

Referenced by [GiNaC::abs\\_expl\\_derivative\(\)](#), [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::integral::derivative\(\)](#), [GiNaC::ncmul::derivative\(\)](#), [GiNaC::power::derivative\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::diff\(\)](#), [GiNaC::basic::diff\(\)](#), [diff\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_expl\\_derivative\(\)](#), [GiNaC::real\\_part\\_expl\\_derivative\(\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

**6.48.3.51 series()**

```
ex GiNaC::ex::series (
 const ex & r,
 int order,
 unsigned options = 0) const
```

Compute the truncated series expansion of an expression.

This function returns an expression containing an object of class pseries to represent the series. If the series does not terminate within the given truncation order, the last term of the series will be an order term.

**Parameters**

|                |                                                            |
|----------------|------------------------------------------------------------|
| <i>r</i>       | expansion relation, lhs holds variable and rhs holds point |
| <i>order</i>   | truncation order of series calculations                    |
| <i>options</i> | of class <a href="#">series_options</a>                    |

**Returns**

an expression holding a pseries object

References [GiNaC::\\_ex0](#), [bp](#), [options](#), [order](#), [r](#), and [series\(\)](#).

Referenced by [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::generalised\\_Bernoulli\\_nu](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_ser](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::S\\_series\(\)](#), [series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

**6.48.3.52 normal()**

```
ex GiNaC::ex::normal () const
```

Normalization of rational functions.

This function converts an expression to its normal form "numerator/denominator", where numerator and denominator are (relatively prime) polynomials. Any subexpressions which are not rational functions (like non-rational numbers, non-integer powers or functions like [sin\(\)](#), [cos\(\)](#) etc.) are replaced by temporary symbols which are re-substituted by the (normalized) subexpressions before [normal\(\)](#) returns (this way, any expression can be treated as a rational function). [normal\(\)](#) is applied recursively to arguments of functions etc.

**Returns**

normalized expression

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [normal\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [denom\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [normal\(\)](#), [GiNaC::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), and [GiNaC::matrix::trace\(\)](#).

### 6.48.3.53 to\_rational()

```
ex GiNaC::ex::to_rational (
 exmap & repl) const
```

Rationalization of non-rational functions.

This function converts a general expression to a rational function by replacing all non-rational subexpressions (like non-rational numbers, non-integer powers or functions like [sin\(\)](#), [cos\(\)](#) etc.) to temporary symbols. This makes it possible to use functions like [gcd\(\)](#) and [divide\(\)](#) on non-rational functions by applying [to\\_rational\(\)](#) on the arguments, calling the desired function and re-substituting the temporary symbols in the result. To make the last step possible, all temporary symbols and their associated expressions are collected in the map specified by the `repl` parameter, ready to be passed as an argument to [ex::subs\(\)](#).

#### Parameters

|             |                                                       |
|-------------|-------------------------------------------------------|
| <i>repl</i> | collects all temporary symbols and their replacements |
|-------------|-------------------------------------------------------|

#### Returns

rationalized expression

References [bp](#), and [to\\_rational\(\)](#).

Referenced by [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::to\\_rational\(\)](#), [to\\_rational\(\)](#), [GiNaC::expairseq::to\\_rational\(\)](#), and [GiNaC::power::to\\_rational\(\)](#).

### 6.48.3.54 to\_polynomial()

```
ex GiNaC::ex::to_polynomial (
 exmap & repl) const
```

References [bp](#), and [to\\_polynomial\(\)](#).

Referenced by [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::to\\_polynomial\(\)](#), [to\\_polynomial\(\)](#), [GiNaC::expairseq::to\\_polynomial\(\)](#), and [GiNaC::power::to\\_polynomial\(\)](#).

### 6.48.3.55 numer()

```
ex GiNaC::ex::numer () const
```

Get numerator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the numerator is returned.

#### See also

[ex::normal](#)

#### Returns

numerator

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [normal\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::numer\(\)](#), and [GiNaC::numer\(\)](#).

**6.48.3.56 denom()**

```
ex GiNaC::ex::denom () const
```

Get denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then the denominator is returned.

See also

[ex::normal](#)

Returns

denominator

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [normal\(\)](#), [op\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::denom\(\)](#), and [GiNaC::denom\(\)](#).

**6.48.3.57 numer\_denom()**

```
ex GiNaC::ex::numer_denom () const
```

Get numerator and denominator of an expression.

If the expression is not of the normal form "numerator/denominator", it is first converted to this form and then a list [numerator, denominator] is returned.

See also

[ex::normal](#)

Returns

a list [numerator, denominator]

References [bp](#), [GINAC\\_ASSERT](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [normal\(\)](#), [GiNaC::container< C >::op\(\)](#), and [subs\(\)](#).

Referenced by [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), and [GiNaC::numer\\_denom\(\)](#).

**6.48.3.58 unit()**

```
ex GiNaC::ex::unit (
 const ex & x) const
```

Compute unit part (= sign of leading coefficient) of a multivariate polynomial in Q[x].

The product of unit part, content part, and primitive part is the polynomial itself.

## Parameters

|                |               |
|----------------|---------------|
| <code>x</code> | main variable |
|----------------|---------------|

## Returns

unit part

## See also

[ex::content](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [c](#), [expand\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [info\(\)](#), [lcoeff\(\)](#), [GiNaC::info\\_flags::negative](#), [unit\(\)](#), and [x](#).

Referenced by [content\(\)](#), [GiNaC::frac\\_cancel\(\)](#), and [unit\(\)](#).

**6.48.3.59 content()**

```
ex GiNaC::ex::content (
 const ex & x) const
```

Compute content part (= unit normal GCD of all coefficients) of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of unit part, content part, and primitive part is the polynomial itself.

## Parameters

|                |               |
|----------------|---------------|
| <code>x</code> | main variable |
|----------------|---------------|

## Returns

content part

## See also

[ex::unit](#), [ex::primpart](#), [ex::unitcontprim](#)

References [GiNaC::\\_ex0](#), [c](#), [coeff\(\)](#), [cont](#), [degree\(\)](#), [expand\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [integer\\_content\(\)](#), [is\\_zero\(\)](#), [lcoeff\(\)](#), [ldegree\(\)](#), [GiNaC::info\\_flags::negative](#), [r](#), [unit\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#), and [unitcontprim\(\)](#).

**6.48.3.60 integer\_content()**

```
numeric GiNaC::ex::integer_content () const
```

Compute the integer content (= GCD of all numeric coefficients) of an expanded polynomial.

For a polynomial with rational coefficients, this returns  $g/l$  where  $g$  is the GCD of the coefficients' numerators and  $l$  is the LCM of the coefficients' denominators.

**Returns**

integer content

References [bp](#), and [GiNaC::numeric::integer\\_content\(\)](#).

Referenced by [content\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::gcd\(\)](#), and [GiNaC::heur\\_gcd\\_z\(\)](#).

**6.48.3.61 primpart() [1/2]**

```
ex GiNaC::ex::primpart (
 const ex & x) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The result will be a unit-normal polynomial with a content part of 1. The product of unit part, content part, and primitive part is the polynomial itself.

**Parameters**

|          |               |
|----------|---------------|
| <i>x</i> | main variable |
|----------|---------------|

**Returns**

primitive part

**See also**

[ex::unit](#), [ex::content](#), [ex::unitcontprim](#)

References [c](#), [unitcontprim\(\)](#), and [x](#).

Referenced by [GiNaC::sr\\_gcd\(\)](#).

**6.48.3.62 primpart() [2/2]**

```
ex GiNaC::ex::primpart (
 const ex & x,
 const ex & c) const
```

Compute primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$  when the content part is already known.

This function is faster in computing the primitive part than the previous function.

**Parameters**

|          |                                  |
|----------|----------------------------------|
| <i>x</i> | main variable                    |
| <i>c</i> | previously computed content part |

**Returns**

primitive part

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [is\\_zero\(\)](#), [GiNaC::quo\(\)](#), [unit](#), and [x](#).

**6.48.3.63 unitcontprim()**

```
void GiNaC::ex::unitcontprim (
 const ex & x,
 ex & u,
 ex & c,
 ex & p) const
```

Compute unit part, content part, and primitive part of a multivariate polynomial in  $\mathbb{Q}[x]$ .

The product of the three parts is the polynomial itself.

**Parameters**

|     |                           |
|-----|---------------------------|
| $x$ | main variable             |
| $u$ | unit part (returned)      |
| $c$ | content part (returned)   |
| $p$ | primitive part (returned) |

**See also**

[ex::unit](#), [ex::content](#), [ex::primpart](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::abs\(\)](#), [c](#), [content\(\)](#), [expand\(\)](#), [info\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::quo\(\)](#), [unit](#), and [x](#).

Referenced by [GiNaC::gcd\(\)](#), and [primpart\(\)](#).

**6.48.3.64 smod()**

```
ex GiNaC::ex::smod (
 const numeric & xi) const [inline]
```

References [bp](#), and [smod\(\)](#).

Referenced by [GiNaC::interpolate\(\)](#), and [smod\(\)](#).

**6.48.3.65 max\_coefficient()**

```
numeric GiNaC::ex::max_coefficient () const
```

Return maximum (absolute value) coefficient of a polynomial.

This function is used internally by [heur\\_gcd\(\)](#).

**Returns**

maximum coefficient

**See also**

[heur\\_gcd](#)

References [bp](#), and [GiNaC::numeric::max\\_coefficient\(\)](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

**6.48.3.66 get\_free\_indices()**

```
exvector GiNaC::ex::get_free_indices () const [inline]
```

References [bp](#).

Referenced by [antisymmetrize\(\)](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::integral::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::ncmul::get\\_free\\_indices\(\)](#), [GiNaC::clifford::get\\_metric\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [symmetrize\(\)](#), and [symmetrize\\_cyclic\(\)](#).

**6.48.3.67 simplify\_indexed() [1/2]**

```
ex GiNaC::ex::simplify_indexed (
 unsigned options = 0) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible and checks whether the free indices in sums are consistent.

**Parameters**

|                |                                           |
|----------------|-------------------------------------------|
| <i>options</i> | Simplification options (currently unused) |
|----------------|-------------------------------------------|

**Returns**

simplified expression

References [GiNaC::simplify\\_indexed\(\)](#).

Referenced by [GiNaC::tensepsilon::contract\\_with\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), and [GiNaC::simplify\\_indexed\(\)](#).

**6.48.3.68 simplify\_indexed() [2/2]**

```
ex GiNaC::ex::simplify_indexed (
 const scalar_products & sp,
 unsigned options = 0) const
```

Simplify/canonicalize expression containing indexed objects.

This performs contraction of dummy indices where possible, checks whether the free indices in sums are consistent, and automatically replaces scalar products by known values if desired.

## Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>sp</i>      | Scalar products to be replaced automatically |
| <i>options</i> | Simplification options (currently unused)    |

## Returns

simplified expression

References [GiNaC::simplify\\_indexed\(\)](#).

**6.48.3.69 compare()**

```
int GiNaC::ex::compare (
 const ex & other) const [inline]
```

References [bp](#), and [share\(\)](#).

Referenced by [GiNaC::expair::compare\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GiNaC::expair::is\\_less\(\)](#), [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#), [GiNaC::ex\\_is\\_less::operator\(\)](#), [GiNaC::expair\\_rest\\_is\\_less::operator\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)](#), [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)](#), [GiNaC::terminfo\\_is\\_less::operator\(\)](#), [GiNaC::spmapkey::operator<\(\)](#), and [GiNaC::spmapkey::spmapkey\(\)](#).

**6.48.3.70 is\_equal()**

```
bool GiNaC::ex::is_equal (
 const ex & other) const [inline]
```

References [bp](#), and [share\(\)](#).

Referenced by [GiNaC::abs\\_power\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\(\)](#), [GiNaC::mul::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::power::degree\(\)](#), [GiNaC::pseries::degree\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::epsilon\\_tensor\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::gcd\\_pf\\_pow\\_pow\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::expair::is\\_canonical\\_numeric\(\)](#), [GiNaC::pseries::is\\_compatible\\_to\(\)](#), [GiNaC::idx::is\\_dummy\\_pair\\_same\\_type\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [GiNaC::expairseq::is\\_equal\\_same\\_type\(\)](#), [is\\_zero\(\)](#), [GiNaC::power::lddegree\(\)](#), [GiNaC::pseries::lddegree\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::lorentz\\_eps\(\)](#), [GiNaC::expairseq::map\(\)](#), [GiNaC::expairseq::match\(\)](#), [GiNaC::idx::match\\_same\\_type\(\)](#), [GiNaC::minimal\\_dim\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::power::normal\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::ex\\_is\\_equal::operator\(\)](#), [GiNaC::op0\\_is\\_equal::operator\(\)](#), [GiNaC::idx\\_is\\_equal\\_ignore\\_dim::operator\(\)](#), [GiNaC::spmapkey::operator==\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::mul::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::add::split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

**6.48.3.71 is\_zero()**

```
bool GiNaC::ex::is_zero () const [inline]
```

References [GiNaC::\\_ex0](#), and [is\\_equal\(\)](#).

Referenced by [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::pseries::add\\_series\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::add::coeff\(\)](#), [content\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::function::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::pseries::evalm\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::is\\_zero\(\)](#), [is\\_zero\\_matrix\(\)](#), [GiNaC::add::ldegree\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li3\\_eval\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::relational::operator safe\\_bool\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::prem\(\)](#), [primpart\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::clifford::same\\_metric\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_yun\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [unitcontprim\(\)](#), and [GiNaC::indexed::validate\(\)](#).

**6.48.3.72 is\_zero\_matrix()**

```
bool GiNaC::ex::is_zero_matrix () const
```

Check whether expression is zero or zero matrix.

References [evalm\(\)](#), and [is\\_zero\(\)](#).

**6.48.3.73 symmetrize() [1/2]**

```
ex GiNaC::ex::symmetrize () const
```

Symmetrize expression over its free indices.

References [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\(\)](#).

Referenced by [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\(\)](#).

**6.48.3.74 symmetrize() [2/2]**

```
ex GiNaC::ex::symmetrize (
 const lst & l) const
```

Symmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symm\(\)](#).

**6.48.3.75 antisymmetrize()** [1/2]

```
ex GiNaC::ex::antisymmetrize () const
```

Antisymmetrize expression over its free indices.

References [GiNaC::antisymmetrize\(\)](#), and [get\\_free\\_indices\(\)](#).

Referenced by [GiNaC::antisymmetrize\(\)](#), and [GiNaC::antisymmetrize\(\)](#).

**6.48.3.76 antisymmetrize()** [2/2]

```
ex GiNaC::ex::antisymmetrize (
 const lst & l) const
```

Antisymmetrize expression over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symm\(\)](#).

**6.48.3.77 symmetrize\_cyclic()** [1/2]

```
ex GiNaC::ex::symmetrize_cyclic () const
```

Symmetrize expression by cyclic permutation over its free indices.

References [get\\_free\\_indices\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

Referenced by [GiNaC::symmetrize\\_cyclic\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.48.3.78 symmetrize\_cyclic()** [2/2]

```
ex GiNaC::ex::symmetrize_cyclic (
 const lst & l) const
```

Symmetrize expression by cyclic permutation over a list of objects (symbols, indices).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::end\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

**6.48.3.79 return\_type()**

```
unsigned GiNaC::ex::return_type () const [inline]
```

References [bp](#).

Referenced by [GiNaC::ncmul::append\\_factors\(\)](#), [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exmul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::indexed::return\\_type\(\)](#), [GiNaC::integral::return\\_type\(\)](#), [GiNaC::power::return\\_type\(\)](#), and [GiNaC::relational::return\\_type\(\)](#).

**6.48.3.80 return\_type\_tinfo()**

```
return_type_t GiNaC::ex::return_type_tinfo () const [inline]
```

References [bp](#).

Referenced by [GiNaC::color\\_trace\(\)](#), [GiNaC::indexed::return\\_type\\_tinfo\(\)](#), [GiNaC::integral::return\\_type\\_tinfo\(\)](#), [GiNaC::power::return\\_type\\_tinfo\(\)](#), and [GiNaC::relational::return\\_type\\_tinfo\(\)](#).

**6.48.3.81 gethash()**

```
unsigned GiNaC::ex::gethash () const [inline]
```

References [bp](#).

Referenced by [GiNaC::basic::calchash\(\)](#), [GiNaC::expairseq::calchash\(\)](#), [GiNaC::function::calchash\(\)](#), [GiNaC::idx::calchash\(\)](#), and [GiNaC::relational::calchash\(\)](#).

**6.48.3.82 construct\_from\_basic()**

```
ptr< basic > GiNaC::ex::construct_from_basic (
 const basic & other) [static], [private]
```

Helper function for the ex-from-basic constructor.

This is where [GiNaC](#)'s automatic evaluator and memory management are implemented.

See also

[ex::ex\(const basic &\)](#)

References [bp](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::basic::eval\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::refcounted::get\\_refcount\(\)](#), [GINAC\\_ASSERT](#), and [GiNaC::basic::setflag\(\)](#).

**6.48.3.83 construct\_from\_int()**

```
basic & GiNaC::ex::construct_from_int (
 int i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), and [GiNaC::\\_num\\_9\\_p](#).

Referenced by [construct\\_from\\_longlong\(\)](#).

#### 6.48.3.84 `construct_from_uint()`

```
basic & GiNaC::ex::construct_from_uint (
 unsigned int i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), and [GiNaC::\\_num9\\_p](#).

Referenced by [construct\\_from\\_ulonglong\(\)](#).

#### 6.48.3.85 `construct_from_long()`

```
basic & GiNaC::ex::construct_from_long (
 long i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), and [GiNaC::\\_num\\_9\\_p](#).

#### 6.48.3.86 `construct_from_ulong()`

```
basic & GiNaC::ex::construct_from_ulong (
 unsigned long i) [static], [private]
```

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), and [GiNaC::\\_num9\\_p](#).

#### 6.48.3.87 `construct_from_longlong()`

```
basic & GiNaC::ex::construct_from_longlong (
 long long i) [static], [private]
```

References [construct\\_from\\_int\(\)](#).

#### 6.48.3.88 `construct_from_ulonglong()`

```
basic & GiNaC::ex::construct_from_ulonglong (
 unsigned long long i) [static], [private]
```

References [construct\\_from\\_uint\(\)](#).

#### 6.48.3.89 `construct_from_double()`

```
basic & GiNaC::ex::construct_from_double (
 double d) [static], [private]
```

**6.48.3.90 construct\_from\_string\_and\_lst()**

```
static ptr< basic > GiNaC::ex::construct_from_string_and_lst (
 const std::string & s,
 const ex & l) [static], [private]
```

**6.48.3.91 makewriteable()**

```
void GiNaC::ex::makewriteable () [private]
```

Make this ex writable (if more than one ex handle the same basic) by unlinking the object and creating an unshared copy of it.

References [bp](#), [GiNaC::status\\_flags::dynallocated](#), and [GINAC\\_ASSERT](#).

Referenced by [let\\_op\(\)](#), [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

**6.48.3.92 share()**

```
void GiNaC::ex::share (
 const ex & other) const [private]
```

Share equal objects between expressions.

See also

[ex::compare\(const ex &\)](#)

References [bp](#), and [GiNaC::status\\_flags::not\\_shareable](#).

Referenced by [compare\(\)](#), and [is\\_equal\(\)](#).

**6.48.4 Friends And Related Symbol Documentation****6.48.4.1 archive\_node**

```
friend class archive_node [friend]
```

**6.48.4.2 are\_ex\_trivially\_equal**

```
bool are_ex_trivially_equal (
 const ex & e1,
 const ex & e2) [friend]
```

Compare two objects of class quickly without doing a deep tree traversal.

Returns

"true" if they are equal "false" if equality cannot be established quickly (e1 and e2 may still be equal, in this case).

**6.48.4.3 ex\_to**

```
template<class T >
const T & ex_to (
 const ex & e) [friend]
```

Return a reference to the basic-derived class T object embedded in an expression.

This is fast but unsafe: the result is undefined if the expression does not contain a T object at its top level. Hence, you should generally check the type of e first. Also, you shouldn't cache the returned reference because [GiNaC's](#) garbage collector may destroy the referenced object any time it's used in another expression.

## Parameters

|          |            |
|----------|------------|
| <i>e</i> | expression |
|----------|------------|

## Returns

reference to object of class T

## See also

[is\\_exactly\\_a<class T>\(\)](#)

## 6.48.4.4 is\_a

```
template<class T >
bool is_a (
 const ex & obj) [friend]
```

Check if *ex* is a handle to a T, including base classes.

## 6.48.4.5 is\_exactly\_a

```
template<class T >
bool is_exactly_a (
 const ex & obj) [friend]
```

Check if *ex* is a handle to a T, not including base classes.

## 6.48.5 Member Data Documentation

## 6.48.5.1 bp

```
ptr<basic> GiNaC::ex::bp [mutable], [private]
```

pointer to basic object managed by this

Referenced by [accept\(\)](#), [GiNaC::archive\\_node::archive\\_node\(\)](#), [coeff\(\)](#), [collect\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [construct\\_from\\_basic\(\)](#), [dbgprint\(\)](#), [dbgprinttree\(\)](#), [degree\(\)](#), [denom\(\)](#), [diff\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [ex\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [gethash\(\)](#), [has\(\)](#), [GiNaC::archive\\_node::has\\_same\\_ex\\_as\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [is\\_equal\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [lhs\(\)](#), [makewriteable\(\)](#), [map\(\)](#), [match\(\)](#), [match\(\)](#), [max\\_coefficient\(\)](#), [nops\(\)](#), [normal\(\)](#), [numer\(\)](#), [numer\\_denom\(\)](#), [op\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [print\(\)](#), [GiNaC::archive\\_node::printraw\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), [series\(\)](#), [share\(\)](#), [smod\(\)](#), [subs\(\)](#), [subs\(\)](#), [subs\(\)](#), [swap\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [ex.cpp](#)
- [indexed.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symmetry.cpp](#)

## 6.49 GiNaC::ex\_base\_is\_less Struct Reference

### Public Member Functions

- `bool operator()` (const `ex` &lh, const `ex` &rh) const

### 6.49.1 Member Function Documentation

#### 6.49.1.1 operator()

```
bool GiNaC::ex_base_is_less::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.50 GiNaC::ex\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- `bool operator()` (const `ex` &lh, const `ex` &rh) const

### 6.50.1 Member Function Documentation

#### 6.50.1.1 operator()

```
bool GiNaC::ex_is_equal::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.51 GiNaC::ex\_is\_less Struct Reference

```
#include <ex.h>
```

## Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

## 6.51.1 Member Function Documentation

### 6.51.1.1 operator()

```
bool GiNaC::ex_is_less::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GiNaC::ex::compare\(\)](#).

The documentation for this struct was generated from the following file:

- [ex.h](#)

## 6.52 GiNaC::ex\_swap Struct Reference

```
#include <ex.h>
```

## Public Member Functions

- void [operator\(\)](#) ([ex](#) &lh, [ex](#) &rh) const

## 6.52.1 Member Function Documentation

### 6.52.1.1 operator()

```
void GiNaC::ex_swap::operator() (
 ex & lh,
 ex & rh) const [inline]
```

References [GiNaC::ex::swap\(\)](#).

The documentation for this struct was generated from the following file:

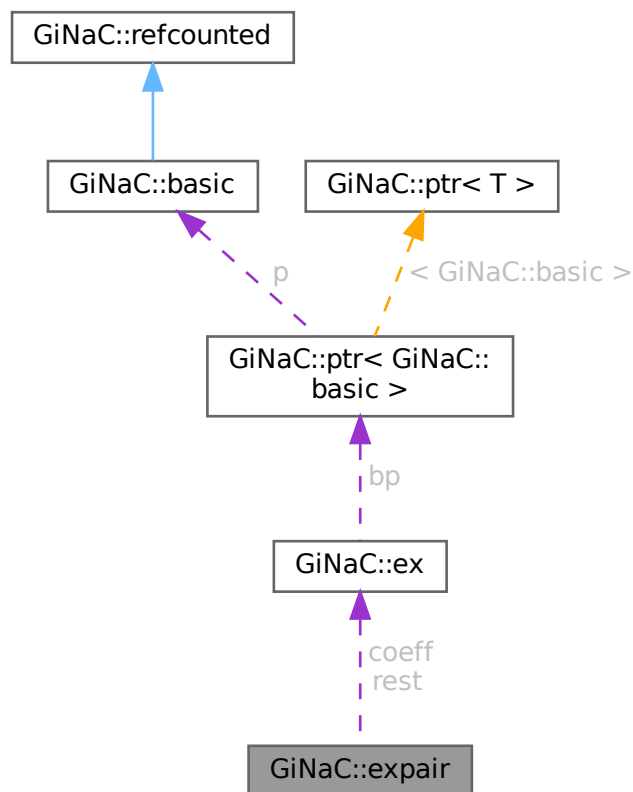
- [ex.h](#)

## 6.53 GiNaC::expair Class Reference

A pair of expressions.

```
#include <expair.h>
```

Collaboration diagram for GiNaC::expair:



### Public Member Functions

- `expair ()`
- `expair (const ex &r, const ex &c)`  
*Construct an expair from two ex.*
- `bool is_equal (const expair &other) const`  
*Member-wise check for canonical ordering equality.*
- `bool is_less (const expair &other) const`  
*Member-wise check for canonical ordering lessness.*
- `int compare (const expair &other) const`  
*Member-wise check for canonical ordering.*
- `void print (std::ostream &os) const`
- `bool is_canonical_numeric () const`  
*True if this is of the form (numeric,ex(1)).*
- `void swap (expair &other)`  
*Swap contents with other expair.*
- `const expair conjugate () const`

## Public Attributes

- [ex rest](#)  
*first member of pair, an arbitrary expression*
- [ex coeff](#)  
*second member of pair, must be numeric*

### 6.53.1 Detailed Description

A pair of expressions.

This is similar to STL's `pair<>`. It is slightly extended since we need to account for methods like [.compare\(\)](#). Also, since this is meant for use by class `expairseq` it must satisfy the invariance that the member `coeff` must be of type `numeric`.

### 6.53.2 Constructor & Destructor Documentation

#### 6.53.2.1 `expair()` [1/2]

```
GiNaC::expair::expair () [inline]
```

Referenced by [conjugate\(\)](#).

#### 6.53.2.2 `expair()` [2/2]

```
GiNaC::expair::expair (
 const ex & r,
 const ex & c) [inline]
```

Construct an `expair` from two `ex`.

References [coeff](#), and [GINAC\\_ASSERT](#).

### 6.53.3 Member Function Documentation

#### 6.53.3.1 `is_equal()`

```
bool GiNaC::expair::is_equal (
 const expair & other) const [inline]
```

Member-wise check for canonical ordering equality.

References [coeff](#), [GiNaC::ex::is\\_equal\(\)](#), and [rest](#).

Referenced by [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::mul::expair\\_needs\\_further\\_processing\(\)](#), and [GiNaC::expairseq::subchildren\(\)](#).

### 6.53.3.2 is\_less()

```
bool GiNaC::expair::is_less (
 const expair & other) const [inline]
```

Member-wise check for canonical ordering lessness.

References [coeff](#), [GiNaC::ex::compare\(\)](#), and [rest](#).

Referenced by [GiNaC::expair\\_is\\_less::operator\(\)\(\)](#).

### 6.53.3.3 compare()

```
int GiNaC::expair::compare (
 const expair & other) const [inline]
```

Member-wise check for canonical ordering.

References [coeff](#), [GiNaC::ex::compare\(\)](#), and [rest](#).

### 6.53.3.4 print()

```
void GiNaC::expair::print (
 std::ostream & os) const
```

References [c](#), [coeff](#), [GiNaC::ex::print\(\)](#), and [rest](#).

### 6.53.3.5 is\_canonical\_numeric()

```
bool GiNaC::expair::is_canonical_numeric () const [inline]
```

True if this is of the form (numeric,ex(1)).

References [coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), and [rest](#).

### 6.53.3.6 swap()

```
void GiNaC::expair::swap (
 expair & other) [inline]
```

Swap contents with other expair.

References [coeff](#), [rest](#), and [GiNaC::ex::swap\(\)](#).

Referenced by [GiNaC::mul::derivative\(\)](#), [GiNaC::expair\\_swap::operator\(\)\(\)](#), and [GiNaC::swap\(\)](#).

### 6.53.3.7 conjugate()

```
const expair GiNaC::expair::conjugate () const
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [coeff](#), [GiNaC::ex::conjugate\(\)](#), [expair\(\)](#), and [rest](#).

## 6.53.4 Member Data Documentation

### 6.53.4.1 rest

`ex` `GiNaC::expair::rest`

first member of pair, an arbitrary expression

Referenced by `GiNaC::expairseq::combine_pair_with_coeff_to_pair()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `compare()`, `conjugate()`, `GiNaC::expairseq::construct_from_2_ex()`, `GiNaC::expairseq::construct_from_expairseq_ex()`, `is_canonical_numeric()`, `is_equal()`, `is_less()`, `GiNaC::expair_rest_is_less::operator()`, `print()`, `GiNaC::expairseq::printpair()`, `GiNaC::expairseq::recombine_pair_to_ex()`, `GiNaC::add::recombine_pair_to_ex()`, `GiNaC::mul::recombine_pair_to_ex()`, and `swap()`.

### 6.53.4.2 coeff

`ex` `GiNaC::expair::coeff`

second member of pair, must be numeric

Referenced by `GiNaC::mul::can_make_flat()`, `GiNaC::expairseq::combine_pair_with_coeff_to_pair()`, `GiNaC::add::combine_pair_with_coeff_to_pair()`, `GiNaC::mul::combine_pair_with_coeff_to_pair()`, `compare()`, `conjugate()`, `GiNaC::expairseq::construct_from_2_ex()`, `GiNaC::expairseq::construct_from_expairseq_ex()`, `expair()`, `GiNaC::power::expand_mul()`, `is_canonical_numeric()`, `is_equal()`, `is_less()`, `print()`, `GiNaC::expairseq::printpair()`, `GiNaC::expairseq::recombine_pair_to_ex()`, `GiNaC::add::recombine_pair_to_ex()`, `GiNaC::mul::recombine_pair_to_ex()`, and `swap()`.

The documentation for this class was generated from the following files:

- `expair.h`
- `expair.cpp`

## 6.54 GiNaC::expair\_is\_less Struct Reference

Function object for insertion into third argument of STL's `sort()` etc.

```
#include <expair.h>
```

### Public Member Functions

- `bool operator() (const expair &lh, const expair &rh) const`

### 6.54.1 Detailed Description

Function object for insertion into third argument of STL's `sort()` etc.

## 6.54.2 Member Function Documentation

### 6.54.2.1 operator()

```
bool GiNaC::expair_is_less::operator() (
 const expair & lh,
 const expair & rh) const [inline]
```

References [GiNaC::expair::is\\_less\(\)](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.55 GiNaC::expair\_rest\_is\_less Struct Reference

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

```
#include <expair.h>
```

### Public Member Functions

- `bool operator() (const expair &lh, const expair &rh) const`

### 6.55.1 Detailed Description

Function object not caring about the numerical coefficients for insertion into third argument of STL's sort().

Note that this does not define a strict weak ordering since for any symbol  $x$  we have neither  $3*x < 2*x$  or  $2*x < 3*x$ . Handle with care!

## 6.55.2 Member Function Documentation

### 6.55.2.1 operator()

```
bool GiNaC::expair_rest_is_less::operator() (
 const expair & lh,
 const expair & rh) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::expair::rest](#).

The documentation for this struct was generated from the following file:

- [expair.h](#)

## 6.56 GiNaC::expair\_swap Struct Reference

```
#include <expair.h>
```

### Public Member Functions

- void [operator\(\)](#) ([expair](#) &lh, [expair](#) &rh) const

### 6.56.1 Member Function Documentation

#### 6.56.1.1 operator()

```
void GiNaC::expair_swap::operator() (
 expair & lh,
 expair & rh) const [inline]
```

References [GiNaC::expair::swap\(\)](#).

The documentation for this struct was generated from the following file:

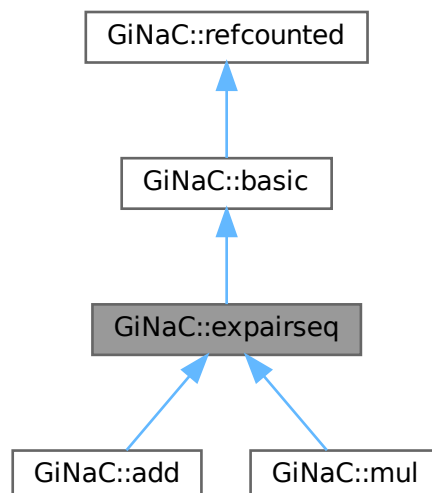
- [expair.h](#)

## 6.57 GiNaC::expairseq Class Reference

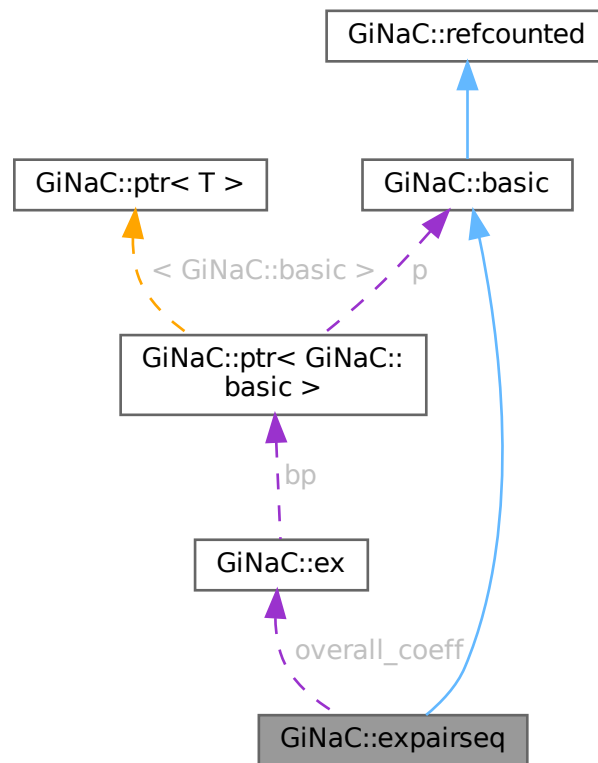
A sequence of class [expair](#).

```
#include <expairseq.h>
```

Inheritance diagram for [GiNaC::expairseq](#):



Collaboration diagram for GiNaC::expairseq:



## Public Member Functions

- `expairseq` (const `ex` &lh, const `ex` &rh)
- `expairseq` (const `exvector` &v)
- `expairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `expairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- size\_t `nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex eval` () const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for expairseqs.*

- `ex to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for `expairseqs`.*
- bool `match` (const `ex` &pattern, `exmap` &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex conjugate` () const override
- void `archive` (`archive_node` &n) const override  
*Save (serialize) the object into archive node.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned `level`=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*

- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree `n` in object `s`.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `return_type` () const override
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- virtual `ex thisexpairseq` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false) const  
*Create an object of this type.*
- virtual `ex thisexpairseq` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false) const
- virtual void `printseq` (const `print_context` &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void `printpair` (const `print_context` &c, const `expair` &p, unsigned upper\_precedence) const
- virtual `expair split_ex_to_pair` (const `ex` &e) const  
*Form an expair from an ex, using the corresponding semantics.*
- virtual `expair combine_ex_with_coeff_to_pair` (const `ex` &e, const `ex` &c) const
- virtual `expair combine_pair_with_coeff_to_pair` (const `expair` &p, const `ex` &c) const
- virtual `ex recombine_pair_to_ex` (const `expair` &p) const  
*Form an ex out of an expair, using the corresponding semantics.*
- virtual bool `expair_needs_further_processing` (`epp` it)
- virtual `ex default_overall_coeff` () const
- virtual void `combine_overall_coeff` (const `ex` &c)
- virtual void `combine_overall_coeff` (const `ex` &c1, const `ex` &c2)
- virtual bool `can_make_flat` (const `expair` &p) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `construct_from_2_ex` (const `ex` &lh, const `ex` &rh)
- void `construct_from_2_expairseq` (const `expairseq` &s1, const `expairseq` &s2)
- void `construct_from_expairseq_ex` (const `expairseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
*Combine this expairseq with argument exvector.*
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
*Combine this expairseq with argument epvector.*
- void `canonicalize` ()  
*Brings this expairseq into a sorted (canonical) form.*
- void `combine_same_terms_sorted_seq` ()  
*Compact a presorted expairseq by combining all matching expairs to one each.*
- bool `is_canonical` () const  
*Check if this expairseq is in sorted (canonical) form.*
- `epvector expandchildren` (unsigned `options`) const  
*Member-wise expand the expairs in this sequence.*
- `epvector evalchildren` () const  
*Member-wise evaluate the expairs in this sequence.*
- `epvector subschildren` (const `exmap` &m, unsigned `options`=0) const  
*Member-wise substitute in this sequence.*

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [epvector](#) seq
- [ex](#) overall\_coeff

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.57.1 Detailed Description

A sequence of class `expair`.

This is used for time-critical classes like sums and products of terms since handling a list of coeff and rest is much faster than handling a list of products or powers, respectively. (Not incidentally, Maple does it the same way, maybe others too.) The semantics is (at least) twofold: one for addition and one for multiplication and several methods have to be overridden by derived classes to reflect the change in semantics. However, most functionality turns out to be shared between addition and multiplication, which is the reason why there is this base class.

### 6.57.2 Constructor & Destructor Documentation

#### 6.57.2.1 `expairseq()` [1/4]

```
GiNaC::expairseq::expairseq (
 const ex & lh,
 const ex & rh)
```

References [construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

Referenced by [thisexpairseq\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.2.2 `expairseq()` [2/4]

```
GiNaC::expairseq::expairseq (
 const exvector & v)
```

References [construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 6.57.2.3 `expairseq()` [3/4]

```
GiNaC::expairseq::expairseq (
 const epvector & v,
 const ex & oc,
 bool do_index_renaming = false)
```

References [construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

### 6.57.2.4 `expairseq()` [4/4]

```
GiNaC::expairseq::expairseq (
 epvector && vp,
 const ex & oc,
 bool do_index_renaming = false)
```

References [construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), and [is\\_canonical\(\)](#).

## 6.57.3 Member Function Documentation

### 6.57.3.1 `precedence()`

```
unsigned GiNaC::expairseq::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

Referenced by [do\\_print\(\)](#), and [printpair\(\)](#).

### 6.57.3.2 `info()`

```
bool GiNaC::expairseq::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).

### 6.57.3.3 nops()

```
size_t GiNaC::expairseq::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [GiNaC::mul::has\(\)](#), and [match\(\)](#).

### 6.57.3.4 op()

```
ex GiNaC::expairseq::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [seq](#).

Referenced by [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::add::conjugate\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::add::get\\_free\\_indices\(\)](#), [GiNaC::mul::get\\_free\\_indices\(\)](#), [match\(\)](#), [GiNaC::add::series\(\)](#), and [GiNaC::mul::series\(\)](#).

### 6.57.3.5 map()

```
ex GiNaC::expairseq::map (
 map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.6 eval()

```
ex GiNaC::expairseq::eval () const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::basic::hold\(\)](#), and [overall\\_coeff](#).

### 6.57.3.7 to\_rational()

```
ex GiNaC::expairseq::to_rational (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), [GiNaC::ex::to\\_rational\(\)](#), and [to\\_rational\(\)](#).

Referenced by [to\\_rational\(\)](#).

### 6.57.3.8 to\_polynomial()

```
ex GiNaC::expairseq::to_polynomial (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for expairseqs.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), [GiNaC::ex::to\\_polynomial\(\)](#), and [to\\_polynomial\(\)](#).

Referenced by [to\\_polynomial\(\)](#).

### 6.57.3.9 match()

```
bool GiNaC::expairseq::match (
 const ex & pattern,
 exmap & repl_lst) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to `repl_lst`.

Reimplemented from [GiNaC::basic](#).

References [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), [nops\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [split\\_ex\\_to\\_pair\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.10 subs()

```
ex GiNaC::expairseq::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The `ex` returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [m](#), [GiNaC::subs\\_options::no\\_index\\_renaming](#), [options](#), [overall\\_coeff](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [subschildren\(\)](#), and [thisexpairseq\(\)](#).

### 6.57.3.11 conjugate()

```
ex GiNaC::expairseq::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [overall\\_coeff](#), [seq](#), [thisexpairseq\(\)](#), and [x](#).

### 6.57.3.12 archive()

```
void GiNaC::expairseq::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

References [n](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.13 read\_archive()

```
void GiNaC::expairseq::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

References [canonicalize\(\)](#), [GiNaC::basic::coeff\(\)](#), [GINAC\\_ASSERT](#), [is\\_canonical\(\)](#), [n](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.14 is\_equal\_same\_type()

```
bool GiNaC::expairseq::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), and [seq](#).

### 6.57.3.15 return\_type()

```
unsigned GiNaC::expairseq::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 6.57.3.16 calchash()

```
unsigned GiNaC::expairseq::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [overall\\_coeff](#), [GiNaC::rotate\\_left\(\)](#), [seq](#), and [GiNaC::basic::setflag\(\)](#).

### 6.57.3.17 expand()

```
ex GiNaC::expairseq::expand (
 unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::mul](#).

References [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [options](#), [overall\\_coeff](#), [GiNaC::basic::setflag\(\)](#), and [thisexpairseq\(\)](#).

**6.57.3.18 thisexpairseq() [1/2]**

```
ex GiNaC::expairseq::thisexpairseq (
 const epvector & v,
 const ex & oc,
 bool do_index_renaming = false) const [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [expairseq\(\)](#).

Referenced by [conjugate\(\)](#), [expand\(\)](#), [map\(\)](#), [match\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.19 thisexpairseq() [2/2]**

```
ex GiNaC::expairseq::thisexpairseq (
 epvector && vp,
 const ex & oc,
 bool do_index_renaming = false) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [expairseq\(\)](#).

**6.57.3.20 printseq()**

```
void GiNaC::expairseq::printseq (
 const print_context & c,
 char delim,
 unsigned this_precedence,
 unsigned upper_precedence) const [protected], [virtual]
```

References [c](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [overall\\_coeff](#), [GiNaC::ex::print\(\)](#), [printpair\(\)](#), and [seq](#).

Referenced by [do\\_print\(\)](#).

**6.57.3.21 printpair()**

```
void GiNaC::expairseq::printpair (
 const print_context & c,
 const expair & p,
 unsigned upper_precedence) const [protected], [virtual]
```

References [c](#), [GiNaC::expair::coeff](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [is\\_canonical\(\)](#), and [printseq\(\)](#).

### 6.57.3.22 split\_ex\_to\_pair()

```
expair GiNaC::expairseq::split_ex_to_pair (
 const ex & e) const [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::\\_ex1](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [match\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.57.3.23 combine\_ex\_with\_coeff\_to\_pair()

```
expair GiNaC::expairseq::combine_ex_with_coeff_to_pair (
 const ex & e,
 const ex & c) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [c](#), and [GINAC\\_ASSERT](#).

Referenced by [evalchildren\(\)](#), and [subschildren\(\)](#).

### 6.57.3.24 combine\_pair\_with\_coeff\_to\_pair()

```
expair GiNaC::expairseq::combine_pair_with_coeff_to_pair (
 const expair & p,
 const ex & c) const [protected], [virtual]
```

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), and [GiNaC::expair::rest](#).

### 6.57.3.25 recombine\_pair\_to\_ex()

```
ex GiNaC::expairseq::recombine_pair_to_ex (
 const expair & p) const [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented in [GiNaC::add](#), and [GiNaC::mul](#).

References [GiNaC::expair::coeff](#), and [GiNaC::expair::rest](#).

Referenced by [map\(\)](#), [op\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.26 expair\_needs\_further\_processing()**

```
bool GiNaC::expairseq::expair_needs_further_processing (
 expair it) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex1](#).

Referenced by [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), and [construct\\_from\\_expairseq\\_ex\(\)](#).

**6.57.3.27 default\_overall\_coeff()**

```
ex GiNaC::expairseq::default_overall_coeff () const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GiNaC::\\_ex0](#).

Referenced by [do\\_print\\_tree\(\)](#), [map\(\)](#), [match\(\)](#), [nops\(\)](#), [op\(\)](#), [printseq\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.57.3.28 combine\_overall\_coeff()** [1/2]

```
void GiNaC::expairseq::combine_overall_coeff (
 const ex & c) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [c](#), [GINAC\\_ASSERT](#), and [overall\\_coeff](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [make\\_flat\(\)](#), and [make\\_flat\(\)](#).

**6.57.3.29 combine\_overall\_coeff()** [2/2]

```
void GiNaC::expairseq::combine_overall_coeff (
 const ex & c1,
 const ex & c2) [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

References [GINAC\\_ASSERT](#), and [overall\\_coeff](#).

**6.57.3.30 can\_make\_flat()**

```
bool GiNaC::expairseq::can_make_flat (
 const expair & p) const [protected], [virtual]
```

Reimplemented in [GiNaC::mul](#).

Referenced by [make\\_flat\(\)](#).

**6.57.3.31 do\_print()**

```
void GiNaC::expairseq::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [precedence\(\)](#), and [printseq\(\)](#).

**6.57.3.32 do\_print\_tree()**

```
void GiNaC::expairseq::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [default\\_overall\\_coeff\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::is\\_equal\(\)](#), [nops\(\)](#), [overall\\_coeff](#), [GiNaC::ex::print\(\)](#), and [seq](#).

**6.57.3.33 construct\_from\_2\_ex()**

```
void GiNaC::expairseq::construct_from_2_ex (
 const ex & lh,
 const ex & rh) [protected]
```

References [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::ex::compare\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.34 construct\_from\_2\_expairseq()**

```
void GiNaC::expairseq::construct_from_2_expairseq (
 const expairseq & s1,
 const expairseq & s2) [protected]
```

References [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [overall\\_coeff](#), and [seq](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).

**6.57.3.35 construct\_from\_expairseq\_ex()**

```
void GiNaC::expairseq::construct_from_expairseq_ex (
 const expairseq & s,
 const ex & e) [protected]
```

References [GiNaC::expair::coeff](#), [combine\\_overall\\_coeff\(\)](#), [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [last](#), [overall\\_coeff](#), [GiNaC::expair::rest](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_2\\_ex\(\)](#).

**6.57.3.36 construct\_from\_exvector()**

```
void GiNaC::expairseq::construct_from_exvector (
 const exvector & v) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.37 construct\_from\_epvector() [1/2]**

```
void GiNaC::expairseq::construct_from_epvector (
 const epvector & v,
 bool do_index_renaming = false) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

Referenced by [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), and [GiNaC::mul::mul\(\)](#).

**6.57.3.38 construct\_from\_epvector() [2/2]**

```
void GiNaC::expairseq::construct_from_epvector (
 epvector && v,
 bool do_index_renaming = false) [protected]
```

References [canonicalize\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), and [make\\_flat\(\)](#).

**6.57.3.39 make\_flat() [1/2]**

```
void GiNaC::expairseq::make_flat (
 const exvector & v) [protected]
```

Combine this expairseq with argument exvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::\\_ex1](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [overall\\_coeff](#), [seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**6.57.3.40 make\_flat() [2/2]**

```
void GiNaC::expairseq::make_flat (
 const epvector & v,
 bool do_index_renaming = false) [protected]
```

Combine this expairseq with argument epvector.

It cares for associativity as well as for special handling of numerics.

References [GiNaC::\\_ex1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [can\\_make\\_flat\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_overall\\_coeff\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [overall\\_coeff](#), and [seq](#).

**6.57.3.41 canonicalize()**

```
void GiNaC::expairseq::canonicalize () [protected]
```

Brings this expairseq into a sorted (canonical) form.

References [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_exvector\(\)](#), and [read\\_archive\(\)](#).

**6.57.3.42 combine\_same\_terms\_sorted\_seq()**

```
void GiNaC::expairseq::combine_same_terms_sorted_seq () [protected]
```

Compact a presorted expairseq by combining all matching expairs to one each.

On an add object, this is responsible for  $2*x+3*x+y \rightarrow 5*x+y$ , for instance.

References [construct\\_from\\_epvector\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [last](#), and [seq](#).

Referenced by [construct\\_from\\_epvector\(\)](#), [construct\\_from\\_epvector\(\)](#), and [construct\\_from\\_exvector\(\)](#).

**6.57.3.43 is\_canonical()**

```
bool GiNaC::expairseq::is_canonical () const [protected]
```

Check if this expairseq is in sorted (canonical) form.

Useful mainly for debugging or in assertions since being sorted is an invariance.

References [printpair\(\)](#), and [seq](#).

Referenced by [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [expairseq\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), and [read\\_archive\(\)](#).

**6.57.3.44 expandchildren()**

```
epvector GiNaC::expairseq::expandchildren (
 unsigned options) const [protected]
```

Member-wise expand the expairs in this sequence.

See also

[expairseq::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [last](#), [options](#), and [seq](#).

Referenced by [GiNaC::add::expand\(\)](#), and [expand\(\)](#).

### 6.57.3.45 evalchildren()

```
epvector GiNaC::expairseq::evalchildren () const [protected]
```

Member-wise evaluate the expairs in this sequence.

See also

[expairseq::eval\(\)](#)

Returns

epvector containing evaluated pairs, empty if no members had to be changed.

References [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [last](#), [seq](#), and [unlikely](#).

Referenced by [GiNaC::add::eval\(\)](#), [eval\(\)](#), and [GiNaC::mul::eval\(\)](#).

### 6.57.3.46 subschildren()

```
epvector GiNaC::expairseq::subschildren (
 const exmap & m,
 unsigned options = 0) const [protected]
```

Member-wise substitute in this sequence.

See also

[expairseq::subs\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expair::is\\_equal\(\)](#), [last](#), [m](#), [options](#), [GiNaC::subs\\_options::pattern\\_is\\_not\\_product](#), [GiNaC::subs\\_options::pattern\\_is\\_product](#), [recombine\\_pair\\_to\\_ex\(\)](#), [seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [subs\(\)](#).

## 6.57.4 Member Data Documentation

### 6.57.4.1 seq

```
epvector GiNaC::expairseq::seq [protected]
```

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonicalize\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [construct\\_from\\_2\\_ex\(\)](#), [construct\\_from\\_2\\_expairseq\(\)](#), [construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::add::degree\(\)](#), [GiNaC::mul::degree\(\)](#), [GiNaC::add::derivative\(\)](#), [GiNaC::mul::derivative\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::add::eval\\_ncmul\(\)](#), [GiNaC::mul::eval\\_ncmul\(\)](#), [evalchildren\(\)](#), [GiNaC::mul::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::mul::find\\_real\\_imag\(\)](#), [GiNaC::add::imag\\_part\(\)](#), [GiNaC::add::info\(\)](#), [info\(\)](#), [GiNaC::mul::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [is\\_canonical\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::add::is\\_polynomial\(\)](#), [GiNaC::mul::is\\_polynomial\(\)](#), [GiNaC::add::ldegree\(\)](#), [GiNaC::mul::ldegree\(\)](#), [make\\_flat\(\)](#), [make\\_flat\(\)](#), [map\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [nops\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [op\(\)](#), [GiNaC::add::print\\_add\(\)](#), [printseq\(\)](#), [read\\_archive\(\)](#), [GiNaC::add::real\\_part\(\)](#), [GiNaC::add::return\\_type\(\)](#), [GiNaC::mul::return\\_type\(\)](#), [GiNaC::add::return\\_type\\_tinfo\(\)](#), [GiNaC::mul::return\\_type\\_tinfo\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [subschildren\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.57.4.2 overall\_coeff

ex `GiNaC::expairseq::overall_coeff` [protected]

Referenced by `GiNaC::add::add()`, `GiNaC::add::add()`, `GiNaC::add::add()`, `GiNaC::add::add()`, `GiNaC::add::add()`, `GiNaC::add::add()`, `archive()`, `calchash()`, `GiNaC::add::coeff()`, `GiNaC::mul::coeff()`, `GiNaC::add::combine_ex_with_coeff_to_pair()`, `combine_overall_coeff()`, `GiNaC::mul::combine_overall_coeff()`, `combine_overall_coeff()`, `GiNaC::mul::combine_overall_coeff()`, `conjugate()`, `GiNaC::mul::conjugate()`, `construct_from_2_expairseq()`, `construct_from_expairseq_ex()`, `GiNaC::add::degree()`, `GiNaC::mul::derivative()`, `GiNaC::add::do_print_csrc()`, `GiNaC::mul::do_print_csrc()`, `do_print_tree()`, `GiNaC::add::eval()`, `eval()`, `GiNaC::mul::eval()`, `GiNaC::power::eval()`, `GiNaC::mul::evalf()`, `GiNaC::add::evalm()`, `GiNaC::mul::evalm()`, `GiNaC::add::expand()`, `expand()`, `GiNaC::mul::expand()`, `GiNaC::power::expand()`, `GiNaC::power::expand_add()`, `GiNaC::power::expand_add_2()`, `GiNaC::mul::find_real_imag()`, `GiNaC::add::imag_part()`, `GiNaC::add::info()`, `GiNaC::mul::info()`, `GiNaC::add::integer_content()`, `GiNaC::mul::integer_content()`, `is_equal_same_type()`, `GiNaC::add::lddegree()`, `make_flat()`, `make_flat()`, `map()`, `GiNaC::add::max_coefficient()`, `GiNaC::mul::max_coefficient()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `GiNaC::mul::mul()`, `nops()`, `GiNaC::add::normal()`, `GiNaC::mul::normal()`, `op()`, `GiNaC::add::print_add()`, `GiNaC::mul::print_overall_coeff()`, `printseq()`, `read_archive()`, `GiNaC::add::real_part()`, `GiNaC::add::series()`, `GiNaC::mul::series()`, `GiNaC::add::smod()`, `GiNaC::mul::smod()`, `GiNaC::add::split_ex_to_pair()`, `subs()`, `to_polynomial()`, and `to_rational()`.

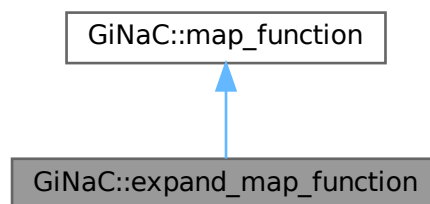
The documentation for this class was generated from the following files:

- `expairseq.h`
- `expairseq.cpp`
- `normal.cpp`

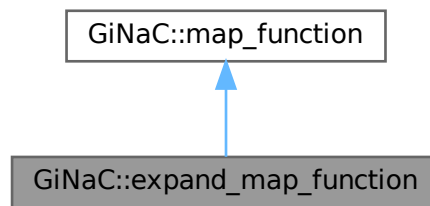
## 6.58 GiNaC::expand\_map\_function Struct Reference

Function object to be applied by `basic::expand()`.

Inheritance diagram for `GiNaC::expand_map_function`:



Collaboration diagram for GiNaC::expand\_map\_function:



### Public Member Functions

- [expand\\_map\\_function](#) (unsigned o)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Public Attributes

- unsigned [options](#)

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.58.1 Detailed Description

Function object to be applied by [basic::expand\(\)](#).

## 6.58.2 Constructor & Destructor Documentation

### 6.58.2.1 [expand\\_map\\_function\(\)](#)

```
GiNaC::expand_map_function::expand_map_function (
 unsigned o) [inline]
```

### 6.58.3 Member Function Documentation

#### 6.58.3.1 operator()

```
ex GiNaC::expand_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::ex::expand\(\)](#), and [options](#).

### 6.58.4 Member Data Documentation

#### 6.58.4.1 options

```
unsigned GiNaC::expand_map_function::options
```

The documentation for this struct was generated from the following file:

- [basic.cpp](#)

## 6.59 GiNaC::expand\_options Class Reference

Flags to control the behavior of [expand\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum { [expand\\_indexed](#) = 0x0001 , [expand\\_function\\_args](#) = 0x0002 , [expand\\_rename\\_idx](#) = 0x0004 , [expand\\_transcendental](#) = 0x0008 }

#### 6.59.1 Detailed Description

Flags to control the behavior of [expand\(\)](#).

### 6.59.2 Member Enumeration Documentation

#### 6.59.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|                                       |                                                   |
|---------------------------------------|---------------------------------------------------|
| <a href="#">expand_indexed</a>        | expands (a+b).i to a.i+b.i                        |
| <a href="#">expand_function_args</a>  | expands the arguments of functions                |
| <a href="#">expand_rename_idx</a>     | used internally by <a href="#">mul::expand()</a>  |
| <a href="#">expand_transcendental</a> | expands transcendental functions like log and exp |

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.60 GiNaC::factor\_options Class Reference

Flags to control the polynomial factorization.

```
#include <flags.h>
```

### Public Types

- enum { [polynomial](#) = 0x0000 , [all](#) = 0x0001 }

### 6.60.1 Detailed Description

Flags to control the polynomial factorization.

### 6.60.2 Member Enumeration Documentation

#### 6.60.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|                            |                                              |
|----------------------------|----------------------------------------------|
| <a href="#">polynomial</a> | factor only expressions that are polynomials |
| <a href="#">all</a>        | factor all polynomial subexpressions         |

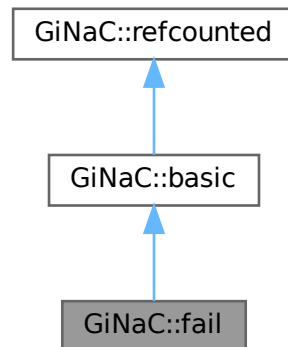
The documentation for this class was generated from the following file:

- [flags.h](#)

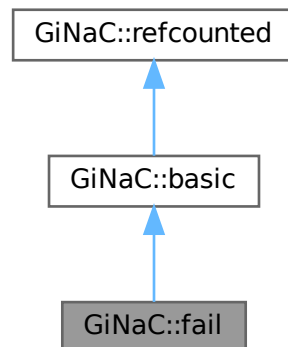
## 6.61 GiNaC::fail Class Reference

```
#include <fail.h>
```

Inheritance diagram for GiNaC::fail:



Collaboration diagram for GiNaC::fail:



### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const

*Returns true if the attributes of two objects are similar enough for a match.*

- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*

- virtual `ex op` (`size_t i`) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (`const ex &index`) const
- virtual `ex operator[]` (`size_t i`) const
- virtual `ex & let_op` (`size_t i`)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (`const ex &index`)
- virtual `ex & operator[]` (`size_t i`)
- virtual `bool has` (`const ex &other`, unsigned `options=0`) const  
*Test for occurrence of a pattern.*
- virtual `bool match` (`const ex &pattern`, `exmap &repls`) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (`const exmap &m`, unsigned `options=0`) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function &f`) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept` (`GiNaC::visitor &v`) const
- virtual `bool is_polynomial` (`const ex &var`) const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree` (`const ex &s`) const  
*Return degree of highest power in object s.*
- virtual `int ldegree` (`const ex &s`) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (`const ex &s`, int `n=1`) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options=0`) const  
*Expand expression, i.e.*
- virtual `ex collect` (`const ex &s`, bool `distributed=false`) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (`const relational &r`, int `order`, unsigned `options=0`) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap &repl`, `exmap &rev_lookup`, `lst &modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap &repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap &repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (`const numeric &xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (`const ex &self`, `const ex &other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (`const ex &self`, `const numeric &other`) const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with` (`exvector::iterator self`, `exvector::iterator other`, `exvector &v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const

- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.61.1 Member Function Documentation

#### 6.61.1.1 [return\\_type\(\)](#)

unsigned [GiNaC::fail::return\\_type](#) ( ) const [inline], [override], [protected], [virtual]

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

### 6.61.1.2 do\_print()

```
void GiNaC::fail::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following file:

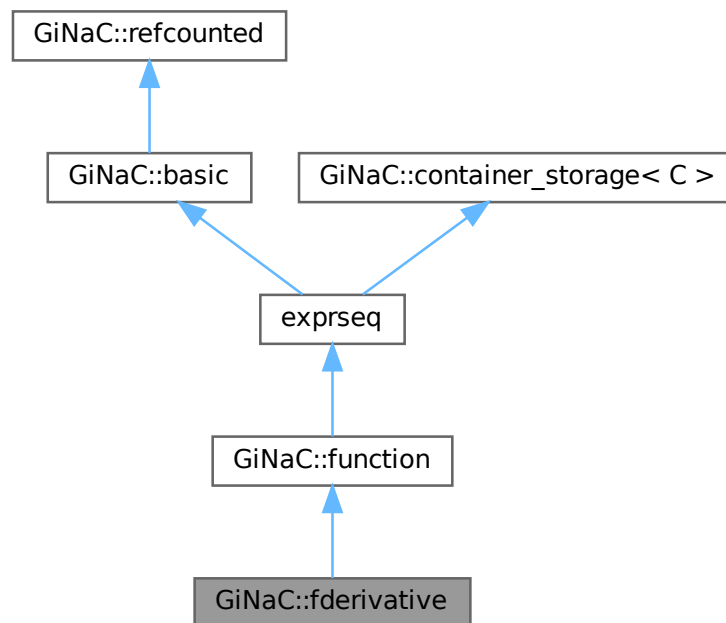
- [fail.h](#)

## 6.62 GiNaC::fderivative Class Reference

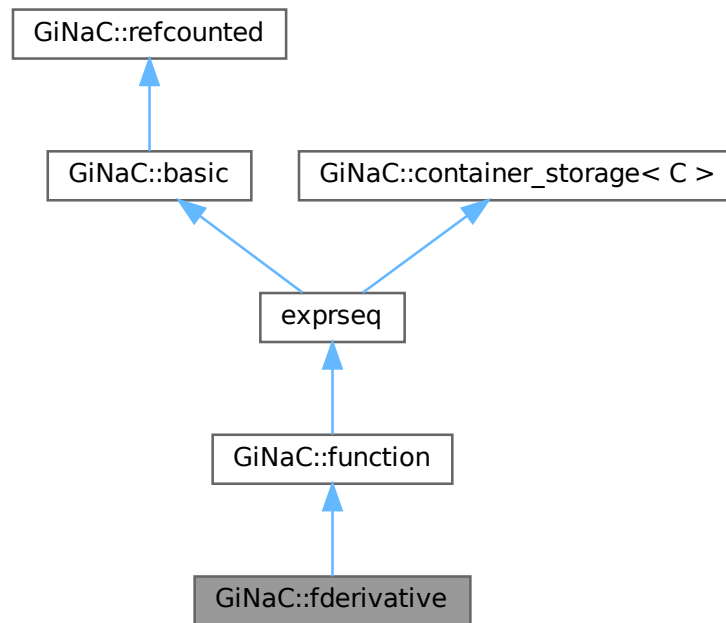
This class represents the (abstract) derivative of a symbolic function.

```
#include <fderivative.h>
```

Inheritance diagram for GiNaC::fderivative:



Collaboration diagram for GiNaC::fderivative:



## Public Member Functions

- **fderivative** (unsigned ser, unsigned param, const **exvector** &args)  
*Construct derivative with respect to one parameter.*
- **fderivative** (unsigned ser, const **paramset** &params, const **exvector** &args)  
*Construct derivative with respect to multiple parameters.*
- **fderivative** (unsigned ser, const **paramset** &params, **exvector** &&v)
- void **print** (const **print\_context** &c, unsigned level=0) const override  
*Output to stream.*
- **ex eval** () const override  
*Perform automatic non-interruptive term rewriting rules.*
- **ex series** (const **relational** &r, int **order**, unsigned **options**=0) const override  
*The series expansion of derivatives falls back to Taylor expansion.*
- **ex thiscontainer** (const **exvector** &v) const override
- **ex thiscontainer** (**exvector** &&v) const override
- void **archive** (**archive\_node** &n) const override  
*Archive the object.*
- void **read\_archive** (const **archive\_node** &n, **lst** &syms) override  
*Load (deserialize) the object from an archive node.*
- const **paramset** & **derivatives** () const  
*Expose this object's derivative structure.*

## Public Member Functions inherited from `GiNaC::function`

- `function` (unsigned ser)
- `function` (unsigned ser, const `ex` &param1)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12, const `ex` &param13)
- `function` (unsigned ser, const `ex` &param1, const `ex` &param2, const `ex` &param3, const `ex` &param4, const `ex` &param5, const `ex` &param6, const `ex` &param7, const `ex` &param8, const `ex` &param9, const `ex` &param10, const `ex` &param11, const `ex` &param12, const `ex` &param13, const `ex` &param14)
- `function` (unsigned ser, const `exprseq` &es)
- `function` (unsigned ser, const `exvector` &v)
- `function` (unsigned ser, `exvector` &&v)
- void `print` (const `print_context` &c, unsigned level=0) const override  
*Output to stream.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthesizing output).*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `ex eval` () const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex evalf` () const override  
*Evaluate object numerically.*
- `ex eval_ncmul` (const `exvector` &v) const override  
*This method is defined to be in line with behavior of `function::return_type()`*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*Implementation of `ex::series` for functions.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- `ex conjugate` () const override  
*Implementation of `ex::conjugate` for functions.*

- `ex real_part ()` const override  
*Implementation of `ex::real_part` for functions.*
- `ex imag_part ()` const override  
*Implementation of `ex::imag_part` for functions.*
- `void archive (archive_node &n)` const override  
*Archive the object.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Construct object from `archive_node`.*
- `bool info (unsigned inf)` const override  
*Implementation of `ex::info` for functions.*
- `ex power (const ex &exp)` const
- `unsigned get_serial ()` const
- `std::string get_name ()` const  
*Return the print name of the function.*

## Public Member Functions inherited from `GiNaC::container< C >`

- `container (STLT const &s)`
- `container (STLT &&v)`
- `container (exvector::const_iterator b, exvector::const_iterator e)`
- `container (std::initializer_list< ex > il)`
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position *i*.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position *i*.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend (const ex &b)`  
*Add element at front.*
- `container & append (const ex &b)`  
*Add element at back.*
- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`  
*Sort elements.*
- `container & unique ()`  
*Remove adjacent duplicate elements.*
- `const_iterator begin ()` const
- `const_iterator end ()` const
- `const_reverse_iterator rbegin ()` const
- `const_reverse_iterator rend ()` const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void dbgprint () const`  
*Little wrapper around `print` to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object `s`.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object `s`.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree `n` in object `s`.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`

- *Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- *Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- *Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- *Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- *Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- *Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- *Test for syntactic equality.*
- const `basic` & `hold` () const
- *Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- *Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override
- *Implementation of `ex::diff()` for derivatives.*
- bool `is_equal_same_type` (const `basic` &other) const override
- *Returns true if two objects of same type are equal.*
- bool `match_same_type` (const `basic` &other) const override
- *Returns true if the attributes of two objects are similar enough for a match.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_context` &c, unsigned level) const
- void `do_print_csrx` (const `print_csrx` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::function](#)

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for functions.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex pderivative](#) (unsigned diff\_param) const
- [ex expl\\_derivative](#) (const [symbol](#) &s) const
- bool [lookup\\_remember\\_table](#) ([ex](#) &result) const
- void [store\\_remember\\_table](#) ([ex](#) const &result) const

## Protected Member Functions inherited from [GiNaC::container< C >](#)

- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subschildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

**Protected Attributes**

- [paramset](#) [parameter\\_set](#)

*Set of parameter numbers with respect to which to take the derivative.*

**Protected Attributes inherited from [GiNaC::function](#)**

- unsigned [serial](#)

**Protected Attributes inherited from [GiNaC::basic](#)**

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

**Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)**

- [STLT](#) [seq](#)

**Additional Inherited Members****Public Types inherited from [GiNaC::container< C >](#)**

- typedef [STLT::const\\_iterator](#) [const\\_iterator](#)
- typedef [STLT::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

**Static Public Member Functions inherited from [GiNaC::function](#)**

- static unsigned [register\\_new](#) ([function\\_options](#) const &opt)
- static unsigned [find\\_function](#) (const std::string &name, unsigned nparams)  
*Find serial number of function by name and number of parameters.*
- static std::vector< [function\\_options](#) > [get\\_registered\\_functions](#) ()

**Static Public Attributes inherited from [GiNaC::function](#)**

- static unsigned [current\\_serial](#) = 0  
*This can be used as a hook for external applications.*

**Protected Types inherited from [GiNaC::container< C >](#)**

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

**Protected Types inherited from [GiNaC::container\\_storage< C >](#)**

- typedef C< [ex](#) > [STLT](#)

## Static Protected Member Functions inherited from [GiNaC::function](#)

- static `std::vector< function\_options > & registered_functions ()`

## Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for `lst`.*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for `lst`.*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for `lst`.*

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, `size_t`)

### 6.62.1 Detailed Description

This class represents the (abstract) derivative of a symbolic function.

It is used to represent the derivatives of functions that do not have a derivative or series expansion procedure defined.

### 6.62.2 Constructor & Destructor Documentation

#### 6.62.2.1 `fderivative()` [1/3]

```
GiNaC::fderivative::fderivative (
 unsigned ser,
 unsigned param,
 const exvector & args)
```

Construct derivative with respect to one parameter.

#### Parameters

|              |                                                                  |
|--------------|------------------------------------------------------------------|
| <i>ser</i>   | Serial number of function                                        |
| <i>param</i> | Number of parameter with respect to which to take the derivative |
| <i>args</i>  | Arguments of derivative function                                 |

References [parameter\\_set](#).

Referenced by [derivative\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 6.62.2.2 `fderivative()` [2/3]

```
GiNaC::fderivative::fderivative (
```

```

 unsigned ser,
 const paramset & params,
 const exvector & args)

```

Construct derivative with respect to multiple parameters.

#### Parameters

|               |                                                                           |
|---------------|---------------------------------------------------------------------------|
| <i>ser</i>    | Serial number of function                                                 |
| <i>params</i> | Set of numbers of parameters with respect to which to take the derivative |
| <i>args</i>   | Arguments of derivative function                                          |

### 6.62.2.3 fderivative() [3/3]

```

GiNaC::fderivative::fderivative (
 unsigned ser,
 const paramset & params,
 exvector && v)

```

## 6.62.3 Member Function Documentation

### 6.62.3.1 print()

```

void GiNaC::fderivative::print (
 const print_context & c,
 unsigned level = 0) const [override], [virtual]

```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

#### Parameters

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| <i>c</i>     | print context object that describes the output formatting                                                 |
| <i>level</i> | value that is used to identify the precedence or indentation level for placing parentheses and formatting |

Reimplemented from [GiNaC::basic](#).

References [c](#), and [GiNaC::basic::print\(\)](#).

### 6.62.3.2 eval()

```

ex GiNaC::fderivative::eval () const [override], [virtual]

```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::function::function\(\)](#), [GiNaC::basic::hold\(\)](#), [parameter\\_set](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 6.62.3.3 series()

```
ex GiNaC::fderivative::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

The series expansion of derivatives falls back to Taylor expansion.

See also

[basic::series](#)

Reimplemented from [GiNaC::basic](#).

References [options](#), [order](#), [r](#), and [GiNaC::basic::series\(\)](#).

### 6.62.3.4 thiscontainer() [1/2]

```
ex GiNaC::fderivative::thiscontainer (
 const exvector & v) const [override]
```

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

### 6.62.3.5 thiscontainer() [2/2]

```
ex GiNaC::fderivative::thiscontainer (
 exvector && v) const [override]
```

References [fderivative\(\)](#), [parameter\\_set](#), and [GiNaC::function::serial](#).

### 6.62.3.6 archive()

```
void GiNaC::fderivative::archive (
 archive_node & n) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::end\(\)](#), [n](#), and [parameter\\_set](#).

### 6.62.3.7 read\_archive()

```
void GiNaC::fderivative::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), and [parameter\\_set](#).

### 6.62.3.8 derivative()

```
ex GiNaC::fderivative::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for derivatives.

It applies the chain rule.

#### See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::diff\(\)](#), [fderivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [parameter\\_set](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

### 6.62.3.9 is\_equal\_same\_type()

```
bool GiNaC::fderivative::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

### 6.62.3.10 match\_same\_type()

```
bool GiNaC::fderivative::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [parameter\\_set](#).

### 6.62.3.11 derivatives()

```
const paramset & GiNaC::fderivative::derivatives () const
```

Expose this object's derivative structure.

Parameter numbers occurring more than once stand for repeated differentiation with respect to that parameter. If a symbolic function  $f(x,y)$  is differentiated with respect to  $x$ , this method will return  $\{0\}$ . If  $f(x,y)$  is differentiated twice with respect to  $y$ , it will return  $\{1,1\}$ . (This corresponds to the way this object is printed.)

Returns

multiset of function's parameter numbers that are abstractly differentiated.

References [parameter\\_set](#).

### 6.62.3.12 do\_print()

```
void GiNaC::fderivative::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.13 do\_print\_latex()

```
void GiNaC::fderivative::do_print_latex (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [order](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.14 do\_print\_csrc()

```
void GiNaC::fderivative::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [parameter\\_set](#), [GiNaC::container< C >::precedence\(\)](#), [GiNaC::function::precedence\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::function::registered\\_functions\(\)](#), and [GiNaC::function::serial](#).

### 6.62.3.15 do\_print\_tree()

```
void GiNaC::fderivative::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::container< C >::nops\(\)](#), [parameter\\_set](#), [GiNaC::function::registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::function::serial](#).

## 6.62.4 Member Data Documentation

### 6.62.4.1 parameter\_set

```
paramset GiNaC::fderivative::parameter_set [protected]
```

Set of parameter numbers with respect to which to take the derivative.

Referenced by [archive\(\)](#), [derivative\(\)](#), [derivatives\(\)](#), [do\\_print\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [fderivative\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

The documentation for this class was generated from the following files:

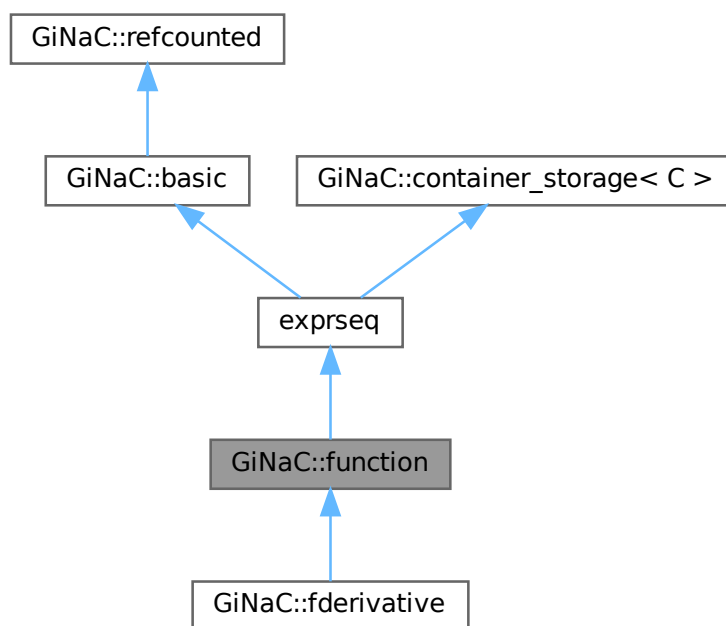
- [fderivative.h](#)
- [fderivative.cpp](#)

## 6.63 GiNaC::function Class Reference

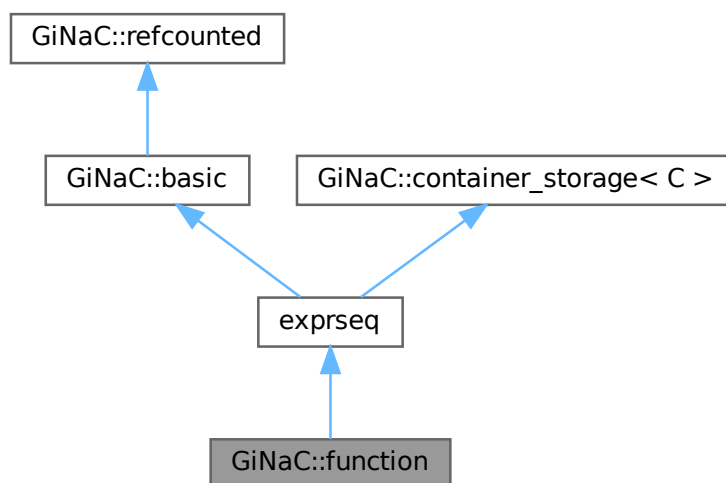
The class function is used to implement builtin functions like sin, cos... and user defined functions.

```
#include <function.h>
```

Inheritance diagram for GiNaC::function:



Collaboration diagram for GiNaC::function:



### Public Member Functions

- [function](#) (unsigned ser)

- [function](#) (unsigned ser, const [ex](#) &param1)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13)
- [function](#) (unsigned ser, const [ex](#) &param1, const [ex](#) &param2, const [ex](#) &param3, const [ex](#) &param4, const [ex](#) &param5, const [ex](#) &param6, const [ex](#) &param7, const [ex](#) &param8, const [ex](#) &param9, const [ex](#) &param10, const [ex](#) &param11, const [ex](#) &param12, const [ex](#) &param13, const [ex](#) &param14)
- [function](#) (unsigned ser, const [exprseq](#) &es)
- [function](#) (unsigned ser, const [exvector](#) &v)
- [function](#) (unsigned ser, [exvector](#) &&v)
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override  
*This method is defined to be in line with behavior of [function::return\\_type\(\)](#)*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series](#) for functions.*
- [ex thiscontainer](#) (const [exvector](#) &v) const override
- [ex thiscontainer](#) ([exvector](#) &&v) const override
- [ex conjugate](#) () const override  
*Implementation of [ex::conjugate](#) for functions.*
- [ex real\\_part](#) () const override  
*Implementation of [ex::real\\_part](#) for functions.*
- [ex imag\\_part](#) () const override

*Implementation of `ex::imag_part` for functions.*

- void `archive` (`archive_node` &n) const override

*Archive the object.*

- void `read_archive` (const `archive_node` &n, `lst` &syms) override

*Construct object from `archive_node`.*

- bool `info` (unsigned inf) const override

*Implementation of `ex::info` for functions.*

- `ex power` (const `ex` &exp) const
- unsigned `get_serial` () const
- std::string `get_name` () const

*Return the print name of the function.*

## Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (exvector::const\_iterator b, exvector::const\_iterator e)
- `container` (std::initializer\_list< `ex` > il)
- size\_t `nops` () const override
- *Number of operands/members.*
- `ex op` (size\_t i) const override
- *Return operand/member at position i.*
- `ex & let_op` (size\_t i) override
- *Return modifiable operand/member at position i.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override
- *Substitute a set of objects by arbitrary expressions.*
- `container & prepend` (const `ex` &b)
- *Add element at front.*
- `container & append` (const `ex` &b)
- *Add element at back.*
- `container & remove_first` ()
- *Remove first element.*
- `container & remove_last` ()
- *Remove last element.*
- `container & remove_all` ()
- *Remove all elements.*
- `container & sort` ()
- *Sort elements.*
- `container & unique` ()
- *Remove adjacent duplicate elements.*
- `const_iterator begin` () const
- `const_iterator end` () const
- `const_reverse_iterator rbegin` () const
- `const_reverse_iterator rend` () const

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices () const`

- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Static Public Member Functions

- static unsigned `register_new` (`function_options` const &opt)
- static unsigned `find_function` (const std::string &name, unsigned nparams)
- Find serial number of function by name and number of parameters.*
- static std::vector< `function_options` > `get_registered_functions` ()

## Static Public Attributes

- static unsigned `current_serial` = 0
- This can be used as a hook for external applications.*

### Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for functions.*
- `bool is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- `bool match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex pderivative` (unsigned diff\_param) const
- `ex expl_derivative` (const `symbol` &s) const
- `bool lookup_remember_table` (ex &result) const
- `void store_remember_table` (ex const &result) const

### Protected Member Functions inherited from `GiNaC::container< C >`

- `virtual ex thiscontainer` (const `STLT` &v) const  
*Similar to `duplicate()`, but with a preset sequence.*
- `virtual ex thiscontainer` (`STLT` &&v) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- `virtual void printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const
- `void do_print_python` (const `print_python` &c, unsigned level) const
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT subchildren` (const `exmap` &m, unsigned options=0) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Member Functions inherited from `GiNaC::container_storage< C >`

- `container_storage` ()
- `container_storage` (size\_t n, const `ex` &e)
- `container_storage` (std::initializer\_list< `ex` > il)
- `template<class In >`  
`container_storage` (In b, In e)
- `void reserve` (size\_t)
- `~container_storage` ()
- `void reserve` (size\_t n)
- `void reserve` (std::vector< `ex` > &v, size\_t n)

### Static Protected Member Functions

- static std::vector< [function\\_options](#) > & [registered\\_functions](#) ()

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) (STLT &, size\_t)

### Protected Attributes

- unsigned [serial](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

### Friends

- class [remember\\_table\\_entry](#)

### Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

## Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef C< [ex](#) > [STLT](#)

### 6.63.1 Detailed Description

The class function is used to implement builtin functions like sin, cos... and user defined functions.

### 6.63.2 Constructor & Destructor Documentation

#### 6.63.2.1 function() [1/18]

```
GiNaC::function::function (
 unsigned ser)
```

Referenced by [GiNaC::fderivative::eval\(\)](#), [evalf\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

#### 6.63.2.2 function() [2/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1)
```

#### 6.63.2.3 function() [3/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2)
```

#### 6.63.2.4 function() [4/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3)
```

#### 6.63.2.5 function() [5/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4)
```

#### 6.63.2.6 function() [6/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5)
```

#### 6.63.2.7 function() [7/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6)
```

#### 6.63.2.8 function() [8/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7)
```

#### 6.63.2.9 function() [9/18]

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8)
```

**6.63.2.10 function() [10/18]**

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9)
```

**6.63.2.11 function() [11/18]**

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9,
 const ex & param10)
```

**6.63.2.12 function() [12/18]**

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9,
 const ex & param10,
 const ex & param11)
```

**6.63.2.13 function() [13/18]**

```
GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
```

```

 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9,
 const ex & param10,
 const ex & param11,
 const ex & param12)

```

#### 6.63.2.14 [function\(\)](#) [14/18]

```

GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9,
 const ex & param10,
 const ex & param11,
 const ex & param12,
 const ex & param13)

```

#### 6.63.2.15 [function\(\)](#) [15/18]

```

GiNaC::function::function (
 unsigned ser,
 const ex & param1,
 const ex & param2,
 const ex & param3,
 const ex & param4,
 const ex & param5,
 const ex & param6,
 const ex & param7,
 const ex & param8,
 const ex & param9,
 const ex & param10,
 const ex & param11,
 const ex & param12,
 const ex & param13,
 const ex & param14)

```

#### 6.63.2.16 [function\(\)](#) [16/18]

```

GiNaC::function::function (
 unsigned ser,
 const exprseq & es)

```

References [GiNaC::basic::clearflag\(\)](#), and [GiNaC::status\\_flags::evaluated](#).

**6.63.2.17 function()** [17/18]

```
GiNaC::function::function (
 unsigned ser,
 const exvector & v)
```

**6.63.2.18 function()** [18/18]

```
GiNaC::function::function (
 unsigned ser,
 exvector && v)
```

**6.63.3 Member Function Documentation****6.63.3.1 print()**

```
void GiNaC::function::print (
 const print_context & c,
 unsigned level = 0) const [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

**Parameters**

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| <i>c</i>     | print context object that describes the output formatting                                                 |
| <i>level</i> | value that is used to identify the precedence or indentation level for placing parentheses and formatting |

Reimplemented from [GiNaC::basic](#).

References [c](#), [current\\_serial](#), [GiNaC::basic::flags](#), [GiNaC::class\\_info< OPT >::get\\_parent\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hashvalue](#), [GiNaC::function\\_options::name](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::function\\_options::nparams](#), [GiNaC::class\\_info< OPT >::options](#), [GiNaC::container< C >::precedence\(\)](#), [precedence\(\)](#), [print\(\)](#), [GiNaC::function\\_options::print\\_di](#), [GiNaC::function\\_options::print\\_use\\_exvector\\_args](#), [GiNaC::container< C >::printseq\(\)](#), [registered\\_functions\(\)](#), [GiNaC::print\\_context::s](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::TeX\\_name](#).

Referenced by [print\(\)](#).

**6.63.3.2 precedence()**

```
unsigned GiNaC::function::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), and [print\(\)](#).

### 6.63.3.3 `expand()`

```
ex GiNaC::function::expand (
 unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::expand\\_f](#), [GiNaC::expand\\_options::expand\\_function\\_args](#), [GiNaC::function\\_options::expand\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::expanded](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

### 6.63.3.4 `eval()`

```
ex GiNaC::function::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [current\\_serial](#), [GiNaC::function\\_options::eval\\_f](#), [GiNaC::function\\_options::eval\\_use\\_exvector\\_args](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::ex](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lookup\\_remember\\_table\(\)](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [store\\_remember\\_table\(\)](#), [GiNaC::function\\_options::symtree](#), [thiscontainer\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

### 6.63.3.5 `evalf()`

```
ex GiNaC::function::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GiNaC::function\\_options::evalf\\_f](#), [GiNaC::function\\_options::evalf\\_params\\_first](#), [GiNaC::function\\_options::evalf\\_use\\_exvector\\_args](#), [function\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

Referenced by [GiNaC::iterated\\_integral2\\_eval\(\)](#), and [GiNaC::iterated\\_integral3\\_eval\(\)](#).

### 6.63.3.6 `eval_ncmul()`

```
ex GiNaC::function::eval_ncmul (
 const exvector & v) const [override], [virtual]
```

This method is defined to be in line with behavior of [function::return\\_type\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::begin\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.63.3.7 calchash()**

```
unsigned GiNaC::function::calchash () const [override], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::rotate\\_left\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

**6.63.3.8 series()**

```
ex GiNaC::function::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series](#) for functions.

@see [ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [current\\_serial](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [options](#), [order](#), [r](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), [GiNaC::basic::series\(\)](#), [GiNaC::function\\_options::series\\_f](#), and [GiNaC::function\\_options::series\\_use\\_exvector\\_args](#).

**6.63.3.9 thiscontainer() [1/2]**

```
ex GiNaC::function::thiscontainer (
 const exvector & v) const [override]
```

References [function\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**6.63.3.10 thiscontainer() [2/2]**

```
ex GiNaC::function::thiscontainer (
 exvector && v) const [override]
```

References [function\(\)](#), and [serial](#).

**6.63.3.11 conjugate()**

```
ex GiNaC::function::conjugate () const [override], [virtual]
```

Implementation of [ex::conjugate](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::function\\_options::conjugate\\_f](#), [GiNaC::function\\_options::conjugate\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.12 real\_part()**

```
ex GiNaC::function::real_part () const [override], [virtual]
```

Implementation of [ex::real\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::basic::real\\_part\(\)](#), [GiNaC::function\\_options::real\\_part\\_f](#), [GiNaC::function\\_options::real\\_part\\_use\\_exvector\\_args](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.13 imag\_part()**

```
ex GiNaC::function::imag_part () const [override], [virtual]
```

Implementation of [ex::imag\\_part](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::imag\\_part\(\)](#), [GiNaC::function\\_options::imag\\_part\\_f](#), [GiNaC::function\\_options::imag\\_part\\_f](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.14 archive()**

```
void GiNaC::function::archive (
 archive_node & n) const [override], [virtual]
```

Archive the object.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [n](#), [registered\\_functions\(\)](#), and [serial](#).

**6.63.3.15 read\_archive()**

```
void GiNaC::function::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Construct object from [archive\\_node](#).

Reimplemented from [GiNaC::container< C >](#).

References [n](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.16 info()**

```
bool GiNaC::function::info (
 unsigned inf) const [override], [virtual]
```

Implementation of [ex::info](#) for functions.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::basic::info\(\)](#), [GiNaC::function\\_options::info\\_f](#), [GiNaC::function\\_options::info\\_use\\_exvector\\_arg](#), [GiNaC::function\\_options::nparams](#), [registered\\_functions\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [serial](#).

**6.63.3.17 derivative()**

```
ex GiNaC::function::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for functions.

It applies the chain rule, except for the Order term function. @see [ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::diff\(\)](#), [expl\\_derivative\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [pderivative\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.63.3.18 is\_equal\_same\_type()**

```
bool GiNaC::function::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::container< C >](#).

References [GINAC\\_ASSERT](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), and [serial](#).

### 6.63.3.19 match\_same\_type()

```
bool GiNaC::function::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

### 6.63.3.20 return\_type()

```
unsigned GiNaC::function::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

### 6.63.3.21 return\_type\_tinfo()

```
return_type_t GiNaC::function::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::return\\_type\\_tinfo](#), [GiNaC::container\\_storage< C >::seq](#), [serial](#), and [GiNaC::function\\_options::use\\_return\\_type](#).

### 6.63.3.22 pderivative()

```
ex GiNaC::function::pderivative (
 unsigned diff_param) const [protected]
```

References [GiNaC::function\\_options::derivative\\_f](#), [GiNaC::function\\_options::derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), and [GiNaC::function\\_options::nparams](#).

Referenced by [derivative\(\)](#), and [GiNaC::fderivative::eval\(\)](#).

**6.63.3.23 expl\_derivative()**

```
ex GiNaC::function::expl_derivative (
 const symbol & s) const [protected]
```

References [GiNaC::function\\_options::expl\\_derivative\\_f](#), [GiNaC::function\\_options::expl\\_derivative\\_use\\_exvector\\_args](#), [GINAC\\_ASSERT](#), and [GiNaC::function\\_options::nparams](#).

Referenced by [derivative\(\)](#).

**6.63.3.24 registered\_functions()**

```
std::vector< function_options > & GiNaC::function::registered_functions () [static], [protected]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::eval\(\)](#), [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [find\\_function\(\)](#), [get\\_name\(\)](#), [get\\_registered\\_functions\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [print\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [register\\_new\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

**6.63.3.25 lookup\_remember\_table()**

```
bool GiNaC::function::lookup_remember_table (
 ex & result) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**6.63.3.26 store\_remember\_table()**

```
void GiNaC::function::store_remember_table (
 ex const & result) const [protected]
```

References [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [serial](#).

Referenced by [eval\(\)](#).

**6.63.3.27 power()**

```
ex GiNaC::function::power (
 const ex & exp) const
```

References [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::function\\_options::nparams](#), [GiNaC::function\\_options::power\\_f](#), and [GiNaC::function\\_options::power\\_use\\_exvector\\_args](#).

**6.63.3.28 register\_new()**

```
unsigned GiNaC::function::register_new (
 function_options const & opt) [static]
```

References [GiNaC::function\\_options::functions\\_with\\_same\\_name](#), [GiNaC::function\\_options::name](#), [registered\\_functions\(\)](#), [GiNaC::function\\_options::remember\\_assoc\\_size](#), [GiNaC::function\\_options::remember\\_size](#), [GiNaC::function\\_options::remember\\_str](#), [GiNaC::remember\\_table::remember\\_tables\(\)](#), and [GiNaC::function\\_options::use\\_remember](#).

#### 6.63.3.29 find\_function()

```
unsigned GiNaC::function::find_function (
 const std::string & name,
 unsigned nparams) [static]
```

Find serial number of function by name and number of parameters.

Throws exception if function was not found.

References [registered\\_functions\(\)](#), and [serial](#).

#### 6.63.3.30 get\_registered\_functions()

```
static std::vector< function_options > GiNaC::function::get_registered_functions () [inline],
[static]
```

References [registered\\_functions\(\)](#).

#### 6.63.3.31 get\_serial()

```
unsigned GiNaC::function::get_serial () const [inline]
```

References [serial](#).

#### 6.63.3.32 get\_name()

```
std::string GiNaC::function::get_name () const
```

Return the print name of the function.

References [GINAC\\_ASSERT](#), [registered\\_functions\(\)](#), and [serial](#).

### 6.63.4 Friends And Related Symbol Documentation

#### 6.63.4.1 remember\_table\_entry

```
friend class remember_table_entry [friend]
```

### 6.63.5 Member Data Documentation

#### 6.63.5.1 current\_serial

```
unsigned GiNaC::function::current_serial = 0 [static]
```

This can be used as a hook for external applications.

Referenced by [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [print\(\)](#), and [series\(\)](#).

### 6.63.5.2 serial

`unsigned GiNaC::function::serial [protected]`

Referenced by [archive\(\)](#), [calchash\(\)](#), [conjugate\(\)](#), [GiNaC::fderivative::derivative\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::eval\(\)](#), [eval\(\)](#), [evalf\(\)](#), [expand\(\)](#), [find\\_function\(\)](#), [get\\_name\(\)](#), [get\\_serial\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [lookup\\_remember\\_table\(\)](#), [match\\_same\\_type\(\)](#), [print\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [series\(\)](#), [store\\_remember\\_table\(\)](#), [GiNaC::fderivative::thiscontainer\(\)](#), [thiscontainer\(\)](#), [GiNaC::fderivative::thiscontainer\(\)](#), and [thiscontainer\(\)](#).

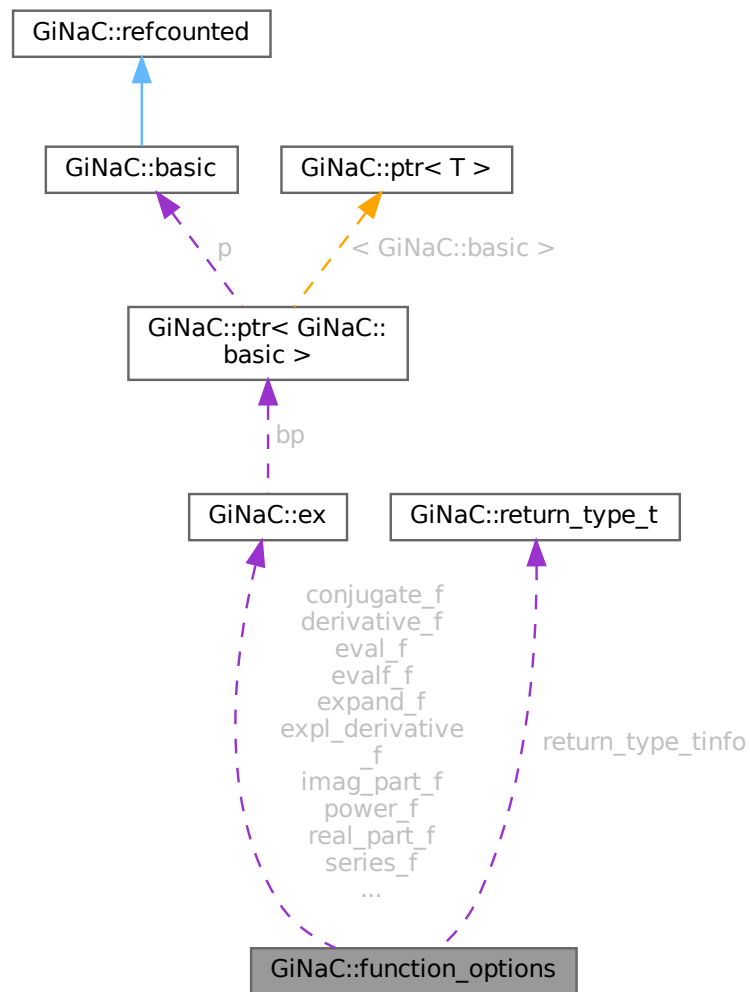
The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

## 6.64 GiNaC::function\_options Class Reference

```
#include <function.h>
```

Collaboration diagram for `GiNaC::function_options`:



## Public Member Functions

- `function_options ()`
- `function_options (std::string const &n, std::string const &tn=std::string())`
- `function_options (std::string const &n, unsigned np)`
- `~function_options ()`
- `void initialize ()`
- `function_options & dummy ()`
- `function_options & set_name (std::string const &n, std::string const &tn=std::string())`
- `function_options & latex_name (std::string const &tn)`
- `function_options & eval_func (eval_funcp_1 e)`
- `function_options & eval_func (eval_funcp_2 e)`
- `function_options & eval_func (eval_funcp_3 e)`
- `function_options & eval_func (eval_funcp_4 e)`
- `function_options & eval_func (eval_funcp_5 e)`

- Generated by Doxygen



- [function\\_options & power\\_func \(power\\_funcp\\_4 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_5 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_6 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_7 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_8 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_9 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_10 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_11 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_12 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_13 e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_14 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_1 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_2 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_3 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_4 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_5 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_6 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_7 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_8 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_9 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_10 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_11 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_12 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_13 e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_14 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_1 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_2 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_3 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_4 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_5 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_6 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_7 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_8 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_9 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_10 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_11 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_12 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_13 e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_14 e\)](#)
- [function\\_options & eval\\_func \(eval\\_funcp\\_exvector e\)](#)
- [function\\_options & evalf\\_func \(evalf\\_funcp\\_exvector e\)](#)
- [function\\_options & conjugate\\_func \(conjugate\\_funcp\\_exvector e\)](#)
- [function\\_options & real\\_part\\_func \(real\\_part\\_funcp\\_exvector e\)](#)
- [function\\_options & imag\\_part\\_func \(imag\\_part\\_funcp\\_exvector e\)](#)
- [function\\_options & expand\\_func \(expand\\_funcp\\_exvector e\)](#)
- [function\\_options & derivative\\_func \(derivative\\_funcp\\_exvector e\)](#)
- [function\\_options & expl\\_derivative\\_func \(expl\\_derivative\\_funcp\\_exvector e\)](#)
- [function\\_options & power\\_func \(power\\_funcp\\_exvector e\)](#)
- [function\\_options & series\\_func \(series\\_funcp\\_exvector e\)](#)
- [function\\_options & info\\_func \(info\\_funcp\\_exvector e\)](#)
- [template<class Ctx >  
function\\_options & print\\_func \(print\\_funcp\\_1 p\)](#)
- [template<class Ctx >  
function\\_options & print\\_func \(print\\_funcp\\_2 p\)](#)

- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_3` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_4` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_5` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_6` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_7` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_8` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_9` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_10` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_11` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_12` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_13` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_14` p)
- `template<class Ctx >`  
`function_options` & `print_func` (`print_funcp_exvector` p)
- `function_options` & `set_return_type` (`unsigned` rt, `const` `return_type_t` \*rtt=nullptr)
- `function_options` & `do_not_evalf_params` ()
- `function_options` & `remember` (`unsigned` size, `unsigned` `assoc_size`=0, `unsigned` `strategy`=`remember_strategies::delete_never`)
- `function_options` & `overloaded` (`unsigned` o)
- `function_options` & `set_symmetry` (`const` `symmetry` &s)
- `std::string` `get_name` () `const`
- `unsigned` `get_nparams` () `const`

### Protected Member Functions

- `bool` `has_derivative` () `const`
- `bool` `has_power` () `const`
- `void` `test_and_set_nparams` (`unsigned` n)
- `void` `set_print_func` (`unsigned` id, `print_funcp` f)

### Protected Attributes

- `std::string` `name`
- `std::string` `TeX_name`
- `unsigned` `nparams`
- `eval_funcp` `eval_f`
- `evalf_funcp` `evalf_f`
- `conjugate_funcp` `conjugate_f`
- `real_part_funcp` `real_part_f`
- `imag_part_funcp` `imag_part_f`
- `expand_funcp` `expand_f`
- `derivative_funcp` `derivative_f`
- `expl_derivative_funcp` `expl_derivative_f`

- [power\\_funcp power\\_f](#)
- [series\\_funcp series\\_f](#)
- [std::vector< print\\_funcp > print\\_dispatch\\_table](#)
- [info\\_funcp info\\_f](#)
- [bool evalf\\_params\\_first](#)
- [bool use\\_return\\_type](#)
- [unsigned return\\_type](#)
- [return\\_type\\_t return\\_type\\_tinfo](#)
- [bool use\\_remember](#)
- [unsigned remember\\_size](#)
- [unsigned remember\\_assoc\\_size](#)
- [unsigned remember\\_strategy](#)
- [bool eval\\_use\\_exvector\\_args](#)
- [bool evalf\\_use\\_exvector\\_args](#)
- [bool conjugate\\_use\\_exvector\\_args](#)
- [bool real\\_part\\_use\\_exvector\\_args](#)
- [bool imag\\_part\\_use\\_exvector\\_args](#)
- [bool expand\\_use\\_exvector\\_args](#)
- [bool derivative\\_use\\_exvector\\_args](#)
- [bool expl\\_derivative\\_use\\_exvector\\_args](#)
- [bool power\\_use\\_exvector\\_args](#)
- [bool series\\_use\\_exvector\\_args](#)
- [bool print\\_use\\_exvector\\_args](#)
- [bool info\\_use\\_exvector\\_args](#)
- [unsigned functions\\_with\\_same\\_name](#)
- [ex symtree](#)

## Friends

- [class function](#)
- [class fderivative](#)

## 6.64.1 Constructor & Destructor Documentation

### 6.64.1.1 function\_options() [1/3]

GiNaC::function\_options::function\_options ( )

References [initialize\(\)](#).

### 6.64.1.2 function\_options() [2/3]

```
GiNaC::function_options::function_options (
 std::string const & n,
 std::string const & tn = std::string())
```

References [initialize\(\)](#), [n](#), [print\\_func\(\)](#), and [set\\_name\(\)](#).

### 6.64.1.3 `function_options()` [3/3]

```
GiNaC::function_options::function_options (
 std::string const & n,
 unsigned np)
```

References [initialize\(\)](#), [n](#), [nparams](#), [print\\_func\(\)](#), and [set\\_name\(\)](#).

### 6.64.1.4 `~function_options()`

```
GiNaC::function_options::~~function_options ()
```

## 6.64.2 Member Function Documentation

### 6.64.2.1 `initialize()`

```
void GiNaC::function_options::initialize ()
```

References [conjugate\\_f](#), [conjugate\\_use\\_exvector\\_args](#), [derivative\\_f](#), [derivative\\_use\\_exvector\\_args](#), [eval\\_f](#), [eval\\_use\\_exvector\\_args](#), [evalf\\_f](#), [evalf\\_params\\_first](#), [evalf\\_use\\_exvector\\_args](#), [expand\\_f](#), [expand\\_use\\_exvector\\_args](#), [expl\\_derivative\\_f](#), [expl\\_derivative\\_use\\_exvector\\_args](#), [functions\\_with\\_same\\_name](#), [imag\\_part\\_f](#), [imag\\_part\\_use\\_exvector\\_args](#), [info\\_f](#), [info\\_use\\_exvector\\_args](#), [nparams](#), [power\\_f](#), [power\\_use\\_exvector\\_args](#), [print\\_use\\_exvector\\_args](#), [real\\_part\\_f](#), [real\\_part\\_use\\_exvector\\_args](#), [series\\_f](#), [series\\_use\\_exvector\\_args](#), [set\\_name\(\)](#), [symtree](#), [use\\_remember](#), and [use\\_return\\_type](#).

Referenced by [function\\_options\(\)](#), [function\\_options\(\)](#), and [function\\_options\(\)](#).

### 6.64.2.2 `dummy()`

```
function_options & GiNaC::function_options::dummy () [inline]
```

### 6.64.2.3 `set_name()`

```
function_options & GiNaC::function_options::set_name (
 std::string const & n,
 std::string const & tn = std::string())
```

References [n](#), [name](#), [print\\_func\(\)](#), and [TeX\\_name](#).

Referenced by [function\\_options\(\)](#), [function\\_options\(\)](#), and [initialize\(\)](#).

### 6.64.2.4 `latex_name()`

```
function_options & GiNaC::function_options::latex_name (
 std::string const & tn)
```

References [print\\_func\(\)](#), and [TeX\\_name](#).

**6.64.2.5 eval\_func()** [1/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_1 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.6 eval\_func()** [2/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_2 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.7 eval\_func()** [3/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_3 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.8 eval\_func()** [4/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_4 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.9 eval\_func()** [5/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_5 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.10 eval\_func()** [6/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_6 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.11 eval\_func()** [7/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_7 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.12 eval\_func()** [8/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_8 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.13 eval\_func()** [9/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_9 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.14 eval\_func()** [10/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_10 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.15 eval\_func()** [11/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_11 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.16 eval\_func()** [12/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_12 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.17 eval\_func()** [13/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_13 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.18 eval\_func()** [14/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_14 e)
```

References [eval\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.19 evalf\_func()** [1/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_1 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.20 evalf\_func()** [2/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_2 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.21 evalf\_func()** [3/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_3 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.22 evalf\_func()** [4/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_4 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.23 evalf\_func()** [5/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_5 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.24 evalf\_func()** [6/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_6 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.25 evalf\_func()** [7/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_7 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.26 evalf\_func()** [8/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_8 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.27 evalf\_func()** [9/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_9 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.28 evalf\_func()** [10/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_10 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.29 evalf\_func()** [11/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_11 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.30 evalf\_func()** [12/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_12 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.31 evalf\_func()** [13/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_13 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.32 evalf\_func()** [14/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_14 e)
```

References [evalf\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.33 conjugate\_func()** [1/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_1 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.34 conjugate\_func()** [2/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_2 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.35 conjugate\_func()** [3/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_3 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.36 conjugate\_func()** [4/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_4 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.37 conjugate\_func()** [5/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_5 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.38 conjugate\_func()** [6/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_6 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.39 conjugate\_func()** [7/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_7 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.40 conjugate\_func()** [8/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_8 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.41 conjugate\_func()** [9/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_9 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.42 conjugate\_func()** [10/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_10 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.43 conjugate\_func()** [11/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_11 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.44 conjugate\_func()** [12/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_12 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.45 conjugate\_func()** [13/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_13 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.46 conjugate\_func()** [14/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_14 e)
```

References [conjugate\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.47 real\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_1 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.48 real\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_2 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.49 real\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_3 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.50 real\_part\_func()** [4/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_4 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.51 real\_part\_func()** [5/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_5 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.52 real\_part\_func()** [6/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_6 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.53 real\_part\_func()** [7/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_7 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.54 real\_part\_func()** [8/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_8 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.55 real\_part\_func()** [9/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_9 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.56 real\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_10 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.57 real\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_11 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.58 real\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_12 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.59 real\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_13 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.60 real\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_14 e)
```

References [real\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.61 imag\_part\_func()** [1/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_1 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.62 imag\_part\_func()** [2/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_2 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.63 imag\_part\_func()** [3/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_3 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.64 imag\_part\_func()** [4/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_4 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.65 imag\_part\_func()** [5/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_5 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.66 imag\_part\_func()** [6/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_6 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.67 imag\_part\_func()** [7/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_7 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.68 imag\_part\_func()** [8/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_8 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.69 imag\_part\_func()** [9/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_9 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.70 imag\_part\_func()** [10/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_10 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.71 imag\_part\_func()** [11/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_11 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.72 imag\_part\_func()** [12/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_12 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.73 imag\_part\_func()** [13/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_13 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.74 imag\_part\_func()** [14/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_14 e)
```

References [imag\\_part\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.75 expand\_func() [1/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_1 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.76 expand\_func() [2/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_2 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.77 expand\_func() [3/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_3 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.78 expand\_func() [4/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_4 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.79 expand\_func() [5/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_5 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.80 expand\_func() [6/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_6 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.81 expand\_func() [7/15]**

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_7 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.82 expand\_func()** [8/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_8 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.83 expand\_func()** [9/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_9 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.84 expand\_func()** [10/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_10 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.85 expand\_func()** [11/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_11 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.86 expand\_func()** [12/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_12 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.87 expand\_func()** [13/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_13 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.88 expand\_func()** [14/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_14 e)
```

References [expand\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.89 derivative\_func()** [1/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_1 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.90 derivative\_func()** [2/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_2 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.91 derivative\_func()** [3/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_3 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.92 derivative\_func()** [4/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_4 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.93 derivative\_func()** [5/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_5 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.94 derivative\_func()** [6/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_6 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.95 derivative\_func()** [7/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_7 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.96 derivative\_func()** [8/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_8 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.97 derivative\_func()** [9/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_9 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.98 derivative\_func()** [10/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_10 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.99 derivative\_func()** [11/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_11 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.100 derivative\_func()** [12/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_12 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.101 derivative\_func()** [13/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_13 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.102 derivative\_func()** [14/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_14 e)
```

References [derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.103 expl\_derivative\_func() [1/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_1 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.104 expl\_derivative\_func() [2/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_2 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.105 expl\_derivative\_func() [3/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_3 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.106 expl\_derivative\_func() [4/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_4 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.107 expl\_derivative\_func() [5/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_5 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.108 expl\_derivative\_func() [6/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_6 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.109 expl\_derivative\_func() [7/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_7 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.110 expl\_derivative\_func() [8/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_8 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.111 expl\_derivative\_func() [9/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_9 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.112 expl\_derivative\_func() [10/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_10 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.113 expl\_derivative\_func() [11/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_11 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.114 expl\_derivative\_func() [12/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_12 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.115 expl\_derivative\_func() [13/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_13 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.116 expl\_derivative\_func() [14/15]**

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_14 e)
```

References [expl\\_derivative\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.117 power\_func() [1/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_1 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.118 power\_func() [2/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_2 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.119 power\_func() [3/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_3 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.120 power\_func() [4/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_4 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.121 power\_func() [5/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_5 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.122 power\_func() [6/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_6 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.123 power\_func() [7/15]**

```
function_options & GiNaC::function_options::power_func (
 power_funcp_7 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.124 power\_func()** [8/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_8 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.125 power\_func()** [9/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_9 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.126 power\_func()** [10/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_10 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.127 power\_func()** [11/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_11 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.128 power\_func()** [12/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_12 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.129 power\_func()** [13/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_13 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.130 power\_func()** [14/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_14 e)
```

References [power\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.131 series\_func() [1/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_1 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.132 series\_func() [2/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_2 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.133 series\_func() [3/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_3 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.134 series\_func() [4/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_4 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.135 series\_func() [5/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_5 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.136 series\_func() [6/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_6 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.137 series\_func() [7/15]**

```
function_options & GiNaC::function_options::series_func (
 series_funcp_7 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.138 series\_func()** [8/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_8 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.139 series\_func()** [9/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_9 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.140 series\_func()** [10/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_10 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.141 series\_func()** [11/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_11 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.142 series\_func()** [12/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_12 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.143 series\_func()** [13/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_13 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.144 series\_func()** [14/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_14 e)
```

References [series\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.145 info\_func() [1/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_1 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.146 info\_func() [2/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_2 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.147 info\_func() [3/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_3 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.148 info\_func() [4/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_4 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.149 info\_func() [5/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_5 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.150 info\_func() [6/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_6 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.151 info\_func() [7/15]**

```
function_options & GiNaC::function_options::info_func (
 info_funcp_7 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.152 info\_func()** [8/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_8 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.153 info\_func()** [9/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_9 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.154 info\_func()** [10/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_10 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.155 info\_func()** [11/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_11 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.156 info\_func()** [12/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_12 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.157 info\_func()** [13/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_13 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.158 info\_func()** [14/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_14 e)
```

References [info\\_f](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.159 eval\_func()** [15/15]

```
function_options & GiNaC::function_options::eval_func (
 eval_funcp_exvector e)
```

References [eval\\_f](#), and [eval\\_use\\_exvector\\_args](#).

**6.64.2.160 evalf\_func()** [15/15]

```
function_options & GiNaC::function_options::evalf_func (
 evalf_funcp_exvector e)
```

References [evalf\\_f](#), and [evalf\\_use\\_exvector\\_args](#).

**6.64.2.161 conjugate\_func()** [15/15]

```
function_options & GiNaC::function_options::conjugate_func (
 conjugate_funcp_exvector e)
```

References [conjugate\\_f](#), and [conjugate\\_use\\_exvector\\_args](#).

**6.64.2.162 real\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::real_part_func (
 real_part_funcp_exvector e)
```

References [real\\_part\\_f](#), and [real\\_part\\_use\\_exvector\\_args](#).

**6.64.2.163 imag\_part\_func()** [15/15]

```
function_options & GiNaC::function_options::imag_part_func (
 imag_part_funcp_exvector e)
```

References [imag\\_part\\_f](#), and [imag\\_part\\_use\\_exvector\\_args](#).

**6.64.2.164 expand\_func()** [15/15]

```
function_options & GiNaC::function_options::expand_func (
 expand_funcp_exvector e)
```

References [expand\\_f](#), and [expand\\_use\\_exvector\\_args](#).

**6.64.2.165 derivative\_func()** [15/15]

```
function_options & GiNaC::function_options::derivative_func (
 derivative_funcp_exvector e)
```

References [derivative\\_f](#), and [derivative\\_use\\_exvector\\_args](#).

**6.64.2.166** `expl_derivative_func()` [15/15]

```
function_options & GiNaC::function_options::expl_derivative_func (
 expl_derivative_funcp_exvector e)
```

References [expl\\_derivative\\_f](#), and [expl\\_derivative\\_use\\_exvector\\_args](#).

**6.64.2.167** `power_func()` [15/15]

```
function_options & GiNaC::function_options::power_func (
 power_funcp_exvector e)
```

References [power\\_f](#), and [power\\_use\\_exvector\\_args](#).

**6.64.2.168** `series_func()` [15/15]

```
function_options & GiNaC::function_options::series_func (
 series_funcp_exvector e)
```

References [series\\_f](#), and [series\\_use\\_exvector\\_args](#).

**6.64.2.169** `info_func()` [15/15]

```
function_options & GiNaC::function_options::info_func (
 info_funcp_exvector e)
```

References [info\\_f](#), and [info\\_use\\_exvector\\_args](#).

**6.64.2.170** `print_func()` [1/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_1 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

Referenced by [function\\_options\(\)](#), [function\\_options\(\)](#), [latex\\_name\(\)](#), [remember\(\)](#), [set\\_name\(\)](#), and [set\\_return\\_type\(\)](#).

**6.64.2.171** `print_func()` [2/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_2 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.172 print\_func()** [3/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_3 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.173 print\_func()** [4/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_4 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.174 print\_func()** [5/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_5 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.175 print\_func()** [6/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_6 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.176 print\_func()** [7/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_7 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.177 print\_func()** [8/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_8 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.178 print\_func()** [9/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_9 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.179 print\_func()** [10/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_10 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.180 print\_func()** [11/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_11 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.181 print\_func()** [12/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_12 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.182 print\_func()** [13/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_13 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.183 print\_func()** [14/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_14 p) [inline]
```

References [options](#), [set\\_print\\_func\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

**6.64.2.184 print\_func()** [15/15]

```
template<class Ctx >
function_options & GiNaC::function_options::print_func (
 print_funcp_exvector p) [inline]
```

References [options](#), [print\\_use\\_exvector\\_args](#), and [set\\_print\\_func\(\)](#).

**6.64.2.185 set\_return\_type()**

```
function_options & GiNaC::function_options::set_return_type (
 unsigned rt,
 const return_type_t * rtt = nullptr)
```

References [print\\_func\(\)](#), [return\\_type](#), [return\\_type\\_tinfo](#), and [use\\_return\\_type](#).

**6.64.2.186 do\_not\_evalf\_params()**

```
function_options & GiNaC::function_options::do_not_evalf_params ()
```

References [evalf\\_params\\_first](#).

**6.64.2.187 remember()**

```
function_options & GiNaC::function_options::remember (
 unsigned size,
 unsigned assoc_size = 0,
 unsigned strategy = remember_strategies::delete_never)
```

References [print\\_func\(\)](#), [remember\\_assoc\\_size](#), [remember\\_size](#), [remember\\_strategy](#), and [use\\_remember](#).

**6.64.2.188 overloaded()**

```
function_options & GiNaC::function_options::overloaded (
 unsigned o)
```

References [functions\\_with\\_same\\_name](#).

**6.64.2.189 set\_symmetry()**

```
function_options & GiNaC::function_options::set_symmetry (
 const symmetry & s)
```

References [symtree](#).



### 6.64.2.195 set\_print\_func()

```
void GiNaC::function_options::set_print_func (
 unsigned id,
 print_funcp f) [protected]
```

References [print\\_dispatch\\_table](#).

Referenced by [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), [print\\_func\(\)](#), and [print\\_func\(\)](#).

## 6.64.3 Friends And Related Symbol Documentation

### 6.64.3.1 function

```
friend class function [friend]
```

### 6.64.3.2 fderivative

```
friend class fderivative [friend]
```

## 6.64.4 Member Data Documentation

### 6.64.4.1 name

```
std::string GiNaC::function_options::name [protected]
```

Referenced by [get\\_name\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::register\\_new\(\)](#), [set\\_name\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

### 6.64.4.2 TeX\_name

```
std::string GiNaC::function_options::TeX_name [protected]
```

Referenced by [latex\\_name\(\)](#), [GiNaC::function::print\(\)](#), and [set\\_name\(\)](#).

### 6.64.4.3 nparams

```
unsigned GiNaC::function_options::nparams [protected]
```

Referenced by [GiNaC::function::conjugate\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::function::expl\\_derivative\(\)](#), [function\\_options\(\)](#), [get\\_nparams\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::function::info\(\)](#), [initialize\(\)](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::function::series\(\)](#), and [test\\_and\\_set\\_nparams\(\)](#).

#### 6.64.4.4 eval\_f

`eval_funcp` `GiNaC::function_options::eval_f` [protected]

Referenced by `GiNaC::function::eval()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, `eval_func()`, and `initialize()`.

#### 6.64.4.5 evalf\_f

`evalf_funcp` `GiNaC::function_options::evalf_f` [protected]

Referenced by `GiNaC::function::evalf()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, `evalf_func()`, and `initialize()`.

#### 6.64.4.6 conjugate\_f

`conjugate_funcp` `GiNaC::function_options::conjugate_f` [protected]

Referenced by `GiNaC::function::conjugate()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, `conjugate_func()`, and `initialize()`.

#### 6.64.4.7 real\_part\_f

`real_part_funcp` `GiNaC::function_options::real_part_f` [protected]

Referenced by `initialize()`, `GiNaC::function::real_part()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, `real_part_func()`, and `real_part_func()`.

#### 6.64.4.8 imag\_part\_f

`imag_part_funcp` `GiNaC::function_options::imag_part_f` [protected]

Referenced by `GiNaC::function::imag_part()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, `imag_part_func()`, and `initialize()`.

#### 6.64.4.9 expand\_f

`expand_funcp` `GiNaC::function_options::expand_f` [protected]

Referenced by `GiNaC::function::expand()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, `expand_func()`, and `initialize()`.

**6.64.4.10 derivative\_f**

`derivative_funcp` GiNaC::function\_options::derivative\_f [protected]

Referenced by `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `derivative_func()`, `has_derivative()`, `initialize()`, and `GiNaC::function::pderivative()`.

**6.64.4.11 expl\_derivative\_f**

`expl_derivative_funcp` GiNaC::function\_options::expl\_derivative\_f [protected]

Referenced by `GiNaC::function::expl_derivative()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, `expl_derivative_func()`, and `initialize()`.

**6.64.4.12 power\_f**

`power_funcp` GiNaC::function\_options::power\_f [protected]

Referenced by `has_power()`, `initialize()`, `GiNaC::function::power()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, `power_func()`, and `power_func()`.

**6.64.4.13 series\_f**

`series_funcp` GiNaC::function\_options::series\_f [protected]

Referenced by `initialize()`, `GiNaC::function::series()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, `series_func()`, and `series_func()`.

**6.64.4.14 print\_dispatch\_table**

`std::vector<print_funcp>` GiNaC::function\_options::print\_dispatch\_table [protected]

Referenced by `GiNaC::function::print()`, and `set_print_func()`.

**6.64.4.15 info\_f**

`info_funcp` GiNaC::function\_options::info\_f [protected]

Referenced by `GiNaC::function::info()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, `info_func()`, and `initialize()`.

#### 6.64.4.16 evalf\_params\_first

`bool` GiNaC::function\_options::evalf\_params\_first [protected]

Referenced by [do\\_not\\_evalf\\_params\(\)](#), [GiNaC::function::evalf\(\)](#), and [initialize\(\)](#).

#### 6.64.4.17 use\_return\_type

`bool` GiNaC::function\_options::use\_return\_type [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.18 return\_type

`unsigned` GiNaC::function\_options::return\_type [protected]

Referenced by [GiNaC::function::return\\_type\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.19 return\_type\_tinfo

`return_type_t` GiNaC::function\_options::return\_type\_tinfo [protected]

Referenced by [GiNaC::function::return\\_type\\_tinfo\(\)](#), and [set\\_return\\_type\(\)](#).

#### 6.64.4.20 use\_remember

`bool` GiNaC::function\_options::use\_remember [protected]

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.21 remember\_size

`unsigned` GiNaC::function\_options::remember\_size [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.22 remember\_assoc\_size

`unsigned` GiNaC::function\_options::remember\_assoc\_size [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.23 remember\_strategy

`unsigned` GiNaC::function\_options::remember\_strategy [protected]

Referenced by [GiNaC::function::register\\_new\(\)](#), and [remember\(\)](#).

#### 6.64.4.24 eval\_use\_exvector\_args

`bool` GiNaC::function\_options::eval\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::eval\(\)](#), [eval\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.25 evalf\_use\_exvector\_args

`bool` GiNaC::function\_options::evalf\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::evalf\(\)](#), [evalf\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.26 conjugate\_use\_exvector\_args

`bool` GiNaC::function\_options::conjugate\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::conjugate\(\)](#), [conjugate\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.27 real\_part\_use\_exvector\_args

`bool` GiNaC::function\_options::real\_part\_use\_exvector\_args [protected]

Referenced by [initialize\(\)](#), [GiNaC::function::real\\_part\(\)](#), and [real\\_part\\_func\(\)](#).

#### 6.64.4.28 imag\_part\_use\_exvector\_args

`bool` GiNaC::function\_options::imag\_part\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::imag\\_part\(\)](#), [imag\\_part\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.29 expand\_use\_exvector\_args

`bool` GiNaC::function\_options::expand\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::expand\(\)](#), [expand\\_func\(\)](#), and [initialize\(\)](#).

#### 6.64.4.30 derivative\_use\_exvector\_args

`bool` GiNaC::function\_options::derivative\_use\_exvector\_args [protected]

Referenced by [derivative\\_func\(\)](#), [initialize\(\)](#), and [GiNaC::function::pderivative\(\)](#).

#### 6.64.4.31 expl\_derivative\_use\_exvector\_args

`bool` GiNaC::function\_options::expl\_derivative\_use\_exvector\_args [protected]

Referenced by [GiNaC::function::expl\\_derivative\(\)](#), [expl\\_derivative\\_func\(\)](#), and [initialize\(\)](#).

**6.64.4.32 power\_use\_exvector\_args**

```
bool GiNaC::function_options::power_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::power\(\)](#), and [power\\_func\(\)](#).

**6.64.4.33 series\_use\_exvector\_args**

```
bool GiNaC::function_options::series_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::series\(\)](#), and [series\\_func\(\)](#).

**6.64.4.34 print\_use\_exvector\_args**

```
bool GiNaC::function_options::print_use_exvector_args [protected]
```

Referenced by [initialize\(\)](#), [GiNaC::function::print\(\)](#), and [print\\_func\(\)](#).

**6.64.4.35 info\_use\_exvector\_args**

```
bool GiNaC::function_options::info_use_exvector_args [protected]
```

Referenced by [GiNaC::function::info\(\)](#), [info\\_func\(\)](#), and [initialize\(\)](#).

**6.64.4.36 functions\_with\_same\_name**

```
unsigned GiNaC::function_options::functions_with_same_name [protected]
```

Referenced by [initialize\(\)](#), [overloaded\(\)](#), and [GiNaC::function::register\\_new\(\)](#).

**6.64.4.37 symtree**

```
ex GiNaC::function_options::symtree [protected]
```

Referenced by [GiNaC::function::eval\(\)](#), [initialize\(\)](#), and [set\\_symmetry\(\)](#).

The documentation for this class was generated from the following files:

- [function.h](#)
- [function.cpp](#)

**6.65 GiNaC::G2\_SERIAL Class Reference**

Generalized multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

## 6.65.1 Detailed Description

Generalized multiple polylogarithm.

## 6.65.2 Member Data Documentation

### 6.65.2.1 serial

```
unsigned GiNaC::G2_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("G", 2).
 evalf_func(G2_evalf).
 eval_func(G2_eval).
 overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.66 GiNaC::G3\_SERIAL Class Reference

Generalized multiple polylogarithm with explicit imaginary parts.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

## 6.66.1 Detailed Description

Generalized multiple polylogarithm with explicit imaginary parts.

## 6.66.2 Member Data Documentation

### 6.66.2.1 serial

```
unsigned GiNaC::G3_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("G", 3).
 evalf_func(G3_evalf).
 eval_func(G3_eval).
 overloaded(2))
```

Referenced by [GiNaC::G\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.67 GiNaC::gcd\_options Struct Reference

Flags to control the behavior of [gcd\(\)](#) and friends.

```
#include <normal.h>
```

### Public Types

- enum { [no\\_heur\\_gcd](#) = 2 , [no\\_part\\_factored](#) = 4 , [use\\_sr\\_gcd](#) = 8 }

### 6.67.1 Detailed Description

Flags to control the behavior of [gcd\(\)](#) and friends.

## 6.67.2 Member Enumeration Documentation

### 6.67.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|                                  |                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">no_heur_gcd</a>      | Usually <a href="#">GiNaC</a> tries heuristic GCD first, because typically it's much faster than anything else. Even if heuristic algorithm fails, the overhead is negligible w.r.t. cost of computing the GCD by some other method. However, some people dislike it, so here's a flag which tells <a href="#">GiNaC</a> to NOT use the heuristic algorithm. |
| <a href="#">no_part_factored</a> | <a href="#">GiNaC</a> tries to avoid expanding expressions when computing GCDs. This is a good idea, but some people dislike it. Hence the flag to disable special handling of partially factored polynomials. DON'T SET THIS unless you <i>really</i> know what are you doing!                                                                              |
| <a href="#">use_sr_gcd</a>       | By default <a href="#">GiNaC</a> uses modular GCD algorithm. Typically it's much faster than PRS (pseudo remainder sequence) algorithm. This flag forces <a href="#">GiNaC</a> to use PRS algorithm                                                                                                                                                          |

The documentation for this struct was generated from the following file:

- [normal.h](#)

## 6.68 GiNaC::gcdheu\_failed Class Reference

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

### 6.68.1 Detailed Description

Exception thrown by [heur\\_gcd\(\)](#) to signal failure.

The documentation for this class was generated from the following file:

- [normal.cpp](#)

## 6.69 GiNaC::has\_distance< T > Class Template Reference

SFINAE test for distance.

```
#include <utils_multi_iterator.h>
```

### Public Types

- enum { [value](#) = sizeof(test<T>(0)) == sizeof(yes\_type) }

### Private Types

- typedef char [yes\\_type](#)[1]
- typedef char [no\\_type](#)[2]

### Static Private Member Functions

- template<typename C >  
static [yes\\_type](#) & [test](#) (decltype(std::distance< C >))
- template<typename C >  
static [no\\_type](#) & [test](#) (...)

### 6.69.1 Detailed Description

```
template<typename T>
class GiNaC::has_distance< T >
```

SFINAE test for distance.

## 6.69.2 Member Typedef Documentation

### 6.69.2.1 yes\_type

```
template<typename T >
typedef char GiNaC::has_distance< T >::yes_type[1] [private]
```

### 6.69.2.2 no\_type

```
template<typename T >
typedef char GiNaC::has_distance< T >::no_type[2] [private]
```

## 6.69.3 Member Enumeration Documentation

### 6.69.3.1 anonymous enum

```
template<typename T >
anonymous enum
```

Enumerator

|       |  |
|-------|--|
| value |  |
|-------|--|

## 6.69.4 Member Function Documentation

### 6.69.4.1 test() [1/2]

```
template<typename T >
template<typename C >
static yes_type & GiNaC::has_distance< T >::test (
 decltype(std::distance< C >)) [static], [private]
```

### 6.69.4.2 test() [2/2]

```
template<typename T >
template<typename C >
static no_type & GiNaC::has_distance< T >::test (
 ...) [static], [private]
```

The documentation for this class was generated from the following file:

- [utils\\_multi\\_iterator.h](#)

## 6.70 GiNaC::has\_options Class Reference

Flags to control the behavior of `has()`.

```
#include <flags.h>
```

### Public Types

- enum { `algebraic` = 0x0001 }

### 6.70.1 Detailed Description

Flags to control the behavior of `has()`.

### 6.70.2 Member Enumeration Documentation

#### 6.70.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|                        |                                        |
|------------------------|----------------------------------------|
| <code>algebraic</code> | <code>enable algebraic matching</code> |
|------------------------|----------------------------------------|

The documentation for this class was generated from the following file:

- `flags.h`

## 6.71 `std::hash< GiNaC::ex >` Struct Reference

Specialization of `std::hash()` for `ex` objects.

```
#include <ex.h>
```

### Public Member Functions

- `std::size_t operator()` (const `GiNaC::ex` &`e`) const noexcept

### 6.71.1 Detailed Description

Specialization of `std::hash()` for `ex` objects.

## 6.71.2 Member Function Documentation

### 6.71.2.1 operator()

```
std::size_t std::hash< GiNaC::ex >::operator() (
 const GiNaC::ex & e) const [inline], [noexcept]
```

The documentation for this struct was generated from the following file:

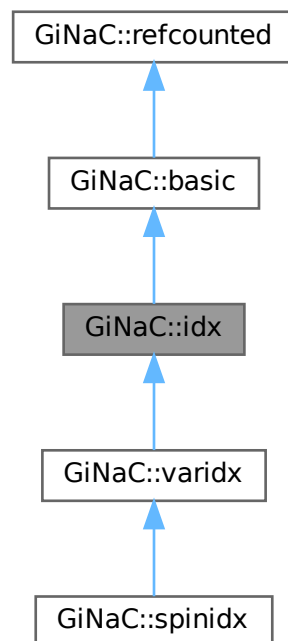
- [ex.h](#)

## 6.72 GiNaC::idx Class Reference

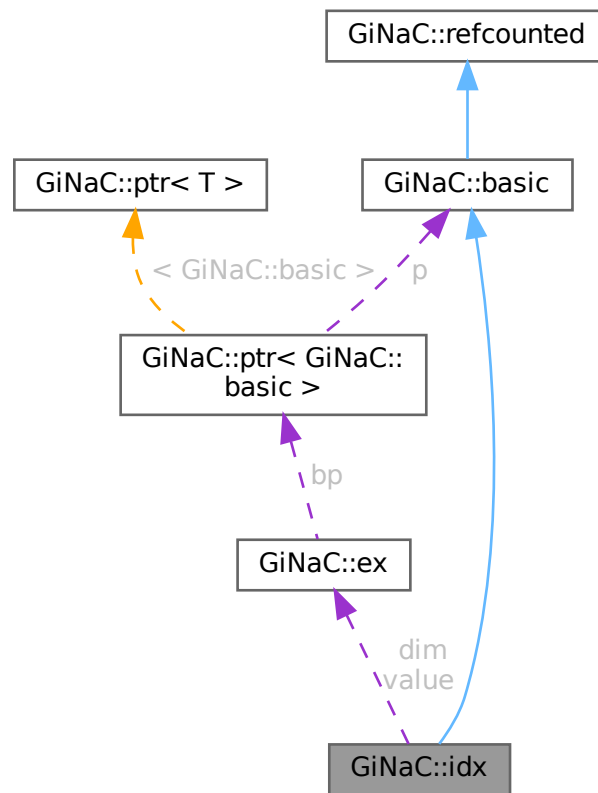
This class holds one index of an indexed object.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::idx:



Collaboration diagram for GiNaC::idx:



## Public Member Functions

- `idx` (const `ex` &`v`, const `ex` &`dim`)  
Construct index with given value and dimension.
- bool `info` (unsigned `inf`) const override  
Information about the object.
- `size_t nops` () const override  
Number of operands/members.
- `ex op` (size\_t `i`) const override  
Return operand/member at position `i`.
- `ex map` (`map_function` &`f`) const override  
Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).
- `ex evalf` () const override  
By default, `basic::evalf` would evaluate the index value but we don't want `a.1` to become `a`.
- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override  
Substitute a set of objects by arbitrary expressions.
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override

- Load (deserialize) the object from an archive node.*

  - virtual bool `is_dummy_pair_same_type` (const `basic` &other) const

*Check whether the index forms a dummy index pair with another index of the same type.*
- `ex get_value` () const
- Get value of index.*
- bool `is_numeric` () const
- Check whether the index is numeric.*
- bool `is_symbolic` () const
- Check whether the index is symbolic.*
- `ex get_dim` () const
- Get dimension of index space.*
- bool `is_dim_numeric` () const
- Check whether the dimension is numeric.*
- bool `is_dim_symbolic` () const
- Check whether the dimension is symbolic.*
- `ex replace_dim` (const `ex` &new\_dim) const
- Make a new index with the same value but a different dimension.*
- `ex minimal_dim` (const `idx` &other) const
- Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()
- basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)
- basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const
- Create a clone of this object on the heap.*
- virtual `ex eval` () const
- Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalm` () const
- Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const
- Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const
- Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const
- Output to stream.*
- virtual void `dbgprint` () const
- Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const
- Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const
- Return relative operator precedence (for parenthezing output).*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)
- Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)

- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const

- *Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an index always returns 0.*
- `bool match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void print_index` (const `print_context` &c, unsigned level) const
- `void do_print` (const `print_context` &c, unsigned level) const
- `void do_print_csrc` (const `print_csrc` &c, unsigned level) const
- `void do_print_latex` (const `print_latex` &c, unsigned level) const
- `void do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &v) const
- `virtual int compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex value](#)  
*Expression that constitutes the index (numeric or symbolic name)*
- [ex dim](#)  
*Dimension of space (can be symbolic or numeric)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.72.1 Detailed Description

This class holds one index of an indexed object.

Indices can theoretically consist of any symbolic expression but they are usually only just a symbol (e.g. "mu", "i") or numeric (integer). Indices belong to a space with a certain numeric or symbolic dimension.

### 6.72.2 Constructor & Destructor Documentation

#### 6.72.2.1 idx()

```
GiNaC::idx::idx (
 const ex & v,
 const ex & dim) [explicit]
```

Construct index with given value and dimension.

#### Parameters

|            |                                                |
|------------|------------------------------------------------|
| <i>v</i>   | Value of index (numeric or symbolic)           |
| <i>dim</i> | Dimension of index space (numeric or symbolic) |

References [dim](#), [GiNaC::ex::info\(\)](#), [is\\_dim\\_numeric\(\)](#), and [GiNaC::info\\_flags::posint](#).

### 6.72.3 Member Function Documentation

#### 6.72.3.1 info()

```
bool GiNaC::idx::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::has\\_indices](#), and [GiNaC::info\\_flags::idx](#).

### 6.72.3.2 nops()

```
size_t GiNaC::idx::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.72.3.3 op()

```
ex GiNaC::idx::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [value](#).

### 6.72.3.4 map()

```
ex GiNaC::idx::map (
 map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::status\\_flags::hash\\_calculate](#) and [value](#).

### 6.72.3.5 evalf()

```
ex GiNaC::idx::evalf () const [override], [virtual]
```

By default, [basic::evalf](#) would evaluate the index value but we don't want a.1 to become a.

(1.0).

Reimplemented from [GiNaC::basic](#).

### 6.72.3.6 subs()

```
ex GiNaC::idx::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::basic::duplicate\(\)](#), [GiNaC::ex::find\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [m](#), [options](#), [GiNaC::subs\\_options::really\\_subs\\_idx](#), [GiNaC::ex::subs\(\)](#), and [value](#).

### 6.72.3.7 archive()

```
void GiNaC::idx::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Loosely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [n](#), and [value](#).

### 6.72.3.8 read\_archive()

```
void GiNaC::idx::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [n](#), and [value](#).

### 6.72.3.9 derivative()

```
ex GiNaC::idx::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an index always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

### 6.72.3.10 match\_same\_type()

```
bool GiNaC::idx::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [GINAC\\_ASSERT](#), and [GiNaC::ex::is\\_equal\(\)](#).

### 6.72.3.11 calchash()

```
unsigned GiNaC::idx::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.72.3.12 is\_dummy\_pair\_same\_type()**

```
bool GiNaC::idx::is_dummy_pair_same_type (
 const basic & other) const [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented in [GiNaC::varidx](#), and [GiNaC::spinidx](#).

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), and [value](#).

Referenced by [GiNaC::is\\_dummy\\_pair\(\)](#).

**6.72.3.13 get\_value()**

```
ex GiNaC::idx::get_value () const [inline]
```

Get value of index.

References [value](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), and [GiNaC::spinmetric::eval\\_indexed\(\)](#).

**6.72.3.14 is\_numeric()**

```
bool GiNaC::idx::is_numeric () const [inline]
```

Check whether the index is numeric.

References [value](#).

**6.72.3.15 is\_symbolic()**

```
bool GiNaC::idx::is_symbolic () const [inline]
```

Check whether the index is symbolic.

References [value](#).

Referenced by [GiNaC::spinmetric::contract\\_with\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**6.72.3.16 get\_dim()**

```
ex GiNaC::idx::get_dim () const [inline]
```

Get dimension of index space.

References [dim](#).

Referenced by [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), and [GiNaC::tensmetric::eval\\_indexed\(\)](#).

**6.72.3.17 is\_dim\_numeric()**

```
bool GiNaC::idx::is_dim_numeric () const [inline]
```

Check whether the dimension is numeric.

References [dim](#).

Referenced by [idx\(\)](#).

**6.72.3.18 is\_dim\_symbolic()**

```
bool GiNaC::idx::is_dim_symbolic () const [inline]
```

Check whether the dimension is symbolic.

References [dim](#).

**6.72.3.19 replace\_dim()**

```
ex GiNaC::idx::replace_dim (
 const ex & new_dim) const
```

Make a new index with the same value but a different dimension.

References [GiNaC::basic::clearflag\(\)](#), [dim](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**6.72.3.20 minimal\_dim()**

```
ex GiNaC::idx::minimal_dim (
 const idx & other) const
```

Return the minimum of the dimensions of this and another index.

If this is undecidable, throw an exception.

References [dim](#), and [GiNaC::minimal\\_dim\(\)](#).

Referenced by [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

**6.72.3.21 print\_index()**

```
void GiNaC::idx::print_index (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [dim](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_options::print\\_index\\_dimensions](#), and [value](#).

Referenced by [do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [GiNaC::spinidx::do\\_print\\_latex\(\)](#).

### 6.72.3.22 do\_print()

```
void GiNaC::idx::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_index\(\)](#).

### 6.72.3.23 do\_print\_csrc()

```
void GiNaC::idx::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::print\(\)](#), and [value](#).

### 6.72.3.24 do\_print\_latex()

```
void GiNaC::idx::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_index\(\)](#).

### 6.72.3.25 do\_print\_tree()

```
void GiNaC::idx::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [value](#).

## 6.72.4 Member Data Documentation

### 6.72.4.1 value

```
ex GiNaC::idx::value [protected]
```

Expression that constitutes the index (numeric or symbolic name)

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [get\\_value\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_numeric\(\)](#), [is\\_symbolic\(\)](#), [map\(\)](#), [op\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

#### 6.72.4.2 dim

```
ex GiNaC::idx::dim [protected]
```

Dimension of space (can be symbolic or numeric)

Referenced by [archive\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [get\\_dim\(\)](#), [idx\(\)](#), [is\\_dim\\_numeric\(\)](#), [is\\_dim\\_symbolic\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [minimal\\_dim\(\)](#), [print\\_index\(\)](#), [read\\_archive\(\)](#), and [replace\\_dim\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

### 6.73 GiNaC::idx\_is\_equal\_ignore\_dim Struct Reference

#### Public Member Functions

- `bool operator() (const ex &lh, const ex &rh) const`

#### 6.73.1 Member Function Documentation

##### 6.73.1.1 operator()

```
bool GiNaC::idx_is_equal_ignore_dim::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#).

The documentation for this struct was generated from the following file:

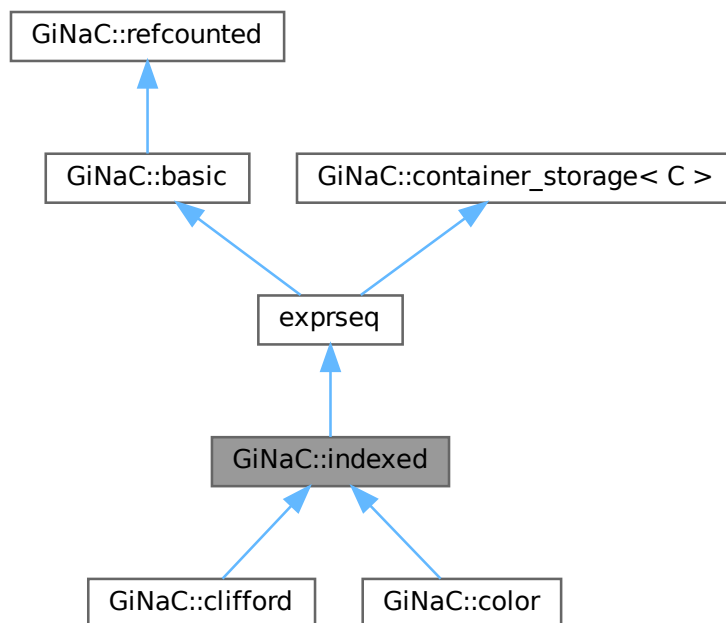
- [indexed.cpp](#)

## 6.74 GiNaC::indexed Class Reference

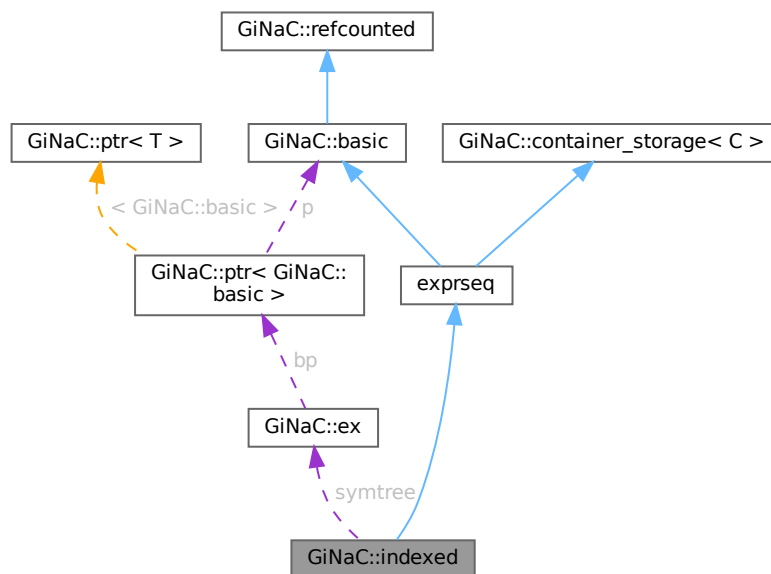
This class holds an indexed expression.

```
#include <indexed.h>
```

Inheritance diagram for GiNaC::indexed:



Collaboration diagram for `GiNaC::indexed`:



## Public Member Functions

- `indexed` (const `ex` &b)  
Construct indexed object with no index.
- `indexed` (const `ex` &b, const `ex` &i1)  
Construct indexed object with one index.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2)  
Construct indexed object with two indices.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
Construct indexed object with three indices.
- `indexed` (const `ex` &b, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
Construct indexed object with four indices.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2)  
Construct indexed object with two indices and a specified symmetry.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3)  
Construct indexed object with three indices and a specified symmetry.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `ex` &i1, const `ex` &i2, const `ex` &i3, const `ex` &i4)  
Construct indexed object with four indices and a specified symmetry.
- `indexed` (const `ex` &b, const `exvector` &iv)  
Construct indexed object with a specified vector of indices.
- `indexed` (const `ex` &b, const `symmetry` &symm, const `exvector` &iv)  
Construct indexed object with a specified vector of indices and symmetry.
- `indexed` (const `symmetry` &symm, const `exprseq` &es)
- `indexed` (const `symmetry` &symm, const `exvector` &v)
- `indexed` (const `symmetry` &symm, `exvector` &&v)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).

- `bool info` (unsigned inf) const override  
*Information about the object.*
- `ex eval ()` const override  
*Perform automatic non-interruptive term rewriting rules.*
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- `void archive (archive_node &n)` const override  
*Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &symbols)` override  
*Read (a.k.a.*
- `bool all_index_values_are` (unsigned inf) const  
*Check whether all index values have a certain property.*
- `exvector get_indices ()` const  
*Return a vector containing the object's indices.*
- `exvector get_dummy_indices ()` const  
*Return a vector containing the dummy indices of the object, if any.*
- `exvector get_dummy_indices` (const indexed &other) const  
*Return a vector containing the dummy indices in the contraction with another indexed object.*
- `bool has_dummy_index_for` (const ex &i) const  
*Check whether the object has an index that forms a dummy index pair with a given index.*
- `ex get_symmetry ()` const  
*Return symmetry properties.*

### Public Member Functions inherited from `GiNaC::container< C >`

- `container` (STLT const &s)
- `container` (STLT &&v)
- `container` (exvector::const\_iterator b, exvector::const\_iterator e)
- `container` (std::initializer\_list< ex > il)
- `size_t nops ()` const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t i) override  
*Return modifiable operand/member at position i.*
- `ex subs` (const exmap &m, unsigned options=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `container & prepend` (const ex &b)  
*Add element at front.*
- `container & append` (const ex &b)  
*Add element at back.*
- `container & remove_first ()`  
*Remove first element.*
- `container & remove_last ()`  
*Remove last element.*
- `container & remove_all ()`  
*Remove all elements.*
- `container & sort ()`

- Sort elements.
- `container & unique ()`
- Remove adjacent duplicate elements.
- `const_iterator begin () const`
- `const_iterator end () const`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex collect (const ex &s, bool distributed=false) const`

- Sort expanded expression in terms of powers of some object(s).
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
Default implementation of `ex::series()`.
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
Default implementation of `ex::normal()`.
- virtual `ex to_rational` (`exmap` &repl) const  
Default implementation of `ex::to_rational()`.
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
Apply symmetric modular homomorphism to an expanded multivariate polynomial.
- virtual `numeric max_coefficient` () const  
Implementation `ex::max_coefficient()`.
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
Add two indexed expressions.
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
Multiply an indexed expression with a scalar.
- virtual `bool contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
Try to contract two indexed expressions that appear in the same product.
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
Like `print()`, but dispatch to the specified class.
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
Like `print()`, but dispatch to the specified class.
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
Helper function for `subs()`.
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
Default interface of `nth` derivative `ex::diff(s, n)`.
- int `compare` (const `basic` &other) const  
Compare objects syntactically to establish canonical ordering.
- bool `is_equal` (const `basic` &other) const  
Test for syntactic equality.
- const `basic` & `hold` () const  
Stop further evaluation.
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
Set some `status_flags`.
- const `basic` & `clearflag` (unsigned f) const  
Clear some `status_flags`.

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an indexed object always returns 0.*
- `ex thiscontainer` (const `exvector` &v) const override
- `ex thiscontainer` (`exvector` &&v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `printindices` (const `print_context` &c, unsigned level) const
- void `print_indexed` (const `print_context` &c, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `validate` () const  
*Check whether all indices are of class `idx` and validate the symmetry tree.*

## Protected Member Functions inherited from `GiNaC::container< C >`

- `ex conjugate` () const override
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- virtual `ex thiscontainer` (const `STLT` &v) const  
*Similar to `duplicate()`, but with a preset sequence.*
- virtual `ex thiscontainer` (`STLT` &&v) const  
*Similar to `duplicate()`, but with a preset sequence (which gets pilfered).*
- virtual void `printseq` (const `print_context` &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- `STLT subschildren` (const `exmap` &m, unsigned `options`=0) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- ~[container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Protected Attributes

- [ex symtree](#)  
  Index symmetry (tree of symmetry objects)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
  of type [status\\_flags](#)
- unsigned [hashvalue](#)  
  hash value

## Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

## Friends

- [ex simplify\\_indexed](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
  Simplify indexed expression, return list of free indices.
- [ex simplify\\_indexed\\_product](#) (const [ex](#) &e, [exvector](#) &free\_indices, [exvector](#) &dummy\_indices, const [scalar\\_products](#) &sp)  
  Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.
- bool [reposition\\_dummy\\_indices](#) ([ex](#) &e, [exvector](#) &variant\_dummy\_indices, [exvector](#) &moved\_indices)  
  Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

## Additional Inherited Members

## Public Types inherited from [GiNaC::container< C >](#)

- typedef STLT::const\_iterator [const\\_iterator](#)
- typedef STLT::const\_reverse\_iterator [const\\_reverse\\_iterator](#)

## Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

## Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef [C< ex >](#) [STLT](#)

## Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

## Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, [size\\_t](#))

### 6.74.1 Detailed Description

This class holds an indexed expression.

It consists of a 'base' expression (the expression being indexed) which can be accessed as `op(0)`, and `n` (`n >= 0`) indices (all of class `idx`), accessible as `op(1)..op(n)`.

### 6.74.2 Constructor & Destructor Documentation

#### 6.74.2.1 `indexed()` [1/13]

```
GiNaC::indexed::indexed (
 const ex & b)
```

Construct indexed object with no index.

#### Parameters

|          |                 |
|----------|-----------------|
| <i>b</i> | Base expression |
|----------|-----------------|

References [GiNaC::not\\_symmetric\(\)](#), and [validate\(\)](#).

Referenced by [GiNaC::clifford::get\\_metric\(\)](#), [thiscontainer\(\)](#), and [thiscontainer\(\)](#).

**6.74.2.2 indexed()** [2/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const ex & i1)
```

Construct indexed object with one index.

The index must be of class `idx`.

**Parameters**

|           |                 |
|-----------|-----------------|
| <i>b</i>  | Base expression |
| <i>i1</i> | The index       |

References [validate\(\)](#).

**6.74.2.3 indexed()** [3/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const ex & i1,
 const ex & i2)
```

Construct indexed object with two indices.

The indices must be of class `idx`.

**Parameters**

|           |                 |
|-----------|-----------------|
| <i>b</i>  | Base expression |
| <i>i1</i> | First index     |
| <i>i2</i> | Second index    |

References [validate\(\)](#).

**6.74.2.4 indexed()** [4/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const ex & i1,
 const ex & i2,
 const ex & i3)
```

Construct indexed object with three indices.

The indices must be of class `idx`.

**Parameters**

|           |                 |
|-----------|-----------------|
| <i>b</i>  | Base expression |
| <i>i1</i> | First index     |
| <i>i2</i> | Second index    |
| <i>i3</i> | Third index     |

References [validate\(\)](#).

#### 6.74.2.5 indexed() [5/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const ex & i1,
 const ex & i2,
 const ex & i3,
 const ex & i4)
```

Construct indexed object with four indices.

The indices must be of class `idx`.

##### Parameters

|           |                 |
|-----------|-----------------|
| <i>b</i>  | Base expression |
| <i>i1</i> | First index     |
| <i>i2</i> | Second index    |
| <i>i3</i> | Third index     |
| <i>i4</i> | Fourth index    |

References [validate\(\)](#).

#### 6.74.2.6 indexed() [6/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const symmetry & symm,
 const ex & i1,
 const ex & i2)
```

Construct indexed object with two indices and a specified symmetry.

The indices must be of class `idx`.

##### Parameters

|             |                     |
|-------------|---------------------|
| <i>b</i>    | Base expression     |
| <i>symm</i> | Symmetry of indices |
| <i>i1</i>   | First index         |
| <i>i2</i>   | Second index        |

References [validate\(\)](#).

#### 6.74.2.7 indexed() [7/13]

```
GiNaC::indexed::indexed (
 const ex & b,
```

```

const symmetry & symm,
const ex & i1,
const ex & i2,
const ex & i3)

```

Construct indexed object with three indices and a specified symmetry.

The indices must be of class `idx`.

#### Parameters

|             |                     |
|-------------|---------------------|
| <i>b</i>    | Base expression     |
| <i>symm</i> | Symmetry of indices |
| <i>i1</i>   | First index         |
| <i>i2</i>   | Second index        |
| <i>i3</i>   | Third index         |

References [validate\(\)](#).

#### 6.74.2.8 indexed() [8/13]

```

GiNaC::indexed::indexed (
 const ex & b,
 const symmetry & symm,
 const ex & i1,
 const ex & i2,
 const ex & i3,
 const ex & i4)

```

Construct indexed object with four indices and a specified symmetry.

The indices must be of class `idx`.

#### Parameters

|             |                     |
|-------------|---------------------|
| <i>b</i>    | Base expression     |
| <i>symm</i> | Symmetry of indices |
| <i>i1</i>   | First index         |
| <i>i2</i>   | Second index        |
| <i>i3</i>   | Third index         |
| <i>i4</i>   | Fourth index        |

References [validate\(\)](#).

#### 6.74.2.9 indexed() [9/13]

```

GiNaC::indexed::indexed (
 const ex & b,
 const exvector & iv)

```

Construct indexed object with a specified vector of indices.

The indices must be of class `idx`.

## Parameters

|           |                   |
|-----------|-------------------|
| <i>b</i>  | Base expression   |
| <i>iv</i> | Vector of indices |

References [GiNaC::container\\_storage< C >::seq](#), and [validate\(\)](#).

**6.74.2.10 indexed()** [10/13]

```
GiNaC::indexed::indexed (
 const ex & b,
 const symmetry & symm,
 const exvector & iv)
```

Construct indexed object with a specified vector of indices and symmetry.

The indices must be of class idx.

## Parameters

|             |                     |
|-------------|---------------------|
| <i>b</i>    | Base expression     |
| <i>symm</i> | Symmetry of indices |
| <i>iv</i>   | Vector of indices   |

References [GiNaC::container\\_storage< C >::seq](#), and [validate\(\)](#).

**6.74.2.11 indexed()** [11/13]

```
GiNaC::indexed::indexed (
 const symmetry & symm,
 const exprseq & es)
```

**6.74.2.12 indexed()** [12/13]

```
GiNaC::indexed::indexed (
 const symmetry & symm,
 const exvector & v)
```

**6.74.2.13 indexed()** [13/13]

```
GiNaC::indexed::indexed (
 const symmetry & symm,
 exvector && v)
```

### 6.74.3 Member Function Documentation

#### 6.74.3.1 precedence()

```
unsigned GiNaC::indexed::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [print\\_indexed\(\)](#).

#### 6.74.3.2 info()

```
bool GiNaC::indexed::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::indexed](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

#### 6.74.3.3 eval()

```
ex GiNaC::indexed::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::canonicalize\(\)](#), [GiNaC::basic::ex](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [thiscontainer\(\)](#).

#### 6.74.3.4 real\_part()

```
ex GiNaC::indexed::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [info\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 6.74.3.5 `imag_part()`

```
ex GiNaC::indexed::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [info\(\)](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::info\\_flags::real](#).

### 6.74.3.6 `get_free_indices()`

```
exvector GiNaC::indexed::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [get\\_dummy\\_indices\(\)](#).

### 6.74.3.7 `archive()`

```
void GiNaC::indexed::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) indexed object into archive.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), and [symtree](#).

### 6.74.3.8 `read_archive()`

```
void GiNaC::indexed::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) indexed object from archive.

Reimplemented from [GiNaC::container< C >](#).

References [n](#), [GiNaC::not\\_symmetric\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_symm\(\)](#), [GiNaC::symm\(\)](#), [symtree](#), and [validate\(\)](#).

**6.74.3.9 derivative()**

```
ex GiNaC::indexed::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for an indexed object always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#).

**6.74.3.10 thiscontainer() [1/2]**

```
ex GiNaC::indexed::thiscontainer (
 const exvector & v) const [override], [protected]
```

References [indexed\(\)](#), and [symtree](#).

Referenced by [eval\(\)](#), and [expand\(\)](#).

**6.74.3.11 thiscontainer() [2/2]**

```
ex GiNaC::indexed::thiscontainer (
 exvector && v) const [override], [protected]
```

References [indexed\(\)](#), and [symtree](#).

**6.74.3.12 return\_type()**

```
unsigned GiNaC::indexed::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

**6.74.3.13 return\_type\_tinfo()**

```
return_type_t GiNaC::indexed::return_type_tinfo () const [inline], [override], [protected],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::op\(\)](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

**6.74.3.14 expand()**

```
ex GiNaC::indexed::expand (
 unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_indexed](#), [GINAC\\_ASSERT](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::container\\_storage< C >::seq](#), and [thiscontainer\(\)](#).

**6.74.3.15 all\_index\_values\_are()**

```
bool GiNaC::indexed::all_index_values_are (
 unsigned inf) const
```

Check whether all index values have a certain property.

See also

class [info\\_flags](#)

References [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.16 get\_indices()**

```
exvector GiNaC::indexed::get_indices () const
```

Return a vector containing the object's indices.

References [GINAC\\_ASSERT](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.17 get\_dummy\_indices() [1/2]**

```
exvector GiNaC::indexed::get_dummy_indices () const
```

Return a vector containing the dummy indices of the object, if any.

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.18 get\_dummy\_indices() [2/2]**

```
exvector GiNaC::indexed::get_dummy_indices (
 const indexed & other) const
```

Return a vector containing the dummy indices in the contraction with another indexed object.

This is symmetric: `a.get_dummy_indices(b) == b.get_dummy_indices(a)`

References [GiNaC::find\\_dummy\\_indices\(\)](#), and [get\\_free\\_indices\(\)](#).

**6.74.3.19 has\_dummy\_index\_for()**

```
bool GiNaC::indexed::has_dummy_index_for (
 const ex & i) const
```

Check whether the object has an index that forms a dummy index pair with a given index.

References [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.74.3.20 get\_symmetry()**

```
ex GiNaC::indexed::get_symmetry () const [inline]
```

Return symmetry properties.

References [symtree](#).

Referenced by [GiNaC::clifford::get\\_metric\(\)](#).

**6.74.3.21 printindices()**

```
void GiNaC::indexed::printindices (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), and [print\\_indexed\(\)](#).

**6.74.3.22 print\_indexed()**

```
void GiNaC::indexed::print_indexed (
 const print_context & c,
 const char * openbrace,
 const char * closebrace,
 unsigned level) const [protected]
```

References [c](#), [precedence\(\)](#), [printindices\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**6.74.3.23 do\_print()**

```
void GiNaC::indexed::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

### 6.74.3.24 do\_print\_latex()

```
void GiNaC::indexed::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_indexed\(\)](#).

### 6.74.3.25 do\_print\_tree()

```
void GiNaC::indexed::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), [printindices\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [symtree](#).

### 6.74.3.26 validate()

```
void GiNaC::indexed::validate () const [protected]
```

Check whether all indices are of class `idx` and validate the symmetry tree.

This function is used internally to make sure that all constructed indexed objects really carry indices and not some other classes.

References [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::container\\_storage< C >::seq](#), [symtree](#), and [validate\(\)](#).

Referenced by [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [indexed\(\)](#), [read\\_archive\(\)](#), and [validate\(\)](#).

## 6.74.4 Friends And Related Symbol Documentation

### 6.74.4.1 simplify\_indexed

```
ex simplify_indexed (
 const ex & e,
 exvector & free_indices,
 exvector & dummy_indices,
 const scalar_products & sp) [friend]
```

Simplify indexed expression, return list of free indices.

Referenced by [GiNaC::clifford::get\\_metric\(\)](#), and [GiNaC::clifford::same\\_metric\(\)](#).

### 6.74.4.2 simplify\_indexed\_product

```
ex simplify_indexed_product (
 const ex & e,
 exvector & free_indices,
 exvector & dummy_indices,
 const scalar_products & sp) [friend]
```

Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free indices.

### 6.74.4.3 reposition\_dummy\_indices

```
bool reposition_dummy_indices (
 ex & e,
 exvector & variant_dummy_indices,
 exvector & moved_indices) [friend]
```

Raise/lower dummy indices in a single indexed objects to canonicalize their variance.

#### Parameters

|                              |                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------|
| <i>e</i>                     | Object to work on                                                                   |
| <i>variant_dummy_indices</i> | The set of indices that might need repositioning (will be changed by this function) |
| <i>moved_indices</i>         | The set of indices that have been repositioned (will be changed by this function)   |

#### Returns

true if 'e' was changed

## 6.74.5 Member Data Documentation

### 6.74.5.1 symtree

```
ex GiNaC::indexed::symtree [protected]
```

Index symmetry (tree of symmetry objects)

Referenced by [archive\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [do\\_print\\_tree\(\)](#), [eval\(\)](#), [get\\_symmetry\(\)](#), [read\\_archive\(\)](#), [thiscontainer\(\)](#), [thiscontainer\(\)](#), and [validate\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.75 GiNaC::info\_flags Class Reference

Possible attributes an object can have.

```
#include <flags.h>
```

#### Public Types

- enum {  
[numeric](#) , [real](#) , [rational](#) , [integer](#) ,  
[crational](#) , [cinteger](#) , [positive](#) , [negative](#) ,  
[nonnegative](#) , [posint](#) , [negint](#) , [nonnegint](#) ,  
[even](#) , [odd](#) , [prime](#) , [relation](#) ,  
[relation\\_equal](#) , [relation\\_not\\_equal](#) , [relation\\_less](#) , [relation\\_less\\_or\\_equal](#) ,  
[relation\\_greater](#) , [relation\\_greater\\_or\\_equal](#) , [symbol](#) , [list](#) ,  
[exprseq](#) , [polynomial](#) , [integer\\_polynomial](#) , [cinteger\\_polynomial](#) ,  
[rational\\_polynomial](#) , [crational\\_polynomial](#) , [rational\\_function](#) , [indexed](#) ,  
[has\\_indices](#) , [idx](#) , [expanded](#) , [indefinite](#) }

### 6.75.1 Detailed Description

Possible attributes an object can have.

### 6.75.2 Member Enumeration Documentation

#### 6.75.2.1 anonymous enum

anonymous enum

##### Enumerator

|                           |  |
|---------------------------|--|
| numeric                   |  |
| real                      |  |
| rational                  |  |
| integer                   |  |
| crational                 |  |
| cinteger                  |  |
| positive                  |  |
| negative                  |  |
| nonnegative               |  |
| posint                    |  |
| negint                    |  |
| nonnegint                 |  |
| even                      |  |
| odd                       |  |
| prime                     |  |
| relation                  |  |
| relation_equal            |  |
| relation_not_equal        |  |
| relation_less             |  |
| relation_less_or_equal    |  |
| relation_greater          |  |
| relation_greater_or_equal |  |
| symbol                    |  |
| list                      |  |
| exprseq                   |  |
| polynomial                |  |
| integer_polynomial        |  |
| cinteger_polynomial       |  |
| rational_polynomial       |  |
| crational_polynomial      |  |
| rational_function         |  |
| indexed                   |  |
| has_indices               |  |
| idx                       |  |
| expanded                  |  |
| indefinite                |  |

The documentation for this class was generated from the following file:

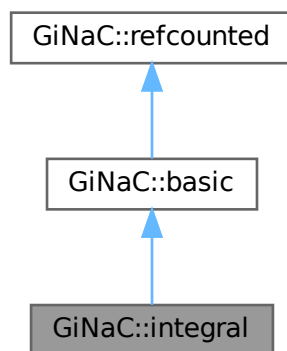
- [flags.h](#)

## 6.76 GiNaC::integral Class Reference

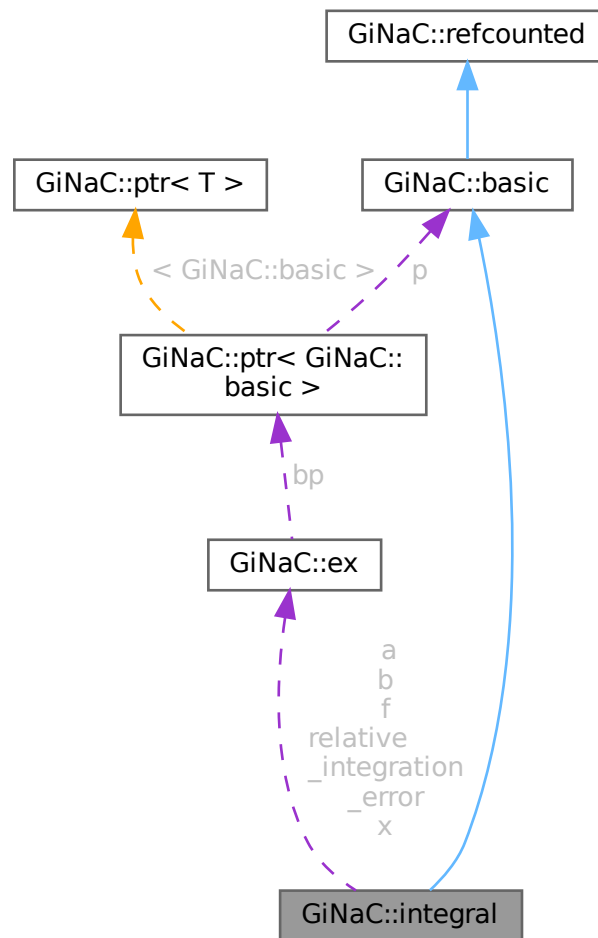
Symbolic integral.

```
#include <integral.h>
```

Inheritance diagram for GiNaC::integral:



Collaboration diagram for GiNaC::integral:



## Public Member Functions

- `integral` (const `ex` &x\_, const `ex` &a\_, const `ex` &b\_, const `ex` &f\_)  
Return relative operator precedence (for parenthesizing output).
- unsigned `precedence` () const override  
Perform automatic non-interruptive term rewriting rules.
- `ex eval` () const override  
Evaluate object numerically.
- int `degree` (const `ex` &s) const override  
Return degree of highest power in object s.
- int `ldegree` (const `ex` &s) const override  
Return degree of lowest power in object s.
- `ex eval_ncmul` (const `exvector` &v) const override
- size\_t `nops` () const override

- *Number of operands/members.*
- `ex op (size_t i)` const override  
*Return operand/member at position i.*
- `ex & let_op (size_t i)` override  
*Return modifiable operand/member at position i.*
- `ex expand (unsigned options=0)` const override  
*Expand expression, i.e.*
- `exvector get_free_indices ()` const override  
*Return a vector containing the free indices of an expression.*
- unsigned `return_type ()` const override
- `return_type_t return_type_tinfo ()` const override
- `ex conjugate ()` const override
- `ex eval_integ ()` const override  
*Evaluate integrals, if result is known.*
- void `archive (archive_node &n)` const override  
*Save (a.k.a.*
- void `read_archive (const archive_node &n, lst &syms)` override  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- const `basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate ()` const  
*Create a clone of this object on the heap.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const

- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).
- virtual void `accept` (`GiNaC::visitor` &`v`) const
- virtual bool `is_polynomial` (const `ex` &`var`) const
  - Check whether this is a polynomial in the given variables.
- virtual `ex` `coeff` (const `ex` &`s`, int `n`=1) const
  - Return coefficient of degree `n` in object `s`.
- virtual `ex` `collect` (const `ex` &`s`, bool `distributed`=false) const
  - Sort expanded expression in terms of powers of some object(s).
- virtual `ex` `normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const
  - Default implementation of `ex::normal()`.
- virtual `ex` `to_rational` (`exmap` &`repl`) const
  - Default implementation of `ex::to_rational()`.
- virtual `ex` `to_polynomial` (`exmap` &`repl`) const
- virtual `numeric` `integer_content` () const
- virtual `ex` `smod` (const `numeric` &`xi`) const
  - Apply symmetric modular homomorphism to an expanded multivariate polynomial.
- virtual `numeric` `max_coefficient` () const
  - Implementation `ex::max_coefficient()`.
- virtual `ex` `add_indexed` (const `ex` &`self`, const `ex` &`other`) const
  - Add two indexed expressions.
- virtual `ex` `scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const
  - Multiply an indexed expression with a scalar.
- virtual bool `contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const
  - Try to contract two indexed expressions that appear in the same product.
- virtual `ex` `real_part` () const
- virtual `ex` `imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &`c`, unsigned level) const
    - Like `print()`, but dispatch to the specified class.
- void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned level) const
  - Like `print()`, but dispatch to the specified class.
- `ex` `subs_one_level` (const `exmap` &`m`, unsigned `options`) const
  - Helper function for `subs()`.
- `ex` `diff` (const `symbol` &`s`, unsigned `nth`=1) const
  - Default interface of `nth` derivative `ex::diff(s, n)`.
- int `compare` (const `basic` &`other`) const
  - Compare objects syntactically to establish canonical ordering.
- bool `is_equal` (const `basic` &`other`) const
  - Test for syntactic equality.
- const `basic` & `hold` () const
  - Stop further evaluation.
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const
  - Set some `status_flags`.
- const `basic` & `clearflag` (unsigned `f`) const
  - Clear some `status_flags`.

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

### Static Public Attributes

- static int `max_integration_level` = 15
- static `ex` `relative_integration_error` = 1e-8

### Protected Member Functions

- `ex` `derivative` (const `symbol` &s) const override  
*Default implementation of `ex::diff()`.*
- `ex` `series` (const `relational` &r, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Private Attributes

- `ex` x
- `ex` a
- `ex` b
- `ex` f

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.76.1 Detailed Description

Symbolic integral.

### 6.76.2 Constructor & Destructor Documentation

#### 6.76.2.1 `integral()`

```
GiNaC::integral::integral (
 const ex & x_,
 const ex & a_,
 const ex & b_,
 const ex & f_)
```

References [x](#).

Referenced by [derivative\(\)](#), [expand\(\)](#), and [series\(\)](#).

### 6.76.3 Member Function Documentation

#### 6.76.3.1 `precedence()`

```
unsigned GiNaC::integral::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\\_latex\(\)](#).

#### 6.76.3.2 `eval()`

```
ex GiNaC::integral::eval () const [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [a](#), [b](#), [GiNaC::status\\_flags::evaluated](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::basic::hold\(\)](#), and [x](#).

#### 6.76.3.3 `evalf()`

```
ex GiNaC::integral::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::adaptivesimpson\(\)](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::ex::evalf\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

#### 6.76.3.4 degree()

```
int GiNaC::integral::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::degree\(\)](#), and [f](#).

#### 6.76.3.5 ldegree()

```
int GiNaC::integral::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), and [f](#).

#### 6.76.3.6 eval\_ncmul()

```
ex GiNaC::integral::eval_ncmul (
 const exvector & v) const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#), and [f](#).

#### 6.76.3.7 nops()

```
size_t GiNaC::integral::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.76.3.8 op()

```
ex GiNaC::integral::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [GINAC\\_ASSERT](#), and [x](#).

### 6.76.3.9 let\_op()

```
ex & GiNaC::integral::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [f](#), and [x](#).

### 6.76.3.10 expand()

```
ex GiNaC::integral::expand (
 unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::basic::ex](#), [GiNaC::ex::expand\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [integral\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [options](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

Referenced by [eval\\_integ\(\)](#), and [expand\(\)](#).

### 6.76.3.11 get\_free\_indices()

```
exvector GiNaC::integral::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), and [GiNaC::ex::get\\_free\\_indices\(\)](#).

### 6.76.3.12 return\_type()

```
unsigned GiNaC::integral::return_type () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#), and [GiNaC::ex::return\\_type\(\)](#).

### 6.76.3.13 return\_type\_tinfo()

```
return_type_t GiNaC::integral::return_type_tinfo () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [f](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

### 6.76.3.14 conjugate()

```
ex GiNaC::integral::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [a](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [b](#), [GiNaC::ex::conjugate\(\)](#), [f](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.15 eval\_integ()

```
ex GiNaC::integral::eval_integ () const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::eval\\_integ\(\)](#), [expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [f](#), [GiNaC::basic::flags](#), [GiNaC::ex::has\(\)](#), [GiNaC::log\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.16 archive()

```
void GiNaC::integral::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).

### 6.76.3.17 read\_archive()

```
void GiNaC::integral::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [f](#), [n](#), and [x](#).

### 6.76.3.18 derivative()

```
ex GiNaC::integral::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [a](#), [b](#), [GiNaC::ex::diff\(\)](#), [f](#), [integral\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.76.3.19 series()

```
ex GiNaC::integral::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [protected], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [a](#), [b](#), [f](#), [integral\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::nops\(\)](#), [options](#), [order](#), [r](#), [GiNaC::ex::series\(\)](#), [series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

Referenced by [series\(\)](#).

### 6.76.3.20 do\_print()

```
void GiNaC::integral::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [GiNaC::ex::print\(\)](#), and [x](#).

### 6.76.3.21 do\_print\_latex()

```
void GiNaC::integral::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [a](#), [b](#), [c](#), [f](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [x](#).

## 6.76.4 Member Data Documentation

### 6.76.4.1 max\_integration\_level

```
int GiNaC::integral::max_integration_level = 15 [static]
```

Referenced by [GiNaC::adaptivesimpson\(\)](#).

### 6.76.4.2 relative\_integration\_error

```
ex GiNaC::integral::relative_integration_error = 1e-8 [static]
```

### 6.76.4.3 x

```
ex GiNaC::integral::x [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [integral\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

### 6.76.4.4 a

```
ex GiNaC::integral::a [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

### 6.76.4.5 b

```
ex GiNaC::integral::b [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), and [series\(\)](#).

### 6.76.4.6 f

```
ex GiNaC::integral::f [private]
```

Referenced by [archive\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [eval\\_integ\(\)](#), [eval\\_ncmul\(\)](#), [evalf\(\)](#), [expand\(\)](#), [get\\_free\\_indices\(\)](#), [ldegree\(\)](#), [let\\_op\(\)](#), [op\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [series\(\)](#).

The documentation for this class was generated from the following files:

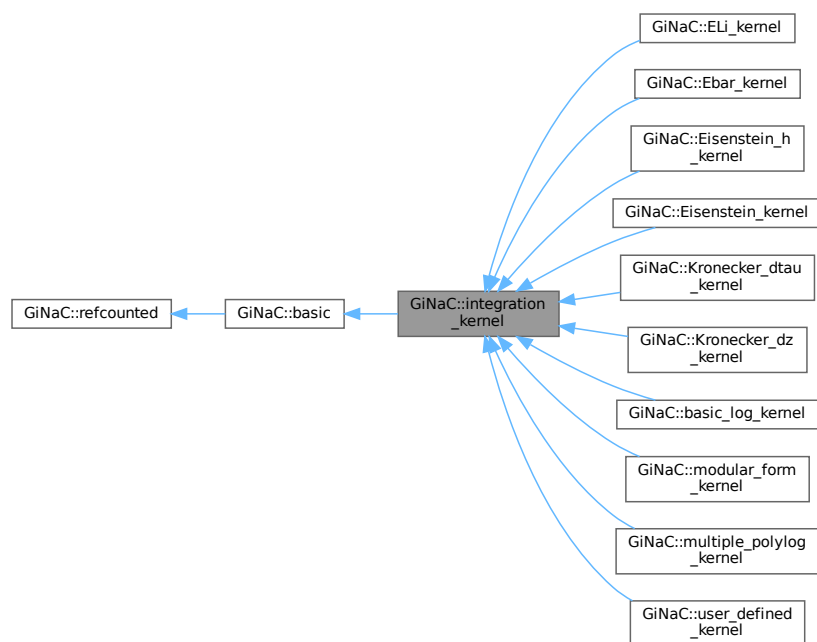
- [integral.h](#)
- [indexed.cpp](#)
- [integral.cpp](#)
- [pseries.cpp](#)

## 6.77 GiNaC::integration\_kernel Class Reference

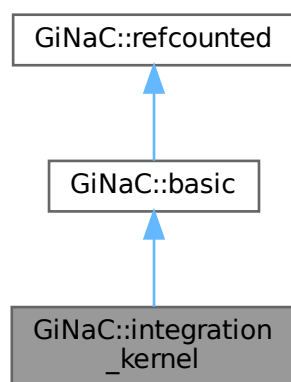
The base class for integration kernels for iterated integrals.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::integration\_kernel:



Collaboration diagram for GiNaC::integration\_kernel:



## Public Member Functions

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual bool **is\_numeric** (void) const  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual **ex get\_numerical\_value** (const ex &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

Public Member Functions inherited from **GiNaC::basic**

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*

- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v)` const
- virtual `bool is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl)` const
- virtual `numeric integer_content ()` const
- virtual `ex smod (const numeric &xi)` const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient ()` const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type ()` const
- virtual `return_type_t return_type_tinfo ()` const

- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- virtual `cln::cl_N series_coeff_impl` (int i) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic ()`
- virtual `ex eval_ncmul (const exvector &v) const`
- virtual `bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative (const symbol &s) const`  
*Default implementation of `ex::diff()`.*
- virtual `int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- virtual `bool is_equal_same_type (const basic &other) const`  
*Returns true if two objects of same type are equal.*
- virtual `unsigned calchash () const`  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable () const`  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print (const print_context &c, unsigned level) const`  
*Default output to stream.*
- void `do_print_tree (const print_tree &c, unsigned level) const`  
*Tree output to stream.*
- void `do_print_python_repr (const print_python_repr &c, unsigned level) const`  
*Python parsable output to stream.*

## Protected Attributes

- `int cache_step_size`
- `std::vector< cln::cl_N > series_vec`

## Protected Attributes inherited from `GiNaC::basic`

- `unsigned flags`  
*of type `status_flags`*
- `unsigned hashvalue`  
*hash value*

### 6.77.1 Detailed Description

The base class for integration kernels for iterated integrals.

This class represents the differential one-form

$$\omega = d\lambda$$

The integration variable is a dummy variable and does not need to be specified.

## 6.77.2 Member Function Documentation

### 6.77.2.1 series()

```
ex GiNaC::integration_kernel::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

Reimplemented in [GiNaC::modular\\_form\\_kernel](#).

References [order](#), and [r](#).

### 6.77.2.2 has\_trailing\_zero()

```
bool GiNaC::integration_kernel::has_trailing_zero (
 void) const [virtual]
```

This routine returns true, if the integration kernel has a trailing zero.

### 6.77.2.3 is\_numeric()

```
bool GiNaC::integration_kernel::is_numeric (
 void) const [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented in [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Kronecker\\_dz\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

### 6.77.2.4 Laurent\_series()

```
ex GiNaC::integration_kernel::Laurent_series (
 const ex & x,
 int order) const [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented in [GiNaC::modular\\_form\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

References [n](#), [order](#), [GiNaC::pow\(\)](#), [GiNaC::ex::series\(\)](#), and [x](#).

### 6.77.2.5 `get_numerical_value()`

```
ex GiNaC::integration_kernel::get_numerical_value (
 const ex & lambda,
 int N_trunc = 0) const [virtual]
```

Evaluates the integrand at lambda.

Reimplemented in [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::Kronecker\\_dz\\_kernel](#).

### 6.77.2.6 `uses_Laurent_series()`

```
bool GiNaC::integration_kernel::uses_Laurent_series () const [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented in [GiNaC::Eisenstein\\_kernel](#), [GiNaC::Eisenstein\\_h\\_kernel](#), [GiNaC::modular\\_form\\_kernel](#), and [GiNaC::user\\_defined\\_kernel](#).

### 6.77.2.7 `series_coeff_impl()`

```
cln::cl_N GiNaC::integration_kernel::series_coeff_impl (
 int i) const [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented in [GiNaC::basic\\_log\\_kernel](#), [GiNaC::multiple\\_polylog\\_kernel](#), [GiNaC::ELi\\_kernel](#), [GiNaC::Ebar\\_kernel](#), [GiNaC::Kronecker\\_dtau\\_kernel](#), and [GiNaC::Kronecker\\_dz\\_kernel](#).

### 6.77.2.8 `get_cache_size()`

```
size_t GiNaC::integration_kernel::get_cache_size (
 void) const
```

Returns the current size of the cache.

### 6.77.2.9 `set_cache_step()`

```
void GiNaC::integration_kernel::set_cache_step (
 int cache_steps) const
```

Sets the step size by which the cache is increased.

### 6.77.2.10 get\_series\_coeff()

```
ex GiNaC::integration_kernel::get_series_coeff (
 int i) const
```

Wrapper around `series_coeff(i)`, converts `cl_N` to numeric.

### 6.77.2.11 series\_coeff()

```
cln::cl_N GiNaC::integration_kernel::series_coeff (
 int i) const
```

Subclasses have either to implement `series_coeff_impl` or the two methods `Laurent_series` and `uses_Laurent_series`.

The method `series_coeff_impl` can be used, if a single coefficient can be computed independently of the others.

The method `Laurent_series` can be used, if it is more efficient to compute a Laurent series in one shot and to determine a range of coefficients from this Laurent series.

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalf\(\)](#), and [x](#).

### 6.77.2.12 get\_numerical\_value\_impl()

```
ex GiNaC::integration_kernel::get_numerical_value_impl (
 const ex & lambda,
 const ex & pre,
 int shift,
 int N_trunc) const [protected]
```

The actual implementation for computing a numerical value for the integrand.

References [GiNaC::Digits](#), [GiNaC::ex::evalf\(\)](#), and [one](#).

Referenced by [GiNaC::ELi\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::get\\_numerical\\_value\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::get\\_numerical\\_value\(\)](#).

### 6.77.2.13 do\_print()

```
void GiNaC::integration_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#).

## 6.77.3 Member Data Documentation

### 6.77.3.1 cache\_step\_size

```
int GiNaC::integration_kernel::cache_step_size [mutable], [protected]
```

### 6.77.3.2 `series_vec`

```
std::vector<cln::cl_N> GiNaC::integration_kernel::series_vec [mutable], [protected]
```

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.78 `GiNaC::is_not_a_clifford` Struct Reference

Predicate for finding non-clifford objects.

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

### 6.78.1 Detailed Description

Predicate for finding non-clifford objects.

### 6.78.2 Member Function Documentation

#### 6.78.2.1 `operator()`

```
bool GiNaC::is_not_a_clifford::operator() (
 const ex & e) [inline]
```

The documentation for this struct was generated from the following file:

- [clifford.cpp](#)

## 6.79 `GiNaC::is_summation_idx` Struct Reference

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &e)

## 6.79.1 Member Function Documentation

### 6.79.1.1 operator()

```
bool GiNaC::is_summation_idx::operator() (
 const ex & e) [inline]
```

References [GiNaC::is\\_dummy\\_pair\(\)](#).

The documentation for this struct was generated from the following file:

- [indexed.cpp](#)

## 6.80 GiNaC::iterated\_integral2\_SERIAL Class Reference

Complete elliptic integral of the first kind.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.80.1 Detailed Description

Complete elliptic integral of the first kind.

Complete elliptic integral of the second kind. Iterated integral.

### 6.80.2 Member Data Documentation

#### 6.80.2.1 serial

```
unsigned GiNaC::iterated_integral2_SERIAL::serial [static]
```

#### Initial value:

```
=
 function::register_new(function_options("iterated_integral", 2).
 eval_func(iterated_integral2_eval).
 evalf_func(iterated_integral2_evalf).
 do_not_evalf_params().
 overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.81 GiNaC::iterated\_integral3\_SERIAL Class Reference

Iterated integral with explicit truncation.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.81.1 Detailed Description

Iterated integral with explicit truncation.

### 6.81.2 Member Data Documentation

#### 6.81.2.1 serial

```
unsigned GiNaC::iterated_integral3_SERIAL::serial [static]
```

**Initial value:**

```
=
 function::register_new(function_options("iterated_integral", 3).
 eval_func(iterated_integral3_eval).
 evalf_func(iterated_integral3_evalf).
 do_not_evalf_params().
 overloaded(2))
```

Referenced by [GiNaC::iterated\\_integral\(\)](#).

The documentation for this class was generated from the following files:

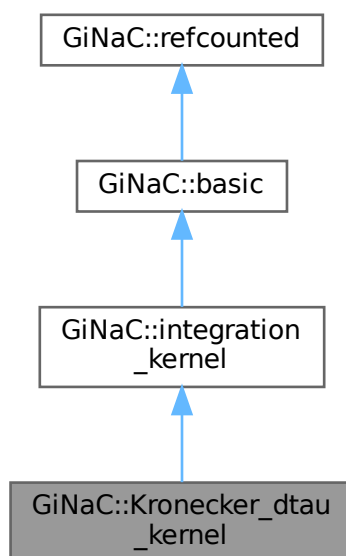
- [inifcns.h](#)
- [inifcns\\_elliptic.cpp](#)

## 6.82 GiNaC::Kronecker\_dtau\_kernel Class Reference

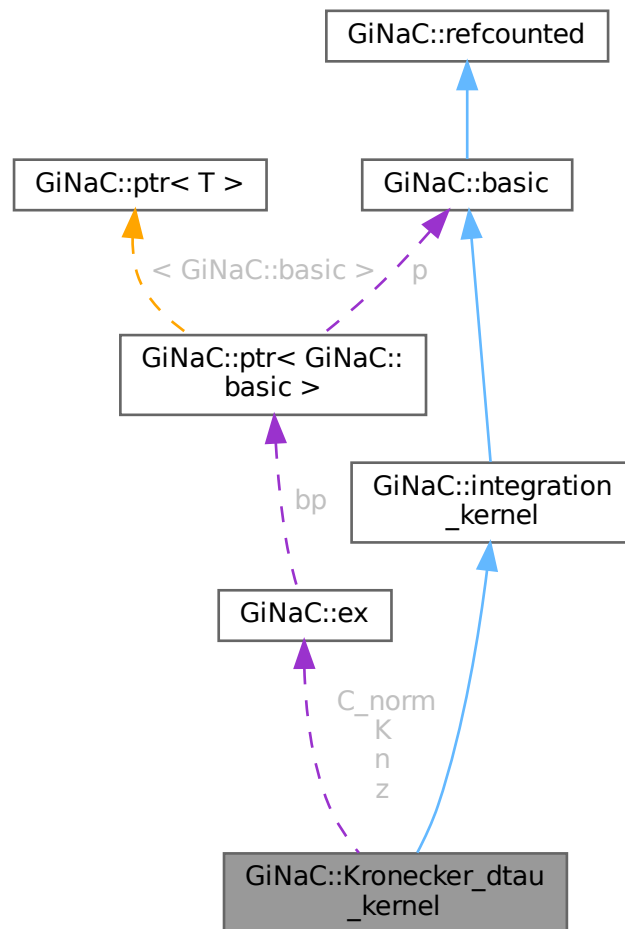
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dtau\_kernel:



Collaboration diagram for GiNaC::Kronecker\_dtau\_kernel:



## Public Member Functions

- `Kronecker_dtau_kernel` (const `ex` &`n`, const `ex` &`z`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex &let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{\wedge}(n)(z, K*\tau)$ , where  $\tau$  is given by `qbar`.*

## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int `i`) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int `shift`, int `N_trunc`) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex z](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.82.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},\tau}(z_j) = \frac{C_n K(n-1)}{(2\pi i)^n} g^{(n)}(z_j, K\tau) \frac{d\bar{q}}{\bar{q}}$$

## 6.82.2 Constructor & Destructor Documentation

### 6.82.2.1 Kronecker\_dtau\_kernel()

```
GiNaC::Kronecker_dtau_kernel::Kronecker_dtau_kernel (
 const ex & n,
 const ex & z,
 const ex & K = numeric(1),
 const ex & C_norm = numeric(1))
```

## 6.82.3 Member Function Documentation

### 6.82.3.1 nops()

```
size_t GiNaC::Kronecker_dtau_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.82.3.2 op()

```
ex GiNaC::Kronecker_dtau_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), and [z](#).

### 6.82.3.3 let\_op()

```
ex & GiNaC::Kronecker_dtau_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), and [z](#).

### 6.82.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dtau_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), and [z](#).

### 6.82.3.5 `get_numerical_value()`

```
ex GiNaC::Kronecker_dtau_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of the  $g^{(n)}(z, K \cdot \tau)$ , where  $\tau$  is given by  $qbar$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#),  $K$ ,  $n$ , [GiNaC::Pi](#), [GiNaC::pow\(\)](#),  $qbar$ , and  $z$ .

### 6.82.3.6 `series_coeff_impl()`

```
cln::cl_N GiNaC::Kronecker_dtau_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::l](#),  $K$ ,  $n$ , [GiNaC::Pi](#), [GiNaC::numeric::to\\_cl\\_N\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and  $z$ .

### 6.82.3.7 `do_print()`

```
void GiNaC::Kronecker_dtau_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [C\\_norm](#),  $K$ ,  $n$ , [GiNaC::ex::print\(\)](#), and  $z$ .

## 6.82.4 Member Data Documentation

### 6.82.4.1 `n`

```
ex GiNaC::Kronecker_dtau_kernel::n [protected]
```

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.82.4.2 `z`

```
ex GiNaC::Kronecker_dtau_kernel::z [protected]
```

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 6.82.4.3 K

`ex GiNaC::Kronecker_dtau_kernel::K [protected]`

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

#### 6.82.4.4 C\_norm

`ex GiNaC::Kronecker_dtau_kernel::C_norm [protected]`

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

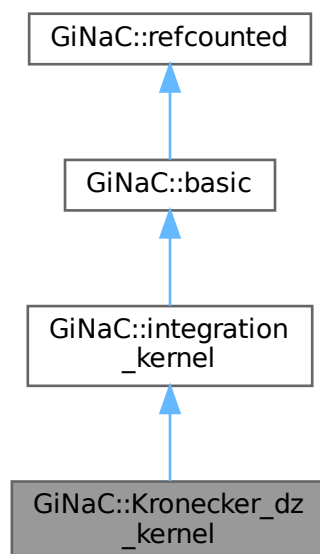
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.83 GiNaC::Kronecker\_dz\_kernel Class Reference

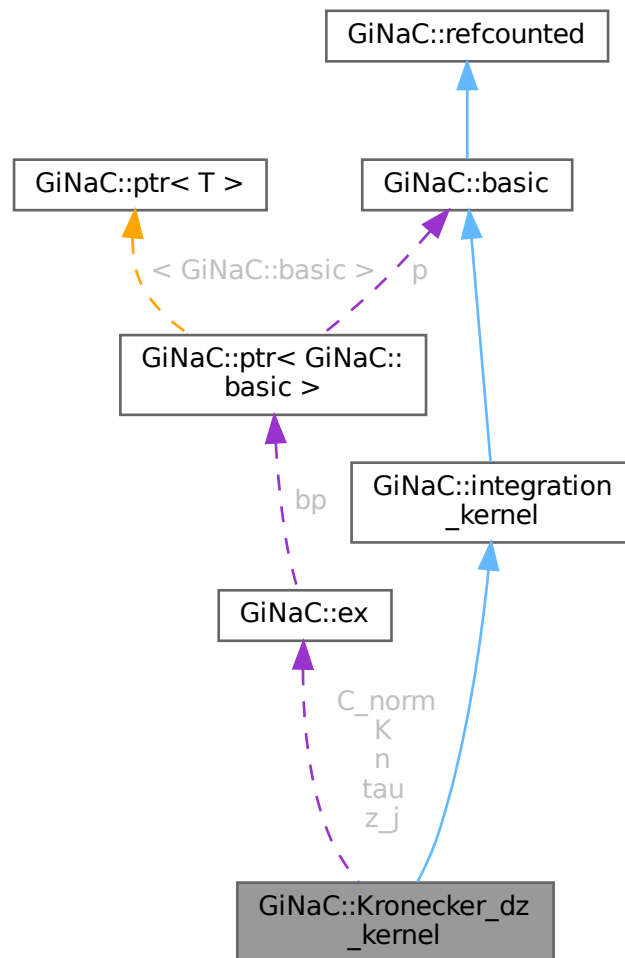
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::Kronecker\_dz\_kernel:



Collaboration diagram for GiNaC::Kronecker\_dz\_kernel:



## Public Member Functions

- `Kronecker_dz_kernel` (const `ex` &`n`, const `ex` &`z_j`, const `ex` &`tau`, const `ex` &`K=numeric(1)`, const `ex` &`C_norm=numeric(1)`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex get_numerical_value` (const `ex` &`z`, int `N_trunc=0`) const override  
*Returns the value of the  $g^{(n-1)}(z-z_j, K*\tau)$ .*

## Public Member Functions inherited from GiNaC::integration\_kernel

- **ex series** (const relational &r, int order, unsigned options=0) const override  
*Default implementation of ex::series().*
- virtual bool **has\_trailing\_zero** (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual **ex Laurent\_series** (const ex &x, int order) const  
*Returns the Laurent series, starting possibly with the pole term.*
- size\_t **get\_cache\_size** (void) const  
*Returns the current size of the cache.*
- void **set\_cache\_step** (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- **ex get\_series\_coeff** (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N **series\_coeff** (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from GiNaC::basic

- virtual **~basic** ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- **basic** (const basic &other)
- const **basic & operator=** (const basic &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual **basic \* duplicate** () const  
*Create a clone of this object on the heap.*
- virtual **ex eval** () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual **ex evalf** () const  
*Evaluate object numerically.*
- virtual **ex evalm** () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual **ex eval\_integ** () const  
*Evaluate integrals, if result is known.*
- virtual **ex eval\_indexed** (const basic &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void **print** (const print\_context &c, unsigned level=0) const  
*Output to stream.*
- virtual void **dbgprint** () const  
*Little wrapper around print to be called within a debugger.*
- virtual void **dbgprinttree** () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned **precedence** () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool **info** (unsigned inf) const  
*Information about the object.*
- virtual **ex operator[]** (const ex &index) const
- virtual **ex operator[]** (size\_t i) const
- virtual **ex & operator[]** (const ex &index)
- virtual **ex & operator[]** (size\_t i)
- virtual bool **has** (const ex &other, unsigned options=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int `i`) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual bool uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int `shift`, int `N_trunc`) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex n](#)
- [ex z\\_j](#)
- [ex tau](#)
- [ex K](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- `std::vector< cln::cl\_N >` [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.83.1 Detailed Description

The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .

This class represents the differential one-form

$$\omega_{n,K}^{\text{Kronecker},z}(z_j, \tau) = C_n (2\pi i)^{2-n} g^{(n-1)}(z - z_j, K\tau) dz$$

## 6.83.2 Constructor & Destructor Documentation

### 6.83.2.1 Kronecker\_dz\_kernel()

```
GiNaC::Kronecker_dz_kernel::Kronecker_dz_kernel (
 const ex & n,
 const ex & z_j,
 const ex & tau,
 const ex & K = numeric(1),
 const ex & C_norm = numeric(1))
```

## 6.83.3 Member Function Documentation

### 6.83.3.1 nops()

```
size_t GiNaC::Kronecker_dz_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.83.3.2 op()

```
ex GiNaC::Kronecker_dz_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [K](#), [n](#), [tau](#), and [z\\_j](#).

### 6.83.3.3 let\_op()

```
ex & GiNaC::Kronecker_dz_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [K](#), [n](#), [tau](#), and [z\\_j](#).

### 6.83.3.4 is\_numeric()

```
bool GiNaC::Kronecker_dz_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [K](#), [n](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::posint](#), [tau](#), and [z\\_j](#).

### 6.83.3.5 `get_numerical_value()`

```
ex GiNaC::Kronecker_dz_kernel::get_numerical_value (
 const ex & z,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of the  $g^{(n-1)}(z-z_j, K\tau)$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), [GiNaC::l](#), [n](#), [GiNaC::Pi](#), and [GiNaC::pow\(\)](#).

### 6.83.3.6 `series_coeff_impl()`

```
cln::cl_N GiNaC::Kronecker_dz_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The  $i$ -th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::bernoulli\(\)](#), [C\\_norm](#), [GiNaC::basic::evalf\(\)](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::l](#), [GiNaC::is\\_even\(\)](#), [K](#), [n](#), [GiNaC::Pi](#), [GiNaC::pow\(\)](#), [qbar](#), [tau](#), [GiNaC::numeric::to\\_int\(\)](#), and [z\\_j](#).

### 6.83.3.7 `do_print()`

```
void GiNaC::Kronecker_dz_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [C\\_norm](#), [K](#), [n](#), [GiNaC::ex::print\(\)](#), [tau](#), and [z\\_j](#).

## 6.83.4 Member Data Documentation

### 6.83.4.1 `n`

```
ex GiNaC::Kronecker_dz_kernel::n [protected]
```

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.2 `z_j`

```
ex GiNaC::Kronecker_dz_kernel::z_j [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.3 tau

`ex GiNaC::Kronecker_dz_kernel::tau` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.4 K

`ex GiNaC::Kronecker_dz_kernel::K` [protected]

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

### 6.83.4.5 C\_norm

`ex GiNaC::Kronecker_dz_kernel::C_norm` [protected]

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.84 GiNaC::lanczos\_coeffs Class Reference

### Public Member Functions

- [lanczos\\_coeffs](#) ()
- bool [sufficiently\\_accurate](#) (int digits)
- int [get\\_order](#) () const
- `cln::cl_N` [calc\\_lanczos\\_A](#) (const `cln::cl_N &`) const

### Private Attributes

- `std::vector< cln::cl_N > *` [current\\_vector](#)

### Static Private Attributes

- static `std::vector< cln::cl_N > *` [coeffs](#) = nullptr

## 6.84.1 Constructor & Destructor Documentation

### 6.84.1.1 lanczos\_coeffs()

`GiNaC::lanczos_coeffs::lanczos_coeffs ( )`

References [coeffs](#).

## 6.84.2 Member Function Documentation

### 6.84.2.1 sufficiently\_accurate()

```
bool GiNaC::lanczos_coeffs::sufficiently_accurate (
 int digits)
```

References [coeffs](#), and [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 6.84.2.2 get\_order()

```
int GiNaC::lanczos_coeffs::get_order () const [inline]
```

References [current\\_vector](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

### 6.84.2.3 calc\_lanczos\_A()

```
cln::cl_N GiNaC::lanczos_coeffs::calc_lanczos_A (
 const cln::cl_N & x) const
```

References [current\\_vector](#), and [x](#).

Referenced by [GiNaC::lgamma\(\)](#), and [GiNaC::tgamma\(\)](#).

## 6.84.3 Member Data Documentation

### 6.84.3.1 coeffs

```
std::vector< cln::cl_N > * GiNaC::lanczos_coeffs::coeffs = nullptr [static], [private]
```

Referenced by [lanczos\\_coeffs\(\)](#), and [sufficiently\\_accurate\(\)](#).

### 6.84.3.2 current\_vector

```
std::vector<cln::cl_N>* GiNaC::lanczos_coeffs::current_vector [private]
```

Referenced by [calc\\_lanczos\\_A\(\)](#), [get\\_order\(\)](#), and [sufficiently\\_accurate\(\)](#).

The documentation for this class was generated from the following file:

- [numeric.cpp](#)

## 6.85 std::less< GiNaC::ptr< T > > Struct Template Reference

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

```
#include <ptr.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [GiNaC::ptr< T >](#) &lhs, const [GiNaC::ptr< T >](#) &rhs) const

### 6.85.1 Detailed Description

```
template<class T>
struct std::less< GiNaC::ptr< T > >
```

Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.

for the use as std::map keys).

### 6.85.2 Member Function Documentation

#### 6.85.2.1 operator()

```
template<class T >
bool std::less< GiNaC::ptr< T > >::operator() (
 const GiNaC::ptr< T > & lhs,
 const GiNaC::ptr< T > & rhs) const [inline]
```

The documentation for this struct was generated from the following file:

- [ptr.h](#)

## 6.86 GiNaC::library\_init Class Reference

Helper class to initialize the library.

```
#include <ex.h>
```

### Public Member Functions

- [library\\_init](#) ()  
*Ctor of static initialization helpers.*
- [~library\\_init](#) ()  
*Dtor of static initialization helpers.*

## Static Private Member Functions

- static void [init\\_unarchivers](#) ()

## Static Private Attributes

- static int [count](#) = 0

*How many static objects were created? Only the first one must create the static flyweights on the heap.*

### 6.86.1 Detailed Description

Helper class to initialize the library.

There must be one static object of this class in every object file that makes use of our flyweights in order to guarantee proper initialization. Hence we put it into this file which is included by every relevant file anyways. This is modeled after section [jos::Init] of the C++ standard, where cout and friends are set up.

See also

[utils.cpp](#)

### 6.86.2 Constructor & Destructor Documentation

#### 6.86.2.1 [library\\_init\(\)](#)

```
GiNaC::library_init::library_init ()
```

Ctor of static initialization helpers.

The fist call to this is going to initialize the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), [GiNaC::\\_num0\\_bp](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num10\\_p](#), [GiNaC::\\_num11\\_p](#), [GiNaC::\\_num120\\_p](#), [GiNaC::\\_num12\\_p](#), [GiNaC::\\_num15\\_p](#), [GiNaC::\\_num18\\_p](#), [GiNaC::\\_num1\\_2\\_p](#), [GiNaC::\\_num1\\_3\\_p](#), [GiNaC::\\_num1\\_4\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num20\\_p](#), [GiNaC::\\_num24\\_p](#), [GiNaC::\\_num25\\_p](#), [GiNaC::\\_num2\\_p](#), [GiNaC::\\_num30\\_p](#), [GiNaC::\\_num3\\_p](#), [GiNaC::\\_num48\\_p](#), [GiNaC::\\_num4\\_p](#), [GiNaC::\\_num5\\_p](#), [GiNaC::\\_num60\\_p](#), [GiNaC::\\_num6\\_p](#), [GiNaC::\\_num7\\_p](#), [GiNaC::\\_num8\\_p](#), [GiNaC::\\_num9\\_p](#), [GiNaC::\\_num\\_10\\_p](#), [GiNaC::\\_num\\_11\\_p](#), [GiNaC::\\_num\\_120\\_p](#), [GiNaC::\\_num\\_12\\_p](#), [GiNaC::\\_num\\_15\\_p](#), [GiNaC::\\_num\\_18\\_p](#), [GiNaC::\\_num\\_1\\_2\\_p](#), [GiNaC::\\_num\\_1\\_3\\_p](#), [GiNaC::\\_num\\_1\\_4\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::\\_num\\_20\\_p](#), [GiNaC::\\_num\\_24\\_p](#), [GiNaC::\\_num\\_25\\_p](#), [GiNaC::\\_num\\_2\\_p](#), [GiNaC::\\_num\\_30\\_p](#), [GiNaC::\\_num\\_3\\_p](#), [GiNaC::\\_num\\_48\\_p](#), [GiNaC::\\_num\\_4\\_p](#), [GiNaC::\\_num\\_5\\_p](#), [GiNaC::\\_num\\_60\\_p](#), [GiNaC::\\_num\\_6\\_p](#), [GiNaC::\\_num\\_7\\_p](#), [GiNaC::\\_num\\_8\\_p](#), [GiNaC::\\_num\\_9\\_p](#), and [count](#).

### 6.86.2.2 ~library\_init()

```
GiNaC::library_init::~~library_init ()
```

Dtor of static initialization helpers.

The last call to this is going to shut down the library, the others do nothing.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex10](#), [GiNaC::\\_ex11](#), [GiNaC::\\_ex12](#), [GiNaC::\\_ex120](#), [GiNaC::\\_ex15](#), [GiNaC::\\_ex18](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex1\\_4](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex20](#), [GiNaC::\\_ex24](#), [GiNaC::\\_ex25](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex30](#), [GiNaC::\\_ex4](#), [GiNaC::\\_ex48](#), [GiNaC::\\_ex5](#), [GiNaC::\\_ex6](#), [GiNaC::\\_ex60](#), [GiNaC::\\_ex7](#), [GiNaC::\\_ex8](#), [GiNaC::\\_ex9](#), [GiNaC::\\_ex\\_1](#), [GiNaC::\\_ex\\_10](#), [GiNaC::\\_ex\\_11](#), [GiNaC::\\_ex\\_12](#), [GiNaC::\\_ex\\_120](#), [GiNaC::\\_ex\\_15](#), [GiNaC::\\_ex\\_18](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_1\\_4](#), [GiNaC::\\_ex\\_2](#), [GiNaC::\\_ex\\_20](#), [GiNaC::\\_ex\\_24](#), [GiNaC::\\_ex\\_25](#), [GiNaC::\\_ex\\_3](#), [GiNaC::\\_ex\\_30](#), [GiNaC::\\_ex\\_4](#), [GiNaC::\\_ex\\_48](#), [GiNaC::\\_ex\\_5](#), [GiNaC::\\_ex\\_6](#), [GiNaC::\\_ex\\_60](#), [GiNaC::\\_ex\\_7](#), [GiNaC::\\_ex\\_8](#), [GiNaC::\\_ex\\_9](#), and [count](#).

## 6.86.3 Member Function Documentation

### 6.86.3.1 init\_unarchivers()

```
void GiNaC::library_init::init_unarchivers () [static], [private]
```

## 6.86.4 Member Data Documentation

### 6.86.4.1 count

```
int GiNaC::library_init::count = 0 [static], [private]
```

How many static objects were created? Only the first one must create the static flyweights on the heap.

Referenced by [library\\_init\(\)](#), and [~library\\_init\(\)](#).

The documentation for this class was generated from the following files:

- [ex.h](#)
- [utils.cpp](#)

## 6.87 GiNaC::make\_flat\_inserter Class Reference

Class to handle the renaming of dummy indices.

```
#include <expairseq.h>
```

### Public Member Functions

- [make\\_flat\\_inserter](#) (const [epvector](#) &epv, bool b)
- [make\\_flat\\_inserter](#) (const [exvector](#) &v, bool b)
- [ex\\_handle\\_factor](#) (const [ex](#) &x, const [ex](#) &coeff)

## Private Member Functions

- void [combine\\_indices](#) (const [exvector](#) &dummies\_of\_factor)

## Private Attributes

- bool [do\\_renaming](#)
- [exvector](#) [used\\_indices](#)

### 6.87.1 Detailed Description

Class to handle the renaming of dummy indices.

It holds a vector of indices that are being used in the expression so far. If the same index occurs again as a dummy index in a factor, it is to be renamed. Unless dummy index renaming was switched off, of course ;-)

### 6.87.2 Constructor & Destructor Documentation

#### 6.87.2.1 [make\\_flat\\_inserter\(\)](#) [1/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
 const epvector & epv,
 bool b) [inline]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [combine\\_indices\(\)](#), and [do\\_renaming](#).

#### 6.87.2.2 [make\\_flat\\_inserter\(\)](#) [2/2]

```
GiNaC::make_flat_inserter::make_flat_inserter (
 const exvector & v,
 bool b) [inline]
```

References [combine\\_indices\(\)](#), and [do\\_renaming](#).

### 6.87.3 Member Function Documentation

#### 6.87.3.1 [handle\\_factor\(\)](#)

```
ex GiNaC::make_flat_inserter::handle_factor (
 const ex & x,
 const ex & coeff) [inline]
```

References [GiNaC::coeff\(\)](#), [combine\\_indices\(\)](#), [do\\_renaming](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [used\\_indices](#), and [x](#).

Referenced by [GiNaC::ncmul::eval\(\)](#), [GiNaC::expairseq::make\\_flat\(\)](#), and [GiNaC::expairseq::make\\_flat\(\)](#).

### 6.87.3.2 combine\_indices()

```
void GiNaC::make_flat_inserter::combine_indices (
 const exvector & dummies_of_factor) [inline], [private]
```

References [used\\_indices](#).

Referenced by [handle\\_factor\(\)](#), [make\\_flat\\_inserter\(\)](#), and [make\\_flat\\_inserter\(\)](#).

## 6.87.4 Member Data Documentation

### 6.87.4.1 do\_renaming

```
bool GiNaC::make_flat_inserter::do_renaming [private]
```

Referenced by [handle\\_factor\(\)](#), [make\\_flat\\_inserter\(\)](#), and [make\\_flat\\_inserter\(\)](#).

### 6.87.4.2 used\_indices

```
exvector GiNaC::make_flat_inserter::used_indices [private]
```

Referenced by [combine\\_indices\(\)](#), and [handle\\_factor\(\)](#).

The documentation for this class was generated from the following file:

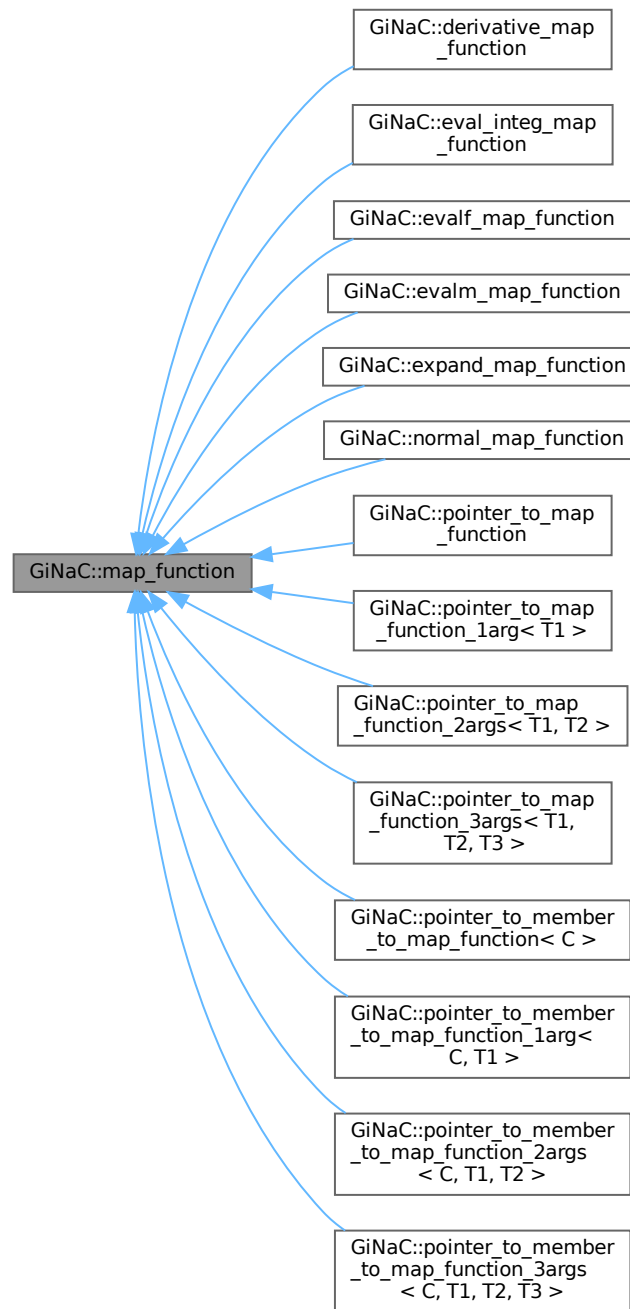
- [expairseq.h](#)

## 6.88 GiNaC::map\_function Struct Reference

Function object for map().

```
#include <basic.h>
```

Inheritance diagram for `GiNaC::map_function`:



### Public Types

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

### Public Member Functions

- virtual `~map_function` ()
- virtual `ex operator()` (const `ex` &`e`)=0

## 6.88.1 Detailed Description

Function object for map().

## 6.88.2 Member Typedef Documentation

### 6.88.2.1 argument\_type

```
typedef const ex& GiNaC::map_function::argument_type
```

### 6.88.2.2 result\_type

```
typedef ex GiNaC::map_function::result_type
```

## 6.88.3 Constructor & Destructor Documentation

### 6.88.3.1 ~map\_function()

```
virtual GiNaC::map_function::~~map_function () \[inline\], \[virtual\]
```

## 6.88.4 Member Function Documentation

### 6.88.4.1 operator>()

```
virtual ex GiNaC::map_function::operator() (
 const ex & e) \[pure virtual\]
```

Implemented in [GiNaC::evalf\\_map\\_function](#), [GiNaC::evalm\\_map\\_function](#), [GiNaC::eval\\_integ\\_map\\_function](#), [GiNaC::derivative\\_map\\_function](#), [GiNaC::expand\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2 >](#) and [GiNaC::normal\\_map\\_function](#).

The documentation for this struct was generated from the following file:

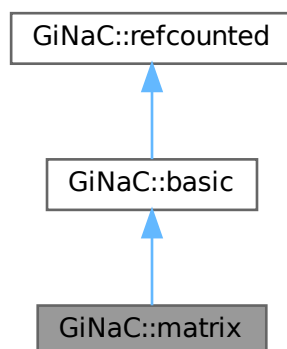
- [basic.h](#)

## 6.89 GiNaC::matrix Class Reference

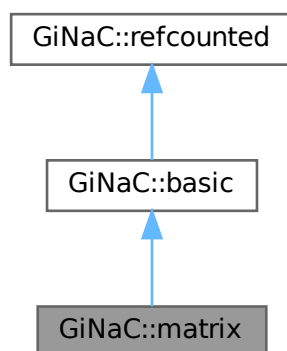
Symbolic matrices.

```
#include <matrix.h>
```

Inheritance diagram for GiNaC::matrix:



Collaboration diagram for GiNaC::matrix:



### Public Member Functions

- `matrix` (unsigned `r`, unsigned `c`)  
*Very common ctor.*
- `matrix` (unsigned `r`, unsigned `c`, const `lst` &l)  
*Construct matrix from (flat) list of elements.*

- **matrix** (std::initializer\_list< std::initializer\_list< **ex** > > l)  
*Construct a matrix from an 2 dimensional initializer list.*
- **size\_t nops** () const override  
*nops is defined to be rows x columns.*
- **ex op** (size\_t i) const override  
*returns matrix entry at position (i/col, icol).*
- **ex & let\_op** (size\_t i) override  
*returns writable matrix entry at position (i/col, icol).*
- **ex evalm** () const override  
*Evaluate sums, products and integer powers of matrices.*
- **ex subs** (const **exmap** &m, unsigned **options**=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- **ex eval\_indexed** (const **basic** &i) const override  
*Automatic symbolic evaluation of an indexed matrix.*
- **ex add\_indexed** (const **ex** &self, const **ex** &other) const override  
*Sum of two indexed matrices.*
- **ex scalar\_mul\_indexed** (const **ex** &self, const **numeric** &other) const override  
*Product of an indexed matrix with a number.*
- **bool contract\_with** (exvector::iterator self, exvector::iterator other, **exvector** &v) const override  
*Contraction of an indexed matrix with something else.*
- **ex conjugate** () const override  
*Complex conjugate every matrix entry.*
- **ex real\_part** () const override
- **ex imag\_part** () const override
- **void archive** (**archive\_node** &n) const override  
*Save (a.k.a.*
- **void read\_archive** (const **archive\_node** &n, **lst** &syms) override  
*Read (a.k.a.*
- **unsigned rows** () const  
*Get number of rows.*
- **unsigned cols** () const  
*Get number of columns.*
- **matrix add** (const **matrix** &other) const  
*Sum of matrices.*
- **matrix sub** (const **matrix** &other) const  
*Difference of matrices.*
- **matrix mul** (const **matrix** &other) const  
*Product of matrices.*
- **matrix mul** (const **numeric** &other) const  
*Product of matrix and scalar.*
- **matrix mul\_scalar** (const **ex** &other) const  
*Product of matrix and scalar expression.*
- **matrix pow** (const **ex** &expn) const  
*Power of a matrix.*
- **const ex & operator()** (unsigned ro, unsigned co) const  
*operator() to access elements for reading.*
- **ex & operator()** (unsigned ro, unsigned co)  
*operator() to access elements for writing.*
- **matrix & set** (unsigned ro, unsigned co, const **ex** &value)
- **matrix transpose** () const  
*Transposed of an m x n matrix, producing a new n x m matrix object that represents the transposed.*

- `ex determinant` (unsigned algo=`determinant_algo::automatic`) const  
*Determinant of square matrix.*
- `ex trace` () const  
*Trace of a matrix.*
- `ex charpoly` (const `ex` &lambda) const  
*Characteristic Polynomial.*
- `matrix inverse` () const  
*Inverse of this matrix, with automatic algorithm selection.*
- `matrix inverse` (unsigned algo) const  
*Inverse of this matrix.*
- `matrix solve` (const `matrix` &vars, const `matrix` &rhs, unsigned algo=`solve_algo::automatic`) const  
*Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.*
- unsigned `rank` () const  
*Compute the rank of this matrix.*
- unsigned `rank` (unsigned `solve_algo`) const  
*Compute the rank of this matrix using the given algorithm, which should be a member of enum `solve_algo`.*
- bool `is_zero_matrix` () const  
*Function to check that all elements of the matrix are zero.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool distributed=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `return_type_t return_type_tinfo` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const
- Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const
- Compare objects syntactically to establish canonical ordering.*

  - bool `is_equal` (const `basic` &other) const
- Test for syntactic equality.*

  - const `basic` & `hold` () const
- Stop further evaluation.*

  - unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned f) const
- Set some `status_flags`.*

  - const `basic` & `clearflag` (unsigned f) const
- Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [matrix](#) (unsigned r, unsigned c, const [exvector](#) &m2)  
*Ctor from representation, for internal use only.*
- [matrix](#) (unsigned r, unsigned c, [exvector](#) &&m2)
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [ex\\_determinant\\_minor](#) () const  
*Recursive determinant for small matrices having at least one symbolic entry.*
- std::vector< unsigned > [echelon\\_form](#) (unsigned algo, int n)
- int [gauss\\_elimination](#) (const bool det=false)  
*Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.*
- int [division\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.*
- int [fraction\\_free\\_elimination](#) (const bool det=false)  
*Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.*
- std::vector< unsigned > [markowitz\\_elimination](#) (unsigned n)
- int [pivot](#) (unsigned ro, unsigned co, bool symbolic=true)  
*Partial pivoting method for matrix elimination schemes.*
- void [print\\_elements](#) (const [print\\_context](#) &c, const char \*row\_start, const char \*row\_end, const char \*row↵\_sep, const char \*col\_sep) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex\\_eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual [ex\\_derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [row](#)  
*number of rows*
- unsigned [col](#)  
*number of columns*
- [exvector m](#)  
*representation (cols indexed first)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.89.1 Detailed Description

Symbolic matrices.

### 6.89.2 Constructor & Destructor Documentation

#### 6.89.2.1 [matrix\(\)](#) [1/5]

```
GiNaC::matrix::matrix (
 unsigned r,
 unsigned c)
```

Very common ctor.

Initializes to r x c-dimensional zero-matrix.

#### Parameters

|          |                |
|----------|----------------|
| <i>r</i> | number of rows |
| <i>c</i> | number of cols |

References [GiNaC::\\_ex0](#), [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [add\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [imag\\_part\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [transpose\(\)](#).

#### 6.89.2.2 [matrix\(\)](#) [2/5]

```
GiNaC::matrix::matrix (
 unsigned r,
```

```
 unsigned c,
 const lst & l)
```

Construct matrix from (flat) list of elements.

If the list has fewer elements than the matrix, the remaining matrix elements are set to zero. If the list has more elements than the matrix, the excessive elements are thrown away.

References [c](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [r](#), [GiNaC::basic::setflag\(\)](#), and [x](#).

### 6.89.2.3 matrix() [3/5]

```
GiNaC::matrix::matrix (
 std::initializer_list< std::initializer_list< ex > > l)
```

Construct a matrix from an 2 dimensional initializer list.

Throws an exception if some row has a different length than all the others.

References [c](#), [col](#), [m](#), [GiNaC::status\\_flags::not\\_shareable](#), [r](#), [row](#), and [GiNaC::basic::setflag\(\)](#).

### 6.89.2.4 matrix() [4/5]

```
GiNaC::matrix::matrix (
 unsigned r,
 unsigned c,
 const exvector & m2) [protected]
```

Ctor from representation, for internal use only.

References [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

### 6.89.2.5 matrix() [5/5]

```
GiNaC::matrix::matrix (
 unsigned r,
 unsigned c,
 exvector && m2) [protected]
```

References [GiNaC::status\\_flags::not\\_shareable](#), and [GiNaC::basic::setflag\(\)](#).

## 6.89.3 Member Function Documentation

### 6.89.3.1 nops()

```
size_t GiNaC::matrix::nops () const [override], [virtual]
```

nops is defined to be rows x columns.

Reimplemented from [GiNaC::basic](#).

References [col](#), and [row](#).

Referenced by [let\\_op\(\)](#), and [op\(\)](#).

### 6.89.3.2 op()

```
ex GiNaC::matrix::op (
 size_t i) const [override], [virtual]
```

returns matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

Referenced by [contract\\_with\(\)](#).

### 6.89.3.3 let\_op()

```
ex & GiNaC::matrix::let_op (
 size_t i) [override], [virtual]
```

returns writable matrix entry at position (i/col, icol).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [m](#), and [nops\(\)](#).

### 6.89.3.4 evalm()

```
ex GiNaC::matrix::evalm () const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

### 6.89.3.5 subs()

```
ex GiNaC::matrix::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [c](#), [col](#), [m](#), [matrix\(\)](#), [options](#), [r](#), [row](#), and [subs\(\)](#).

Referenced by [fraction\\_free\\_elimination\(\)](#), and [subs\(\)](#).

### 6.89.3.6 eval\_indexed()

```
ex GiNaC::matrix::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed matrix.

Reimplemented from [GiNaC::basic](#).

References [col](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [row](#), and [trace\(\)](#).

### 6.89.3.7 add\_indexed()

```
ex GiNaC::matrix::add_indexed (
 const ex & self,
 const ex & other) const [override], [virtual]
```

Sum of two indexed matrices.

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [row](#), and [transpose\(\)](#).

### 6.89.3.8 scalar\_mul\_indexed()

```
ex GiNaC::matrix::scalar_mul_indexed (
 const ex & self,
 const numeric & other) const [override], [virtual]
```

Product of an indexed matrix with a number.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [mul\(\)](#), [GiNaC::ex::nops\(\)](#), and [GiNaC::ex::op\(\)](#).

### 6.89.3.9 contract\_with()

```
bool GiNaC::matrix::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed matrix with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [mul\(\)](#), [GiNaC::ex::op\(\)](#), [op\(\)](#), [row](#), and [transpose\(\)](#).

### 6.89.3.10 conjugate()

```
ex GiNaC::matrix::conjugate () const [override], [virtual]
```

Complex conjugate every matrix entry.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [col](#), [GiNaC::ex::conjugate\(\)](#), [m](#), [matrix\(\)](#), [row](#), and [x](#).

### 6.89.3.11 real\_part()

```
ex GiNaC::matrix::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

### 6.89.3.12 imag\_part()

```
ex GiNaC::matrix::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

### 6.89.3.13 archive()

```
void GiNaC::matrix::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [n](#), and [row](#).

### 6.89.3.14 read\_archive()

```
void GiNaC::matrix::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [col](#), [m](#), [n](#), and [row](#).

### 6.89.3.15 match\_same\_type()

```
bool GiNaC::matrix::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [col](#), [cols\(\)](#), [GINAC\\_ASSERT](#), [row](#), and [rows\(\)](#).

### 6.89.3.16 return\_type()

```
unsigned GiNaC::matrix::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#).

### 6.89.3.17 rows()

```
unsigned GiNaC::matrix::rows () const [inline]
```

Get number of rows.

References [row](#).

Referenced by [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [solve\(\)](#), and [transpose\(\)](#).

### 6.89.3.18 cols()

```
unsigned GiNaC::matrix::cols () const [inline]
```

Get number of columns.

References [col](#).

Referenced by [determinant\\_minor\(\)](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [GiNaC::lsolve\(\)](#), [match\\_same\\_type\(\)](#), [mul\(\)](#), [solve\(\)](#), and [transpose\(\)](#).

### 6.89.3.19 add()

```
matrix GiNaC::matrix::add (
 const matrix & other) const
```

Sum of matrices.

## Exceptions

|                    |                         |
|--------------------|-------------------------|
| <i>logic_error</i> | (incompatible matrices) |
|--------------------|-------------------------|

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

Referenced by [add\\_indexed\(\)](#), and [GiNaC::add::evalm\(\)](#).

**6.89.3.20 sub()**

```
matrix GiNaC::matrix::sub (
 const matrix & other) const
```

Difference of matrices.

## Exceptions

|                    |                         |
|--------------------|-------------------------|
| <i>logic_error</i> | (incompatible matrices) |
|--------------------|-------------------------|

References [col](#), [m](#), [matrix\(\)](#), and [row](#).

**6.89.3.21 mul() [1/2]**

```
matrix GiNaC::matrix::mul (
 const matrix & other) const
```

Product of matrices.

## Exceptions

|                    |                         |
|--------------------|-------------------------|
| <i>logic_error</i> | (incompatible matrices) |
|--------------------|-------------------------|

References [c](#), [col](#), [cols\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [matrix\(\)](#), [row](#), and [rows\(\)](#).

Referenced by [charpoly\(\)](#), [contract\\_with\(\)](#), [GiNaC::ncmul::evalm\(\)](#), [pow\(\)](#), and [scalar\\_mul\\_indexed\(\)](#).

**6.89.3.22 mul() [2/2]**

```
matrix GiNaC::matrix::mul (
 const numeric & other) const
```

Product of matrix and scalar.

References [c](#), [col](#), [m](#), [matrix\(\)](#), [r](#), and [row](#).

**6.89.3.23 mul\_scalar()**

```
matrix GiNaC::matrix::mul_scalar (
 const ex & other) const
```

Product of matrix and scalar expression.

References [c](#), [col](#), [GiNaC::return\\_types::commutative](#), [m](#), [matrix\(\)](#), [r](#), [GiNaC::ex::return\\_type\(\)](#), and [row](#).

**6.89.3.24 pow()**

```
matrix GiNaC::matrix::pow (
 const ex & expn) const
```

Power of a matrix.

Currently handles integer exponents only.

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num2\\_p](#), [col](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [inverse\(\)](#), [GiNaC::numeric::is\\_odd\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [mul\(\)](#), [GiNaC::info\\_flags::negative](#), [r](#), and [row](#).

**6.89.3.25 operator>() [1/2]**

```
const ex & GiNaC::matrix::operator() (
 unsigned ro,
 unsigned co) const
```

`operator()` to access elements for reading.

**Parameters**

|           |                   |
|-----------|-------------------|
| <i>ro</i> | row of element    |
| <i>co</i> | column of element |

**Exceptions**

|                    |                      |
|--------------------|----------------------|
| <i>range_error</i> | (index out of range) |
|--------------------|----------------------|

References [col](#), [m](#), and [row](#).

**6.89.3.26 operator>() [2/2]**

```
ex & GiNaC::matrix::operator() (
 unsigned ro,
 unsigned co)
```

`operator()` to access elements for writing.

## Parameters

|           |                   |
|-----------|-------------------|
| <i>ro</i> | row of element    |
| <i>co</i> | column of element |

## Exceptions

|                    |                      |
|--------------------|----------------------|
| <i>range_error</i> | (index out of range) |
|--------------------|----------------------|

References [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [m](#), and [row](#).

**6.89.3.27 set()**

```
matrix & GiNaC::matrix::set (
 unsigned ro,
 unsigned co,
 const ex & value) [inline]
```

References [value](#).

**6.89.3.28 transpose()**

```
matrix GiNaC::matrix::transpose () const
```

Transposed of an  $m \times n$  matrix, producing a new  $n \times m$  matrix object that represents the transposed.

References [c](#), [cols\(\)](#), [m](#), [matrix\(\)](#), [r](#), and [rows\(\)](#).

Referenced by [add\\_indexed\(\)](#), [contract\\_with\(\)](#), and [GiNaC::transpose\(\)](#).

**6.89.3.29 determinant()**

```
ex GiNaC::matrix::determinant (
 unsigned algo = determinant_algo::automatic) const
```

Determinant of square matrix.

This routine doesn't actually calculate the determinant, it only implements some heuristics about which algorithm to run. If all the elements of the matrix are elements of an integral domain the determinant is also in that integral domain and the result is expanded only. If one or more elements are from a quotient field the determinant is usually also in that quotient field and the result is normalized before it is returned. This implies that the determinant of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b) & 1 \\ b/(a-b) & 1 \end{bmatrix}$  is returned as unity. (In this respect, it behaves like MapleV and unlike Mathematica.)

## Parameters

|             |                              |
|-------------|------------------------------|
| <i>algo</i> | allows to chose an algorithm |
|-------------|------------------------------|

**Returns**

the determinant as a new expression

**Exceptions**

|                    |                     |
|--------------------|---------------------|
| <i>logic_error</i> | (matrix not square) |
|--------------------|---------------------|

**See also**

[determinant\\_algo](#)

References [GiNaC::\\_ex0](#), [GiNaC::determinant\\_algo::automatic](#), [GiNaC::determinant\\_algo::bareiss](#), [c](#), [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::determinant\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::determinant\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::determinant\\_algo::laplace](#), [m](#), [matrix\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::basic::normal\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::permutation\\_sign\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), [row](#), and [GiNaC::ex::to\\_rational\(\)](#).

Referenced by [charpoly\(\)](#), and [GiNaC::tensepsilon::contract\\_with\(\)](#).

**6.89.3.30 trace()**

```
ex GiNaC::matrix::trace () const
```

Trace of a matrix.

The result is normalized if it is in some quotient field and expanded only otherwise. This implies that the trace of the symbolic 2x2 matrix  $\begin{bmatrix} a/(a-b), x \\ y, b/(b-a) \end{bmatrix}$  is recognized to be unity.

**Returns**

the sum of diagonal elements

**Exceptions**

|                    |                     |
|--------------------|---------------------|
| <i>logic_error</i> | (matrix not square) |
|--------------------|---------------------|

References [col](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::info\(\)](#), [m](#), [GiNaC::ex::normal\(\)](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), and [row](#).

Referenced by [charpoly\(\)](#), and [eval\\_indexed\(\)](#).

**6.89.3.31 charpoly()**

```
ex GiNaC::matrix::charpoly (
 const ex & lambda) const
```

Characteristic Polynomial.

Following mathematica notation the characteristic polynomial of a matrix  $M$  is defined as the determinant of  $(M - \lambda * 1)$  where  $1$  stands for the unit matrix of the same dimension as  $M$ . Note that some CASs define it with a sign inside the determinant which gives rise to an overall sign if the dimension is odd. This method returns the characteristic polynomial collected in powers of  $\lambda$  as a new expression.

#### Returns

characteristic polynomial as new expression

#### Exceptions

|                    |                     |
|--------------------|---------------------|
| <i>logic_error</i> | (matrix not square) |
|--------------------|---------------------|

#### See also

[matrix::determinant\(\)](#)

References [c](#), [col](#), [GiNaC::ex::collect\(\)](#), [determinant\(\)](#), [GiNaC::basic::ex](#), [m](#), [mul\(\)](#), [GiNaC::info\\_flags::numeric](#), [poly](#), [r](#), [row](#), and [trace\(\)](#).

#### 6.89.3.32 inverse() [1/2]

```
matrix GiNaC::matrix::inverse () const
```

Inverse of this matrix, with automatic algorithm selection.

References [GiNaC::solve\\_algo::automatic](#), and [inverse\(\)](#).

Referenced by [inverse\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::inverse\(\)](#), and [pow\(\)](#).

#### 6.89.3.33 inverse() [2/2]

```
matrix GiNaC::matrix::inverse (
 unsigned algo) const
```

Inverse of this matrix.

#### Parameters

|             |                                                            |
|-------------|------------------------------------------------------------|
| <i>algo</i> | selects the algorithm (one of <a href="#">solve_algo</a> ) |
|-------------|------------------------------------------------------------|

#### Returns

the inverted matrix

#### Exceptions

|                      |                     |
|----------------------|---------------------|
| <i>logic_error</i>   | (matrix not square) |
| <i>runtime_error</i> | (singular matrix)   |

References [GiNaC::\\_ex1](#), [c](#), [col](#), [r](#), [row](#), and [solve\(\)](#).

### 6.89.3.34 solve()

```
matrix GiNaC::matrix::solve (
 const matrix & vars,
 const matrix & rhs,
 unsigned algo = solve_algo::automatic) const
```

Solve a linear system consisting of a  $m \times n$  matrix and a  $m \times p$  right hand side by applying an elimination scheme to the augmented matrix.

#### Parameters

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>vars</i> | $n \times p$ matrix, all elements must be symbols |
| <i>rhs</i>  | $m \times p$ matrix                               |
| <i>algo</i> | selects the solving algorithm                     |

#### Returns

$n \times p$  solution matrix

#### Exceptions

|                         |                                          |
|-------------------------|------------------------------------------|
| <i>logic_error</i>      | (incompatible matrices)                  |
| <i>invalid_argument</i> | (1st argument must be matrix of symbols) |
| <i>runtime_error</i>    | (inconsistent linear system)             |

#### See also

[solve\\_algo](#)

References [c](#), [cols\(\)](#), [echelon\\_form\(\)](#), [GiNaC::basic::info\(\)](#), [m](#), [n](#), [GiNaC::basic::normal\(\)](#), [r](#), [GiNaC::rhs\(\)](#), [rows\(\)](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [inverse\(\)](#), [GiNaC::lsolve\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

### 6.89.3.35 rank() [1/2]

```
unsigned GiNaC::matrix::rank () const
```

Compute the rank of this matrix.

References [GiNaC::solve\\_algo::automatic](#), and [rank\(\)](#).

Referenced by [rank\(\)](#), and [GiNaC::rank\(\)](#).

**6.89.3.36 rank()** [2/2]

```
unsigned GiNaC::matrix::rank (
 unsigned solve_algo) const
```

Compute the rank of this matrix using the given algorithm, which should be a member of enum [solve\\_algo](#).

References [col](#), [echelon\\_form\(\)](#), [GINAC\\_ASSERT](#), [m](#), [r](#), and [row](#).

**6.89.3.37 is\_zero\_matrix()**

```
bool GiNaC::matrix::is_zero_matrix () const
```

Function to check that all elements of the matrix are zero.

References [m](#).

**6.89.3.38 determinant\_minor()**

```
ex GiNaC::matrix::determinant_minor () const [protected]
```

Recursive determinant for small matrices having at least one symbolic entry.

The basic algorithm, known as Laplace-expansion, is enhanced by some bookkeeping to avoid calculation of the same submatrices ("minors") more than once. According to W.M.Gentleman and S.C.Johnson this algorithm is better than elimination schemes for matrices of sparse multivariate polynomials and also for matrices of dense univariate polynomials if the matrix' dimension is larger than 7.

**Returns**

the determinant as a new expression (in expanded form)

**See also**

[matrix::determinant\(\)](#)

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [cols\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [n](#), and [r](#).

**6.89.3.39 echelon\_form()**

```
std::vector< unsigned > GiNaC::matrix::echelon_form (
 unsigned algo,
 int n) [protected]
```

References [GiNaC::solve\\_algo::automatic](#), [GiNaC::solve\\_algo::bareiss](#), [c](#), [col](#), [GiNaC::solve\\_algo::divfree](#), [division\\_free\\_elimination\(\)](#), [fraction\\_free\\_elimination\(\)](#), [GiNaC::solve\\_algo::gauss](#), [gauss\\_elimination\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), [GiNaC::solve\\_algo::markowitz](#), [markowitz\\_elimination\(\)](#), [n](#), [GiNaC::info\\_flags::numeric](#), [r](#), and [row](#).

Referenced by [rank\(\)](#), and [solve\(\)](#).

**6.89.3.40 gauss\_elimination()**

```
int GiNaC::matrix::gauss_elimination (
 const bool det = false) [protected]
```

Perform the steps of an ordinary Gaussian elimination to bring the m x n matrix into an upper echelon form.

The algorithm is ok for matrices with numeric coefficients but quite unsuited for symbolic matrices.

## Parameters

|            |                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>det</i> | may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [c](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::info\(\)](#), [GiNaC::is\\_zero\(\)](#), [m](#), [n](#), [GiNaC::ex::normal\(\)](#), [GiNaC::info\\_flags::numeric](#), [pivot\(\)](#), [r](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.41 division\_free\_elimination()**

```
int GiNaC::matrix::division_free_elimination (
 const bool det = false) [protected]
```

Perform the steps of division free elimination to bring the m x n matrix into an upper echelon form.

## Parameters

|            |                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>det</i> | may be set to true to save a lot of space if one is only interested in the diagonal elements (i.e. for calculating determinants). The others are set to zero in this case. |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [c](#), [cols\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GINAC\\_ASSERT](#), [m](#), [n](#), [pivot\(\)](#), [r](#), and [rows\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.42 fraction\_free\_elimination()**

```
int GiNaC::matrix::fraction_free_elimination (
 const bool det = false) [protected]
```

Perform the steps of Bareiss' one-step fraction free elimination to bring the matrix into an upper echelon form.

Fraction free elimination means that divide is used straightforwardly, without computing GCDs first. This is possible, since we know the divisor at each step.

## Parameters

|            |                                                                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>det</i> | may be set to true to save a lot of space if one is only interested in the last element (i.e. for calculating determinants). The others are set to zero in this case. |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

sign is 1 if an even number of rows was swapped, -1 if an odd number of rows was swapped and 0 if the matrix is singular.

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [cols\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [m](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::normal\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::ex::op\(\)](#), [r](#), [rows\(\)](#), [subs\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

Referenced by [determinant\(\)](#), and [echelon\\_form\(\)](#).

**6.89.3.43 markowitz\_elimination()**

```
std::vector< unsigned > GiNaC::matrix::markowitz_elimination (
 unsigned n) [protected]
```

References [GiNaC::\\_ex0](#), [c](#), [col](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [k](#), [m](#), [n](#), [GiNaC::basic::normal\(\)](#), [r](#), [row](#), and [std::swap\(\)](#).

Referenced by [echelon\\_form\(\)](#).

**6.89.3.44 pivot()**

```
int GiNaC::matrix::pivot (
 unsigned ro,
 unsigned co,
 bool symbolic = true) [protected]
```

Partial pivoting method for matrix elimination schemes.

Usual pivoting (symbolic==false) returns the index to the element with the largest absolute value in column ro and swaps the current row with the one where the element was found. With (symbolic==true) it does the same thing with the first non-zero element.

## Parameters

|                 |                                                                                                                       |
|-----------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>ro</i>       | is the row from where to begin                                                                                        |
| <i>co</i>       | is the column to be inspected                                                                                         |
| <i>symbolic</i> | signal if we want the first non-zero element to be pivoted (true) or the one with the largest absolute value (false). |

## Returns

0 if no interchange occurred, -1 if all are zero (usually signaling a degeneracy) and positive integer k means that rows ro and k were swapped.

References [GiNaC::abs\(\)](#), [c](#), [col](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [GiNaC::basic::expand\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::numeric::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [k](#), [m](#), [row](#), and [GiNaC::swap\(\)](#).

Referenced by [division\\_free\\_elimination\(\)](#), and [gauss\\_elimination\(\)](#).

#### 6.89.3.45 print\_elements()

```
void GiNaC::matrix::print_elements (
 const print_context & c,
 const char * row_start,
 const char * row_end,
 const char * row_sep,
 const char * col_sep) const [protected]
```

References [c](#), [col](#), [m](#), and [row](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

#### 6.89.3.46 do\_print()

```
void GiNaC::matrix::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_elements\(\)](#).

#### 6.89.3.47 do\_print\_latex()

```
void GiNaC::matrix::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), [col](#), and [print\\_elements\(\)](#).

#### 6.89.3.48 do\_print\_python\_repr()

```
void GiNaC::matrix::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_elements\(\)](#).

### 6.89.4 Member Data Documentation

#### 6.89.4.1 row

```
unsigned GiNaC::matrix::row [protected]
```

number of rows

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)](#), [operator\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [rows\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

### 6.89.4.2 col

`unsigned GiNaC::matrix::col` [protected]

number of columns

Referenced by [add\(\)](#), [add\\_indexed\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [cols\(\)](#), [conjugate\(\)](#), [contract\\_with\(\)](#), [determinant\(\)](#), [do\\_print\\_latex\(\)](#), [echelon\\_form\(\)](#), [eval\\_indexed\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [markowitz\\_elimination\(\)](#), [match\\_same\\_type\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [nops\(\)](#), [operator\(\)\(\)](#), [operator\(\)\(\)](#), [pivot\(\)](#), [pow\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [sub\(\)](#), [subs\(\)](#), and [trace\(\)](#).

### 6.89.4.3 m

`exvector GiNaC::matrix::m` [protected]

representation (cols indexed first)

Referenced by [add\(\)](#), [archive\(\)](#), [charpoly\(\)](#), [conjugate\(\)](#), [determinant\(\)](#), [determinant\\_minor\(\)](#), [division\\_free\\_elimination\(\)](#), [echelon\\_form\(\)](#), [fraction\\_free\\_elimination\(\)](#), [gauss\\_elimination\(\)](#), [imag\\_part\(\)](#), [is\\_zero\\_matrix\(\)](#), [let\\_op\(\)](#), [markowitz\\_elimination\(\)](#), [matrix\(\)](#), [matrix\(\)](#), [mul\(\)](#), [mul\(\)](#), [mul\\_scalar\(\)](#), [op\(\)](#), [operator\(\)\(\)](#), [operator\(\)\(\)](#), [pivot\(\)](#), [print\\_elements\(\)](#), [rank\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [solve\(\)](#), [sub\(\)](#), [subs\(\)](#), [trace\(\)](#), and [transpose\(\)](#).

The documentation for this class was generated from the following files:

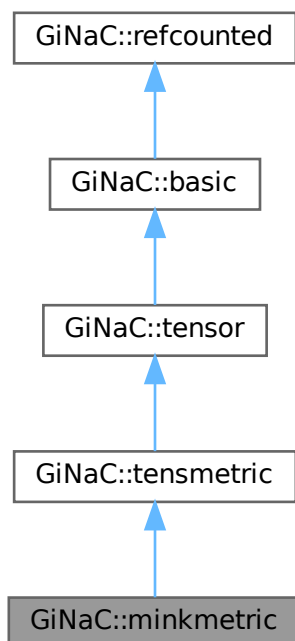
- [matrix.h](#)
- [matrix.cpp](#)

## 6.90 GiNaC::minkmetric Class Reference

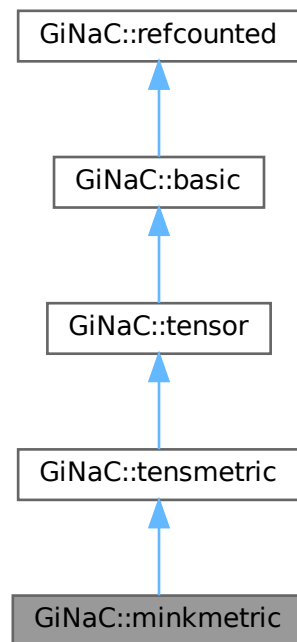
This class represents a Minkowski metric tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::minkmetric:



Collaboration diagram for GiNaC::minkmetric:



### Public Member Functions

- `minkmetric` (bool `pos_sig`)  
*Construct Lorentz metric tensor with given signature.*
- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed Lorentz metric tensor.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &`syms`) override  
*Read (a.k.a.*

### Public Member Functions inherited from `GiNaC::tensmetric`

- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &`s`) const

*Return degree of highest power in object `s`.*
- virtual int `ldegree` (const `ex` &`s`) const

*Return degree of lowest power in object `s`.*
- virtual `ex` `coeff` (const `ex` &`s`, int `n`=1) const

*Return coefficient of degree `n` in object `s`.*
- virtual `ex` `expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex` `collect` (const `ex` &`s`, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex` `series` (const `relational` &`r`, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex` `normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const

*Default implementation of `ex::normal()`.*
- virtual `ex` `to_rational` (`exmap` &`repl`) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex` `to_polynomial` (`exmap` &`repl`) const
- virtual `numeric` `integer_content` () const
- virtual `ex` `smod` (const `numeric` &`xi`) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric` `max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector` `get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex` `add_indexed` (const `ex` &`self`, const `ex` &`other`) const

*Add two indexed expressions.*
- virtual `ex` `scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const

*Multiply an indexed expression with a scalar.*
- virtual `return_type_t` `return_type_tinfo` () const
- virtual `ex` `conjugate` () const
- virtual `ex` `real_part` () const
- virtual `ex` `imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &`c`, unsigned `level`) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned `level`) const

*Like `print()`, but dispatch to the specified class.*
- `ex` `subs_one_level` (const `exmap` &`m`, unsigned `options`) const

*Helper function for `subs()`.*
- `ex` `diff` (const `symbol` &`s`, unsigned `nth`=1) const

*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &`other`) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &`other`) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const

*Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensmetric](#)

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- bool [pos\\_sig](#)  
*If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.90.1 Detailed Description

This class represents a Minkowski metric tensor.

It has all the properties of a metric tensor and is (as a matrix) equal to  $\text{diag}(1,-1,-1,\dots)$  or  $\text{diag}(-1,1,1,\dots)$ .

## 6.90.2 Constructor & Destructor Documentation

### 6.90.2.1 minkmetric()

```
GiNaC::minkmetric::minkmetric (
 bool pos_sig)
```

Construct Lorentz metric tensor with given signature.

## 6.90.3 Member Function Documentation

### 6.90.3.1 info()

```
bool GiNaC::minkmetric::info (
 unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.90.3.2 eval\_indexed()

```
ex GiNaC::minkmetric::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed Lorentz metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), and [pos\\_sig](#).

### 6.90.3.3 archive()

```
void GiNaC::minkmetric::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

### 6.90.3.4 read\_archive()

```
void GiNaC::minkmetric::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), and [pos\\_sig](#).

### 6.90.3.5 return\_type()

```
unsigned GiNaC::minkmetric::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.90.3.6 do\_print()

```
void GiNaC::minkmetric::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.90.3.7 do\_print\_latex()

```
void GiNaC::minkmetric::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

## 6.90.4 Member Data Documentation

### 6.90.4.1 pos\_sig

```
bool GiNaC::minkmetric::pos_sig [private]
```

If true, the metric is  $\text{diag}(-1, 1, 1, \dots)$ .

Otherwise it is  $\text{diag}(1, -1, -1, \dots)$ .

Referenced by [archive\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

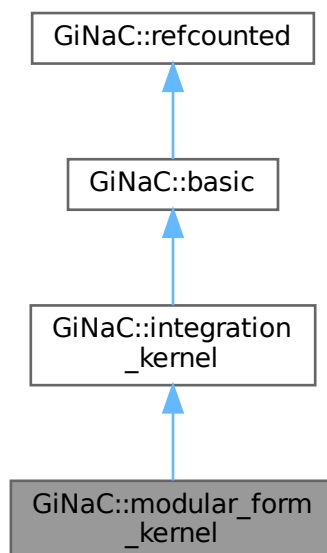
- [tensor.h](#)
- [tensor.cpp](#)

## 6.91 GiNaC::modular\_form\_kernel Class Reference

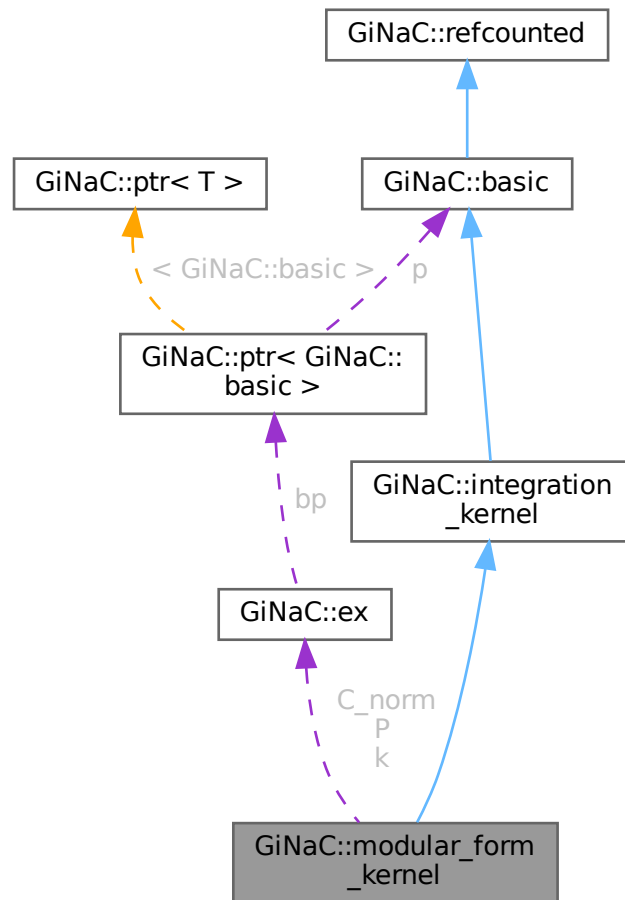
A kernel corresponding to a polynomial in Eisenstein series.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::modular\_form\_kernel:



Collaboration diagram for GiNaC::modular\_form\_kernel:



## Public Member Functions

- `modular_form_kernel` (const `ex` &`k`, const `ex` &`P`, const `ex` &`C_norm=numeric(1)`)
- `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const override  
*The series method for this class returns the qbar-expansion of the modular form, without an additional factor of C\_norm/qbar.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position i.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position i.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- `ex Laurent_series` (const `ex` &`qbar`, int `order`) const override  
*Returns the Laurent series, starting possibly with the pole term.*
- `ex get_numerical_value` (const `ex` &`qbar`, int `N_trunc=0`) const override  
*Returns the value of the modular form.*
- `ex q_expansion_modular_form` (const `ex` &`q`, int `order`) const

## Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex](#) [get\\_series\\_coeff](#) (int i) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- cln::cl\_N [series\\_coeff](#) (int i) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex](#) [eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex](#) [evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex](#) [evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex](#) [eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex](#) [eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual [ex](#) [operator\[\]](#) (const [ex](#) &index) const
- virtual [ex](#) [operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex](#) [subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const

- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
  - template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

  - `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*

  - `ex diff` (const `symbol` &s, unsigned nth=1) const

Default interface of *nth* derivative *ex::diff(s, n)*.

- int [compare](#) (const [basic](#) &other) const  
Compare objects syntactically to establish canonical ordering.
- bool [is\\_equal](#) (const [basic](#) &other) const  
Test for syntactic equality.
- const [basic](#) & [hold](#) () const  
Stop further evaluation.
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
Set some *status\_flags*.
- const [basic](#) & [clearflag](#) (unsigned f) const  
Clear some *status\_flags*.

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- bool [uses\\_Laurent\\_series](#) () const override  
Returns true, if the coefficients are computed from the Laurent series (in which case the method *Laurent\_series* needs to be implemented).
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::integration\\_kernel](#)

- virtual [cln::cl\\_N](#) [series\\_coeff\\_impl](#) (int i) const  
For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.
- [ex](#) [get\\_numerical\\_value\\_impl](#) (const [ex](#) &lambda, const [ex](#) &pre, int shift, int N\_trunc) const  
The actual implementation for computing a numerical value for the integrand.
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex](#) [eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
Returns true if the attributes of two objects are similar enough for a match.
- virtual [ex](#) [derivative](#) (const [symbol](#) &s) const  
Default implementation of *ex::diff()*.
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
Returns order relation between two objects of same type.
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
Returns true if two objects of same type are equal.
- virtual unsigned [calchash](#) () const

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

- void [ensure\\_if\\_modifiable](#) () const

Ensure the object may be modified without hurting others, throws if this is not the case.

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

Default output to stream.

- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

Tree output to stream.

- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

Python parsable output to stream.

## Protected Attributes

- [ex k](#)
- [ex P](#)
- [ex C\\_norm](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< cln::cl\_N > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 6.91.1 Detailed Description

A kernel corresponding to a polynomial in Eisenstein series.

This class represents the differential one-form

$$\omega^{\text{modular}}(P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)})) = C_k P_k(\eta_{k_1}^{(1)}, \dots, \eta_{k_r}^{(r)}) \frac{d\bar{q}_N}{\bar{q}_N}.$$

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 modular\_form\_kernel()

```
GiNaC::modular_form_kernel::modular_form_kernel (
 const ex & k,
 const ex & P,
 const ex & C_norm = numeric(1))
```

### 6.91.3 Member Function Documentation

#### 6.91.3.1 series()

```
ex GiNaC::modular_form_kernel::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

The series method for this class returns the qbar-expansion of the modular form, without an additional factor of  $C\_norm/qbar$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::lhs\(\)](#), [order](#), [P](#), [GiNaC::pow\(\)](#), [qbar](#), [r](#), and [GiNaC::ex::series\(\)](#).

Referenced by [q\\_expansion\\_modular\\_form\(\)](#).

#### 6.91.3.2 nops()

```
size_t GiNaC::modular_form_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.91.3.3 op()

```
ex GiNaC::modular_form_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position  $i$ .

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [k](#), and [P](#).

#### 6.91.3.4 let\_op()

```
ex & GiNaC::modular_form_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position  $i$ .

Reimplemented from [GiNaC::basic](#).

References [C\\_norm](#), [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [k](#), and [P](#).

### 6.91.3.5 is\_numeric()

```
bool GiNaC::modular_form_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [k](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [GiNaC::series\\_to\\_poly\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 6.91.3.6 Laurent\_series()

```
ex GiNaC::modular_form_kernel::Laurent_series (
 const ex & x,
 int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [order](#), [q\\_expansion\\_modular\\_form\(\)](#), [qbar](#), [GiNaC::ex::series\(\)](#), and [GiNaC::series\\_to\\_poly\(\)](#).

### 6.91.3.7 get\_numerical\_value()

```
ex GiNaC::modular_form_kernel::get_numerical_value (
 const ex & qbar,
 int N_trunc = 0) const [override], [virtual]
```

Returns the value of the modular form.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [C\\_norm](#), [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [qbar](#).

### 6.91.3.8 uses\_Laurent\_series()

```
bool GiNaC::modular_form_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.91.3.9 q\_expansion\_modular\_form()

```
ex GiNaC::modular_form_kernel::q_expansion_modular_form (
 const ex & q,
 int order) const
```

References [series\(\)](#).

Referenced by [is\\_numeric\(\)](#), and [Laurent\\_series\(\)](#).

### 6.91.3.10 do\_print()

```
void GiNaC::modular_form_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [C\\_norm](#), [k](#), [P](#), and [GiNaC::ex::print\(\)](#).

## 6.91.4 Member Data Documentation

### 6.91.4.1 k

```
ex GiNaC::modular_form_kernel::k [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

### 6.91.4.2 P

```
ex GiNaC::modular_form_kernel::P [protected]
```

Referenced by [do\\_print\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\(\)](#).

### 6.91.4.3 C\_norm

```
ex GiNaC::modular_form_kernel::C_norm [protected]
```

Referenced by [do\\_print\(\)](#), [get\\_numerical\\_value\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.92 GiNaC::basic\_partition\_generator::mpartition2 Struct Reference

```
#include <utils.h>
```

## Public Member Functions

- [mpartition2](#) (unsigned n\_, unsigned m\_)
- bool [next\\_partition](#) ()

## Public Attributes

- std::vector< unsigned > [x](#)
- unsigned [n](#)
- unsigned [m](#)

## 6.92.1 Constructor & Destructor Documentation

### 6.92.1.1 mpartition2()

```
GiNaC::basic_partition_generator::mpartition2::mpartition2 (
 unsigned n_,
 unsigned m_) [inline]
```

References [k](#), [m](#), [n](#), and [x](#).

## 6.92.2 Member Function Documentation

### 6.92.2.1 next\_partition()

```
bool GiNaC::basic_partition_generator::mpartition2::next_partition () [inline]
```

References [k](#), [m](#), and [x](#).

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), and [GiNaC::partition\\_generator::next\(\)](#).

## 6.92.3 Member Data Documentation

### 6.92.3.1 x

```
std::vector<unsigned> GiNaC::basic_partition_generator::mpartition2::x
```

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [mpartition2\(\)](#), and [next\\_partition\(\)](#).

### 6.92.3.2 n

```
unsigned GiNaC::basic_partition_generator::mpartition2::n
```

Referenced by [mpartition2\(\)](#), and [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#).

### 6.92.3.3 m

unsigned GiNaC::basic\_partition\_generator::mpartition2::m

Referenced by [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::partition\\_generator::get\(\)](#), [mpartition2\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), and [next\\_partition\(\)](#).

The documentation for this struct was generated from the following file:

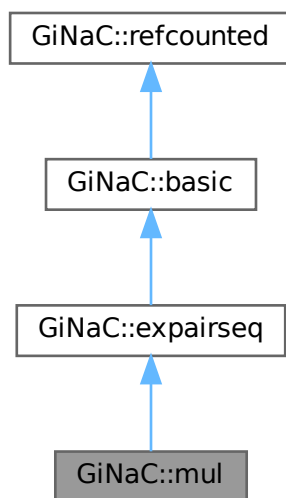
- [utils.h](#)

## 6.93 GiNaC::mul Class Reference

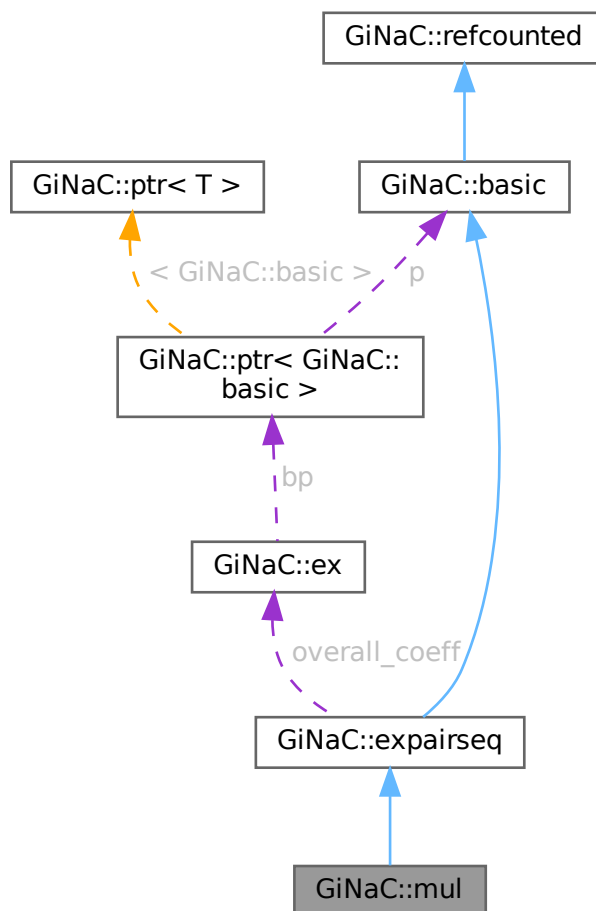
Product of expressions.

```
#include <mul.h>
```

Inheritance diagram for GiNaC::mul:



Collaboration diagram for `GiNaC::mul`:



## Public Member Functions

- `mul` (const `ex` &lh, const `ex` &rh)
- `mul` (const `exvector` &v)
- `mul` (const `epvector` &v)
- `mul` (const `epvector` &v, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (`epvector` &&vp)
- `mul` (`epvector` &&vp, const `ex` &oc, bool do\_index\_renaming=false)
- `mul` (const `ex` &lh, const `ex` &mh, const `ex` &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override

- *Return degree of highest power in object s.*  
• `int ldegree` (const `ex` &s) const override
- *Return degree of lowest power in object s.*  
• `ex coeff` (const `ex` &s, int `n=1`) const override
- *Return coefficient of degree n in object s.*  
• `bool has` (const `ex` &other, unsigned `options=0`) const override
- *Test for occurrence of a pattern.*  
• `ex eval` () const override
- *Perform automatic term rewriting rules in this class.*  
• `ex evalf` () const override
- *Evaluate object numerically.*  
• `ex real_part` () const override
- `ex imag_part` () const override
- `ex evalm` () const override
- *Evaluate sums, products and integer powers of matrices.*  
• `ex series` (const `relational` &s, int `order`, unsigned `options=0`) const override
- *Implementation of `ex::series()` for product.*  
• `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override
- *Implementation of `ex::normal()` for a product.*  
• `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override
- *Apply symmetric modular homomorphism to an expanded multivariate polynomial.*  
• `numeric max_coefficient` () const override
- *Implementation `ex::max_coefficient()`.*  
• `exvector get_free_indices` () const override
- *Return a vector containing the free indices of an expression.*  
• `ex conjugate` () const override
- `ex algebraic_subs_mul` (const `exmap` &m, unsigned `options`) const

## Public Member Functions inherited from `GiNaC::expairseq`

- `expairseq` (const `ex` &lh, const `ex` &rh)
- `expairseq` (const `exvector` &v)
- `expairseq` (const `epvector` &v, const `ex` &oc, bool `do_index_renaming=false`)
- `expairseq` (`epvector` &&vp, const `ex` &oc, bool `do_index_renaming=false`)
- `size_t nops` () const override
- *Number of operands/members.*  
• `ex op` (size\_t `i`) const override
- *Return operand/member at position i.*  
• `ex map` (`map_function` &f) const override
- *Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*  
• `ex to_rational` (`exmap` &repl) const override
- *Implementation of `ex::to_rational()` for expairseqs.*  
• `ex to_polynomial` (`exmap` &repl) const override
- *Implementation of `ex::to_polynomial()` for expairseqs.*  
• `bool match` (const `ex` &pattern, `exmap` &repl\_lst) const override
- *Check whether the expression matches a given pattern.*  
• `ex subs` (const `exmap` &m, unsigned `options=0`) const override
- *Substitute a set of objects by arbitrary expressions.*  
• `void archive` (`archive_node` &n) const override
- *Save (serialize) the object into archive node.*  
• `void read_archive` (const `archive_node` &n, `lst` &syms) override
- *Load (deserialize) the object from an archive node.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around `print` to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for a product.*
- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- [ex thisexpairseq](#) (const [epvector](#) &v, const [ex](#) &oc, bool do\_index\_renaming=false) const override  
*Create an object of this type.*
- [ex thisexpairseq](#) ([epvector](#) &&vp, const [ex](#) &oc, bool do\_index\_renaming=false) const override
- [expair split\\_ex\\_to\\_pair](#) (const [ex](#) &e) const override  
*Form an expair from an ex, using the corresponding semantics.*
- [expair combine\\_ex\\_with\\_coeff\\_to\\_pair](#) (const [ex](#) &e, const [ex](#) &c) const override
- [expair combine\\_pair\\_with\\_coeff\\_to\\_pair](#) (const [expair](#) &p, const [ex](#) &c) const override
- [ex recombine\\_pair\\_to\\_ex](#) (const [expair](#) &p) const override  
*Form an ex out of an expair, using the corresponding semantics.*
- bool [expair\\_needs\\_further\\_processing](#) ([ep](#) it) override
- [ex default\\_overall\\_coeff](#) () const override
- void [combine\\_overall\\_coeff](#) (const [ex](#) &c) override
- void [combine\\_overall\\_coeff](#) (const [ex](#) &c1, const [ex](#) &c2) override
- bool [can\\_make\\_flat](#) (const [expair](#) &p) const override
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- void [find\\_real\\_imag](#) ([ex](#) &, [ex](#) &) const
- void [print\\_overall\\_coeff](#) (const [print\\_context](#) &c, const char \*mul\_sym) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [epvector](#) [expandchildren](#) (unsigned [options](#)) const  
*Member-wise expand the expairs representing this sequence.*

## Protected Member Functions inherited from GiNaC::expairseq

- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- virtual void [printseq](#) (const [print\\_context](#) &c, char delim, unsigned this\_precedence, unsigned upper\_precedence) const
- virtual void [printpair](#) (const [print\\_context](#) &c, const [expair](#) &p, unsigned upper\_precedence) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [construct\\_from\\_2\\_ex](#) (const [ex](#) &lh, const [ex](#) &rh)

- void `construct_from_2_expireseq` (const `expireseq` &s1, const `expireseq` &s2)
- void `construct_from_expireseq_ex` (const `expireseq` &s, const `ex` &e)
- void `construct_from_exvector` (const `exvector` &v)
- void `construct_from_epvector` (const `epvector` &v, bool do\_index\_renaming=false)
- void `construct_from_epvector` (`epvector` &&v, bool do\_index\_renaming=false)
- void `make_flat` (const `exvector` &v)  
*Combine this expireseq with argument exvector.*
- void `make_flat` (const `epvector` &v, bool do\_index\_renaming=false)  
*Combine this expireseq with argument epvector.*
- void `canonicalize` ()  
*Brings this expireseq into a sorted (canonical) form.*
- void `combine_same_terms_sorted_seq` ()  
*Compact a presorted expireseq by combining all matching expires to one each.*
- bool `is_canonical` () const  
*Check if this expireseq is in sorted (canonical) form.*
- `epvector` `expandchildren` (unsigned `options`) const  
*Member-wise expand the expires in this sequence.*
- `epvector` `evalchildren` () const  
*Member-wise evaluate the expires in this sequence.*
- `epvector` `subchildren` (const `exmap` &m, unsigned `options`=0) const  
*Member-wise substitute in this sequence.*

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Static Protected Member Functions

- static bool `can_be_further_expanded` (const `ex` &e)

## Friends

- class `add`
- class `ncmul`
- class `power`

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::expairseq](#)

- [epvector seq](#)
- [ex overall\\_coeff](#)

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.93.1 Detailed Description

Product of expressions.

## 6.93.2 Constructor & Destructor Documentation

### 6.93.2.1 `mul()` [1/7]

```
GiNaC::mul::mul (
 const ex & lh,
 const ex & rh)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_2\\_ex\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

Referenced by [do\\_print\\_latex\(\)](#), and [expand\(\)](#).

### 6.93.2.2 `mul()` [2/7]

```
GiNaC::mul::mul (
 const exvector & v)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

### 6.93.2.3 `mul()` [3/7]

```
GiNaC::mul::mul (
 const epvector & v)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.4 mul() [4/7]**

```
GiNaC::mul::mul (
 const epvector & v,
 const ex & oc,
 bool do_index_renaming = false)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.5 mul() [5/7]**

```
GiNaC::mul::mul (
 epvector && vp)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.6 mul() [6/7]**

```
GiNaC::mul::mul (
 epvector && vp,
 const ex & oc,
 bool do_index_renaming = false)
```

References [GiNaC::expairseq::construct\\_from\\_epvector\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.2.7 mul() [7/7]**

```
GiNaC::mul::mul (
 const ex & lh,
 const ex & mh,
 const ex & rh)
```

References [GiNaC::\\_ex1](#), [GiNaC::expairseq::construct\\_from\\_exvector\(\)](#), [factors](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::is\\_canonical\(\)](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.3 Member Function Documentation****6.93.3.1 precedence()**

```
unsigned GiNaC::mul::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::expairseq](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_cscc\(\)](#), [do\\_print\\_latex\(\)](#), and [print\\_overall\\_coeff\(\)](#).

### 6.93.3.2 info()

```
bool GiNaC::mul::info (
 unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_num1\\_p](#), [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::factor\(\)](#), [GiNaC::basic::flags](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::status\\_flags::is\\_negative](#), [GiNaC::status\\_flags::is\\_positive](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [expand\(\)](#).

### 6.93.3.3 is\_polynomial()

```
bool GiNaC::mul::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::nonnegint](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.4 degree()

```
int GiNaC::mul::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::degree\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.5 ldegree()

```
int GiNaC::mul::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::ldegree\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.6 `coeff()`

```
ex GiNaC::mul::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [c](#), [GiNaC::ex::coeff\(\)](#), [n](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [print\\_overall\\_coeff\(\)](#).

### 6.93.3.7 `has()`

```
bool GiNaC::mul::has (
 const ex & pattern,
 unsigned options = 0) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [options](#).

### 6.93.3.8 `eval()`

```
ex GiNaC::mul::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x, x_1, x_2, \dots$  stand for a symbolic variables of type `ex` and  $c, c_1, c_2, \dots$  stand for such expressions that contain a plain number.

- $*(\dots, x; 0) \rightarrow 0$
- $*(+(x_1, x_2, \dots); c) \rightarrow *(*(x_1, c), *(x_2, c), \dots)$
- $*(x; 1) \rightarrow x$
- $*(; c) \rightarrow c$

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_c](#), [GiNaC::basic::hold\(\)](#), [GiNaC::numeric::integer\\_content\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [last](#), [likely](#), [GiNaC::numeric::mul\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [unlikely](#).

### 6.93.3.9 evalf()

```
ex GiNaC::mul::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.10 real\_part()

```
ex GiNaC::mul::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).

### 6.93.3.11 imag\_part()

```
ex GiNaC::mul::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [find\\_real\\_imag\(\)](#).

### 6.93.3.12 evalm()

```
ex GiNaC::mul::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::evalm\(\)](#), [m](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.93.3.13 series()

```
ex GiNaC::mul::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for product.

This performs series multiplication when multiplying series.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::ldegree\(\)](#), [GiNaC::ex::lhs\(\)](#), [GiNaC::pseries::mul\\_const\(\)](#), [GiNaC::pseries::mul\\_series\(\)](#), [GiNaC::expairseq::op\(\)](#), [options](#), [order](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::ex::rhs\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::ex::series\(\)](#).

### 6.93.3.14 normal()

```
ex GiNaC::mul::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a product.

It cancels common factors from fractions.

See also

[ex::normal\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::frac\\_cancel\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.15 integer\_content()

```
numeric GiNaC::mul::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

### 6.93.3.16 smod()

```
ex GiNaC::mul::smod (
 const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

Parameters

|      |         |
|------|---------|
| $xi$ | modulus |
|------|---------|

Returns

mapped polynomial

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [GiNaC::smod\(\)](#).

**6.93.3.17 max\_coefficient()**

```
numeric GiNaC::mul::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

**6.93.3.18 get\_free\_indices()**

```
exvector GiNaC::mul::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::expairseq::nops\(\)](#), and [GiNaC::expairseq::op\(\)](#).

**6.93.3.19 conjugate()**

```
ex GiNaC::mul::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [c](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), [thisexpairseq\(\)](#), and [x](#).

**6.93.3.20 derivative()**

```
ex GiNaC::mul::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::diff\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [GiNaC::expairseq::seq](#), [split\\_ex\\_to\\_pair\(\)](#), and [GiNaC::expair::swap\(\)](#).

**6.93.3.21 eval\_ncmul()**

```
ex GiNaC::mul::eval_ncmul (
 const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**6.93.3.22 return\_type()**

```
unsigned GiNaC::mul::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_comp](#) and [GiNaC::expairseq::seq](#).

**6.93.3.23 return\_type\_tinfo()**

```
return_type_t GiNaC::mul::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::expairseq::seq](#).

**6.93.3.24 thisexpairseq() [1/2]**

```
ex GiNaC::mul::thisexpairseq (
 const epvector & v,
 const ex & oc,
 bool do_index_renaming = false) const [override], [protected], [virtual]
```

Create an object of this type.

This method works similar to a constructor. It is useful because expairseq has (at least) two possible different semantics but we want to inherit methods thus avoiding code duplication. Sometimes a method in expairseq has to create a new one of the same semantics, which cannot be done by a ctor because the name (add, mul,...) is unknown on the expairseq level. In order for this trick to work a derived class must of course override this definition.

Reimplemented from [GiNaC::expairseq](#).

Referenced by [conjugate\(\)](#).

**6.93.3.25 thisexpairseq() [2/2]**

```
ex GiNaC::mul::thisexpairseq (
 epvector && vp,
 const ex & oc,
 bool do_index_renaming = false) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

**6.93.3.26 split\_ex\_to\_pair()**

```
expair GiNaC::mul::split_ex_to_pair (
 const ex & e) const [override], [protected], [virtual]
```

Form an expair from an ex, using the corresponding semantics.

See also

[expairseq::recombine\\_pair\\_to\\_ex\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::power::basis](#), and [GiNaC::power::exponent](#).

Referenced by [combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [derivative\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expand\(\)](#), and [expandchildren\(\)](#).

**6.93.3.27 combine\_ex\_with\_coeff\_to\_pair()**

```
expair GiNaC::mul::combine_ex_with_coeff_to_pair (
 const ex & e,
 const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GINAC\\_ASSERT](#), [GiNaC::pow\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.93.3.28 combine\_pair\_with\_coeff\_to\_pair()**

```
expair GiNaC::mul::combine_pair_with_coeff_to_pair (
 const expair & p,
 const ex & c) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::pow\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expair::rest](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.93.3.29 recombine\_pair\_to\_ex()**

```
ex GiNaC::mul::recombine_pair_to_ex (
 const expair & p) const [override], [protected], [virtual]
```

Form an ex out of an expair, using the corresponding semantics.

See also

[expairseq::split\\_ex\\_to\\_pair\(\)](#)

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::expair::rest](#).

Referenced by [coeff\(\)](#), [combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [eval\(\)](#), [evalm\(\)](#), [expair\\_needs\\_further\\_processing\(\)](#), [expandchildren\(\)](#), [find\\_real\\_imag\(\)](#), [info\(\)](#), [integer\\_content\(\)](#), [ldegree\(\)](#), [max\\_coefficient\(\)](#), [normal\(\)](#), [series\(\)](#), and [smod\(\)](#).

**6.93.3.30 expair\_needs\_further\_processing()**

```
bool GiNaC::mul::expair_needs_further_processing (
 exp it) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::expair::is\\_equal\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

**6.93.3.31 default\_overall\_coeff()**

```
ex GiNaC::mul::default_overall_coeff () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#).

**6.93.3.32 combine\_overall\_coeff() [1/2]**

```
void GiNaC::mul::combine_overall_coeff (
 const ex & c) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [c](#), [GINAC\\_ASSERT](#), and [GiNaC::expairseq::overall\\_coeff](#).

**6.93.3.33 combine\_overall\_coeff() [2/2]**

```
void GiNaC::mul::combine_overall_coeff (
 const ex & c1,
 const ex & c2) [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GINAC\\_ASSERT](#), [GiNaC::expairseq::overall\\_coeff](#), and [power](#).

**6.93.3.34 can\_make\_flat()**

```
bool GiNaC::mul::can_make_flat (
 const expair & p) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::expair::coeff](#), [GINAC\\_ASSERT](#), [GiNaC::ex::info\(\)](#), and [GiNaC::info\\_flags::integer](#).

### 6.93.3.35 expand()

```
ex GiNaC::mul::expand (
 unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::expairseq](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_num0\\_p](#), [can\\_be\\_further\\_expanded\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::expand\\_options::expand\\_ren](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [factors](#), [GiNaC::get\\_all\\_dummy\\_indices\\_safely\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [mul\(\)](#), [GiNaC::numeric::mul\(\)](#), [n](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [split\\_ex\\_to\\_pair\(\)](#).

### 6.93.3.36 algebraic\_subs\_mul()

```
ex GiNaC::mul::algebraic_subs_mul (
 const exmap & m,
 unsigned options) const
```

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), [options](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

### 6.93.3.37 find\_real\_imag()

```
void GiNaC::mul::find_real_imag (
 ex & rp,
 ex & ip) const [protected]
```

References [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::ex::real\\_part\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

Referenced by [imag\\_part\(\)](#), and [real\\_part\(\)](#).

### 6.93.3.38 print\_overall\_coeff()

```
void GiNaC::mul::print_overall_coeff (
 const print_context & c,
 const char * mul_sym) const [protected]
```

References [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [c](#), [coeff\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), and [GiNaC::ex::print\(\)](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_latex\(\)](#).

**6.93.3.39 do\_print()**

```
void GiNaC::mul::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

**6.93.3.40 do\_print\_latex()**

```
void GiNaC::mul::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), [GINAC\\_ASSERT](#), [mul\(\)](#), [precedence\(\)](#), [print\\_overall\\_coeff\(\)](#), [recombine\\_pair\\_to\\_ex\(\)](#), and [GiNaC::expairseq::seq](#).

**6.93.3.41 do\_print\_csrc()**

```
void GiNaC::mul::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [c](#), [GiNaC::basic::ex](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::expairseq::overall\\_coeff](#), [power](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), and [GiNaC::expairseq::seq](#).

**6.93.3.42 do\_print\_python\_repr()**

```
void GiNaC::mul::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::expairseq::nops\(\)](#), [GiNaC::expairseq::op\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.93.3.43 can\_be\_further\_expanded()**

```
bool GiNaC::mul::can_be_further_expanded (
 const ex & e) [static], [protected]
```

References [GiNaC::ex::info\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::info\\_flags::posint](#).

Referenced by [expand\(\)](#).

#### 6.93.3.44 expandchildren()

```
epvector GiNaC::mul::expandchildren (
 unsigned options) const [protected]
```

Member-wise expand the expairs representing this sequence.

This must be overridden from [expairseq::expandchildren\(\)](#) and done iteratively in order to allow for early cancellations and thus save memory.

See also

[mul::expand\(\)](#)

Returns

epvector containing expanded pairs, empty if no members had to be changed.

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::factor\(\)](#), [last](#), [options](#), [recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), and [split\\_ex\\_to\\_pair\(\)](#).

Referenced by [expand\(\)](#).

### 6.93.4 Friends And Related Symbol Documentation

#### 6.93.4.1 add

```
friend class add [friend]
```

#### 6.93.4.2 ncmul

```
friend class ncmul [friend]
```

#### 6.93.4.3 power

```
friend class power [friend]
```

Referenced by [combine\\_overall\\_coeff\(\)](#), and [do\\_print\\_csrc\(\)](#).

The documentation for this class was generated from the following files:

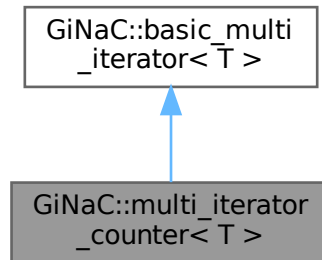
- [mul.h](#)
- [indexed.cpp](#)
- [mul.cpp](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.94 GiNaC::multi\_iterator\_counter< T > Class Template Reference

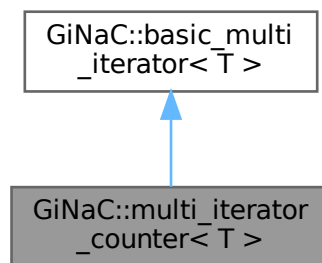
The class `multi_iterator_counter` defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_counter< T >`:



Collaboration diagram for `GiNaC::multi_iterator_counter< T >`:



### Public Member Functions

- `multi_iterator_counter` (void)  
*Default constructor.*
- `multi_iterator_counter` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_counter` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

**Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit  $N$ , lower limit  $B$  and size  $k$ .*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &v)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > &[get\\_vector](#) (void) const  
*Returns a reference to the vector  $v$ .*
- T [operator\[\]](#) (size\_t [i](#)) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t [i](#))  
*Subscription via [].*
- T [operator\(\)](#) (size\_t [i](#)) const  
*Subscription via ()*
- T & [operator\(\)](#) (size\_t [i](#))  
*Subscription via ()*

**Friends**

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_counter](#)< TT > &v)

**Additional Inherited Members****Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

**6.94.1 Detailed Description**

```
template<class T>
class GiNaC::multi_iterator_counter< T >
```

The class [multi\\_iterator\\_counter](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

## 6.94.2 Constructor & Destructor Documentation

### 6.94.2.1 multi\_iterator\_counter() [1/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
 void) [inline]
```

Default constructor.

### 6.94.2.2 multi\_iterator\_counter() [2/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
 T B,
 T N,
 size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.94.2.3 multi\_iterator\_counter() [3/3]

```
template<class T >
GiNaC::multi_iterator_counter< T >::multi_iterator_counter (
 T B,
 T N,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

## 6.94.3 Member Function Documentation

### 6.94.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.94.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.94.4 Friends And Related Symbol Documentation

### 6.94.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_counter< TT > & v) [friend]
```

The documentation for this class was generated from the following file:

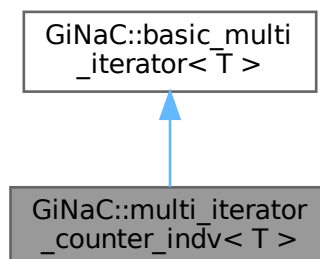
- [utils\\_multi\\_iterator.h](#)

## 6.95 GiNaC::multi\_iterator\_counter\_indv< T > Class Template Reference

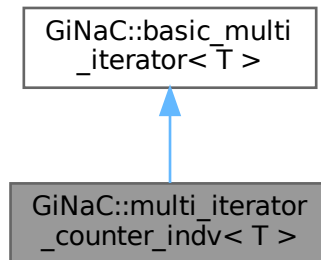
The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_counter\_indv< T >:



Collaboration diagram for `GiNaC::multi_iterator_counter_indv< T >`:



### Public Member Functions

- `multi_iterator_counter_indv` (void)  
*Default constructor.*
- `multi_iterator_counter_indv` (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- `multi_iterator_counter_indv` (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Public Member Functions inherited from `GiNaC::basic_multi_iterator< T >`

- `basic_multi_iterator` (void)  
*Default constructor.*
- `basic_multi_iterator` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- `basic_multi_iterator` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual `~basic_multi_iterator` ()  
*Destructor.*
- size\_t `size` (void) const  
*Returns the size of a multi\_iterator.*
- bool `overflow` (void) const  
*Return the overflow flag.*
- const std::vector< T > &`get_vector` (void) const  
*Returns a reference to the vector v.*
- T `operator[]` (size\_t i) const  
*Subscription via [].*
- T &`operator[]` (size\_t i)  
*Subscription via [].*
- T `operator()` (size\_t i) const  
*Subscription via ().*
- T &`operator()` (size\_t i)  
*Subscription via ().*

**Protected Attributes**

- `std::vector< T > Nv`

**Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

**Friends**

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_counter\_indv< TT > &v)`

**6.95.1 Detailed Description**

`template<class T>`  
**class** [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#)

The class [multi\\_iterator\\_counter\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

**6.95.2 Constructor & Destructor Documentation****6.95.2.1 [multi\\_iterator\\_counter\\_indv\(\)](#) [1/3]**

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
 void) [inline]
```

Default constructor.

**6.95.2.2 [multi\\_iterator\\_counter\\_indv\(\)](#) [2/3]**

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
 T B,
 const std::vector< T > & Nv,
 size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.95.2.3 multi\_iterator\_counter\_indv() [3/3]

```
template<class T >
GiNaC::multi_iterator_counter_indv< T >::multi_iterator_counter_indv (
 T B,
 const std::vector< T > & Nv,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

## 6.95.3 Member Function Documentation

### 6.95.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.95.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_counter_indv< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.95.4 Friends And Related Symbol Documentation

### 6.95.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_counter_indv< TT > & v) [friend]
```

### 6.95.5 Member Data Documentation

#### 6.95.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_counter_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

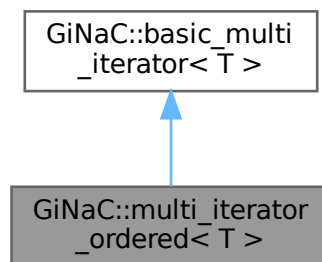
- [utils\\_multi\\_iterator.h](#)

## 6.96 GiNaC::multi\_iterator\_ordered< T > Class Template Reference

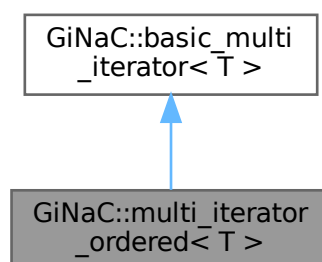
The class [multi\\_iterator\\_ordered](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_ordered< T >:



Collaboration diagram for GiNaC::multi\_iterator\_ordered< T >:



## Public Member Functions

- [multi\\_iterator\\_ordered](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > &init (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > &operator++ (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > &[get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T &[operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T &[operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

### 6.96.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_ordered< T >
```

The class `multi_iterator_ordered` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j < i_{j+1}.$$

It is assumed that  $k > 0$  and  $N - B \geq k$ .

### 6.96.2 Constructor & Destructor Documentation

#### 6.96.2.1 multi\_iterator\_ordered() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
 void) [inline]
```

Default constructor.

#### 6.96.2.2 multi\_iterator\_ordered() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
 T B,
 T N,
 size_t k) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit N and size k .

#### 6.96.2.3 multi\_iterator\_ordered() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered< T >::multi_iterator_ordered (
 T B,
 T N,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

### 6.96.3 Member Function Documentation

#### 6.96.3.1 `init()`

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

#### 6.96.3.2 `operator++()`

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

### 6.96.4 Friends And Related Symbol Documentation

#### 6.96.4.1 `operator<<`

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_ordered< TT > & v) [friend]
```

The documentation for this class was generated from the following file:

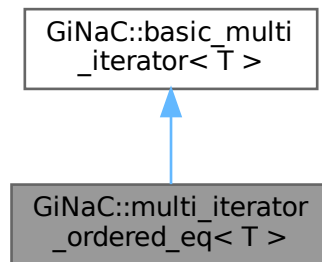
- [utils\\_multi\\_iterator.h](#)

## 6.97 GiNaC::multi\_iterator\_ordered\_eq< T > Class Template Reference

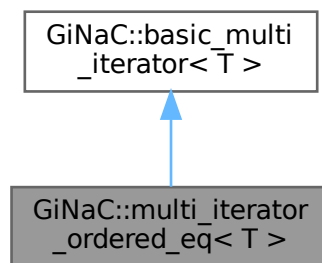
The class `multi_iterator_ordered_eq` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for `GiNaC::multi_iterator_ordered_eq< T >`:



Collaboration diagram for `GiNaC::multi_iterator_ordered_eq< T >`:



### Public Member Functions

- `multi_iterator_ordered_eq` (void)  
*Default constructor.*
- `multi_iterator_ordered_eq` (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k.*
- `multi_iterator_ordered_eq` (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- `basic_multi_iterator< T > &init` (void)  
*Initialize the multi-index to.*
- `basic_multi_iterator< T > &operator++` (int)  
*The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.*

## Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Friends

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_ordered\\_eq](#)< TT > &v)

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- T [N](#)
- T [B](#)
- std::vector< T > [v](#)
- bool [flag\\_overflow](#)

### 6.97.1 Detailed Description

template<class T>  
class [GiNaC::multi\\_iterator\\_ordered\\_eq](#)< T >

The class [multi\\_iterator\\_ordered\\_eq](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N$$

and

$$i_j \leq i_{j+1}.$$

It is assumed that  $k > 0$ .

## 6.97.2 Constructor & Destructor Documentation

### 6.97.2.1 multi\_iterator\_ordered\_eq() [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
 void) [inline]
```

Default constructor.

### 6.97.2.2 multi\_iterator\_ordered\_eq() [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
 T B,
 T N,
 size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

### 6.97.2.3 multi\_iterator\_ordered\_eq() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq< T >::multi_iterator_ordered_eq (
 T B,
 T N,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

## 6.97.3 Member Function Documentation

### 6.97.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, \dots, n_k) = (B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.97.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.97.4 Friends And Related Symbol Documentation

### 6.97.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_ordered_eq< TT > & v) [friend]
```

The documentation for this class was generated from the following file:

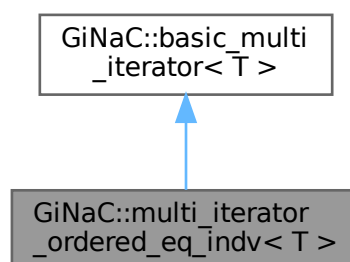
- [utils\\_multi\\_iterator.h](#)

## 6.98 GiNaC::multi\_iterator\_ordered\_eq\_indv< T > Class Template Reference

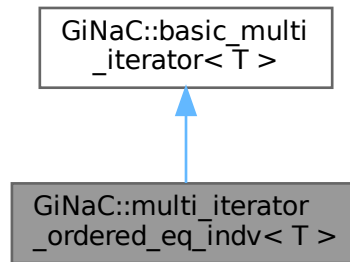
The class [multi\\_iterator\\_ordered\\_eq\\_indv](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#):



Collaboration diagram for GiNaC::multi\_iterator\_ordered\_eq\_indv< T >:



### Public Member Functions

- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_ordered\\_eq\\_indv](#) (T B, const std::vector< T > &Nv, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator< T > & init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator< T > & operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Protected Attributes

- `std::vector< T > Nv`

## Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered_eq_indv< TT > &v)`

## 6.98.1 Detailed Description

`template<class T>`  
`class GiNaC::multi_iterator_ordered_eq_indv< T >`

The class `multi_iterator_ordered_eq_indv` defines a `multi_iterator`  $(i_1, i_2, \dots, i_k)$ , such that.

$$B \leq i_j < N_j$$

and

$$i_j \leq i_{j+1}.$$

## 6.98.2 Constructor & Destructor Documentation

### 6.98.2.1 `multi_iterator_ordered_eq_indv()` [1/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
 void) [inline]
```

Default constructor.

### 6.98.2.2 `multi_iterator_ordered_eq_indv()` [2/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
 T B,
 const std::vector< T > & Nv,
 size_t k) [inline], [explicit]
```

Construct a `multi_iterator` with upper limit `N` and size `k`.

### 6.98.2.3 multi\_iterator\_ordered\_eq\_indv() [3/3]

```
template<class T >
GiNaC::multi_iterator_ordered_eq_indv< T >::multi_iterator_ordered_eq_indv (
 T B,
 const std::vector< T > & Nv,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

## 6.98.3 Member Function Documentation

### 6.98.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B, B, B, \dots, B)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

### 6.98.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_ordered_eq_indv< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

## 6.98.4 Friends And Related Symbol Documentation

### 6.98.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_ordered_eq_indv< TT > & v) [friend]
```

## 6.98.5 Member Data Documentation

### 6.98.5.1 Nv

```
template<class T >
std::vector<T> GiNaC::multi_iterator_ordered_eq_indv< T >::Nv [protected]
```

The documentation for this class was generated from the following file:

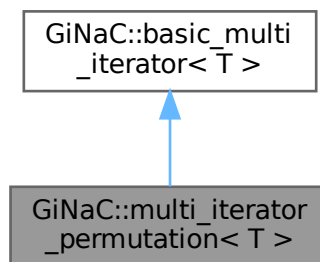
- [utils\\_multi\\_iterator.h](#)

## 6.99 GiNaC::multi\_iterator\_permutation< T > Class Template Reference

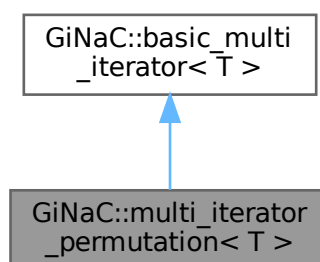
The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_permutation< T >:



Collaboration diagram for GiNaC::multi\_iterator\_permutation< T >:



**Public Member Functions**

- [multi\\_iterator\\_permutation](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_permutation](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N and size k .*
- [multi\\_iterator\\_permutation](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index n++, which will update n to the next configuration.*
- int [get\\_sign](#) (void) const  
*Returns the sign of the permutation, defined by.*

**Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

**Friends**

- template<class TT >  
std::ostream & [operator<<](#) (std::ostream &os, const [multi\\_iterator\\_permutation](#)< TT > &v)

**Additional Inherited Members****Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)**

- T N
- T B
- std::vector< T > v
- bool [flag\\_overflow](#)

### 6.99.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_permutation< T >
```

The class [multi\\_iterator\\_permutation](#) defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.

$$B \leq i_j < N$$

and

$$i_i \neq i_j$$

In particular, if  $N - B = k$ , [multi\\_iterator\\_permutation](#) loops over all permutations of  $k$  elements.

### 6.99.2 Constructor & Destructor Documentation

#### 6.99.2.1 multi\_iterator\_permutation() [1/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
 void) [inline]
```

Default constructor.

#### 6.99.2.2 multi\_iterator\_permutation() [2/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
 T B,
 T N,
 size_t k) [inline], [explicit]
```

Construct a multi\_iterator with upper limit N and size k .

#### 6.99.2.3 multi\_iterator\_permutation() [3/3]

```
template<class T >
GiNaC::multi_iterator_permutation< T >::multi_iterator_permutation (
 T B,
 T N,
 const std::vector< T > & vv) [inline], [explicit]
```

Construct from a vector.

### 6.99.3 Member Function Documentation

#### 6.99.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to.

$$(n_1, n_2, n_3, \dots, n_k) = (B + 0, B + 1, B + 2, \dots, B + k - 1)$$

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

#### 6.99.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_permutation< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.

If  $n$  is in the last configuration and the increment operator  $++$  is applied to  $n$ , the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#).

#### 6.99.3.3 get\_sign()

```
template<class T >
int GiNaC::multi_iterator_permutation< T >::get_sign (
 void) const [inline]
```

Returns the sign of the permutation, defined by.

$$(-1)^{n_{inv}},$$

where  $n_{inv}$  is the number of inversions, e.g. the number of pairs  $i < j$  for which

$$n_i > n_j.$$

References [k](#).

## 6.99.4 Friends And Related Symbol Documentation

### 6.99.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_permutation< TT > & v) [friend]
```

The documentation for this class was generated from the following file:

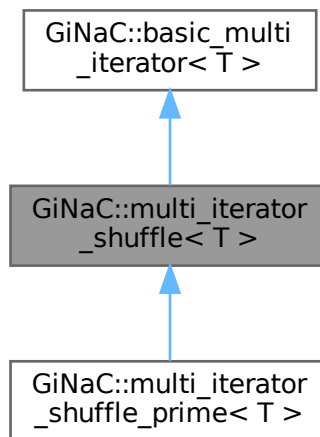
- [utils\\_multi\\_iterator.h](#)

## 6.100 GiNaC::multi\_iterator\_shuffle< T > Class Template Reference

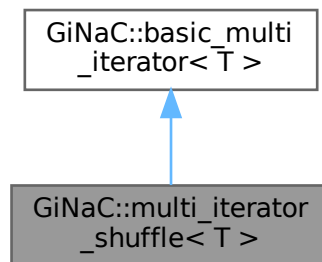
The class [multi\\_iterator\\_shuffle](#) defines a multi\_iterator, which runs over all shuffles of a and b.

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle< T >:



Collaboration diagram for GiNaC::multi\_iterator\_shuffle< T >:



### Public Member Functions

- [multi\\_iterator\\_shuffle](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to the first shuffle.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index  $n++$ , which will update  $n$  to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T B, T N, size\_t k)  
*Construct a multi\_iterator with upper limit N, lower limit B and size k .*
- [basic\\_multi\\_iterator](#) (T B, T N, const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()  
*Destructor.*
- size\_t [size](#) (void) const  
*Returns the size of a multi\_iterator.*
- bool [overflow](#) (void) const  
*Return the overflow flag.*
- const std::vector< T > & [get\\_vector](#) (void) const  
*Returns a reference to the vector v.*
- T [operator\[\]](#) (size\_t i) const  
*Subscription via [].*
- T & [operator\[\]](#) (size\_t i)  
*Subscription via [].*
- T [operator\(\)](#) (size\_t i) const  
*Subscription via ().*
- T & [operator\(\)](#) (size\_t i)  
*Subscription via ().*

## Protected Attributes

- `size_t` [N\\_internal](#)
- `std::vector< size_t >` [v\\_internal](#)
- `std::vector< T >` [v\\_orig](#)

## Protected Attributes inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- `T` [N](#)
- `T` [B](#)
- `std::vector< T >` [v](#)
- `bool` [flag\\_overflow](#)

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi\_iterator\_shuffle< TT > &v)`

## 6.100.1 Detailed Description

`template<class T>`  
**class** [GiNaC::multi\\_iterator\\_shuffle< T >](#)

The class [multi\\_iterator\\_shuffle](#) defines a `multi_iterator`, which runs over all shuffles of `a` and `b`.

## 6.100.2 Constructor & Destructor Documentation

### 6.100.2.1 [multi\\_iterator\\_shuffle\(\)](#) [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
 void) [inline]
```

Default constructor.

### 6.100.2.2 [multi\\_iterator\\_shuffle\(\)](#) [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle< T >::multi_iterator_shuffle (
 const std::vector< T > & a,
 const std::vector< T > & b) [inline], [explicit]
```

Construct from a vector.

References [GiNaC::basic\\_multi\\_iterator< T >::B](#), [GiNaC::multi\\_iterator\\_shuffle< T >::N\\_internal](#), [GiNaC::basic\\_multi\\_iterator< T >](#), [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_internal](#), and [GiNaC::multi\\_iterator\\_shuffle< T >::v\\_orig](#).

## 6.100.3 Member Function Documentation

### 6.100.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

Reimplemented in [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#).

### 6.100.3.2 operator++()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle< T >::operator++ (
 int) [inline], [virtual]
```

The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.

If `n` is in the last configuration and the increment operator `++` is applied to `n`, the overflow flag will be raised.

Reimplemented from [GiNaC::basic\\_multi\\_iterator< T >](#).

References [k](#), and [GiNaC::basic\\_multi\\_iterator< T >::size\(\)](#).

## 6.100.4 Friends And Related Symbol Documentation

### 6.100.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_shuffle< TT > & v) [friend]
```

## 6.100.5 Member Data Documentation

### 6.100.5.1 N\_internal

```
template<class T >
size_t GiNaC::multi_iterator_shuffle< T >::N_internal [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

### 6.100.5.2 v\_internal

```
template<class T >
std::vector<size_t> GiNaC::multi_iterator_shuffle< T >::v_internal [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

### 6.100.5.3 v\_orig

```
template<class T >
std::vector<T> GiNaC::multi_iterator_shuffle< T >::v_orig [protected]
```

Referenced by [GiNaC::multi\\_iterator\\_shuffle< T >::multi\\_iterator\\_shuffle\(\)](#).

The documentation for this class was generated from the following file:

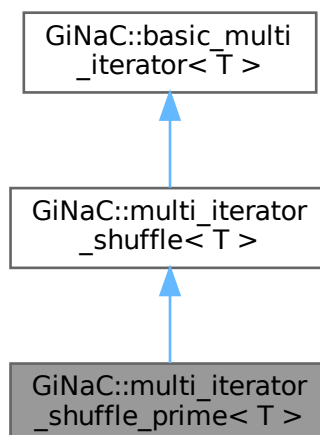
- [utils\\_multi\\_iterator.h](#)

## 6.101 GiNaC::multi\_iterator\_shuffle\_prime< T > Class Template Reference

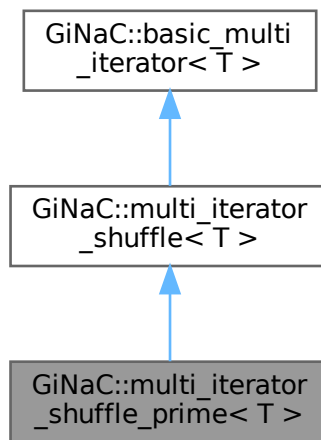
The class [multi\\_iterator\\_shuffle\\_prime](#) defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

```
#include <utils_multi_iterator.h>
```

Inheritance diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



Collaboration diagram for GiNaC::multi\_iterator\_shuffle\_prime< T >:



### Public Member Functions

- [multi\\_iterator\\_shuffle\\_prime](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle\\_prime](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [init](#) (void)  
*Initialize the multi-index to the first shuffle.*

### Public Member Functions inherited from [GiNaC::multi\\_iterator\\_shuffle< T >](#)

- [multi\\_iterator\\_shuffle](#) (void)  
*Default constructor.*
- [multi\\_iterator\\_shuffle](#) (const std::vector< T > &a, const std::vector< T > &b)  
*Construct from a vector.*
- [basic\\_multi\\_iterator](#)< T > & [operator++](#) (int)  
*The postfix increment operator allows to write for a multi-index `n++`, which will update `n` to the next configuration.*

### Public Member Functions inherited from [GiNaC::basic\\_multi\\_iterator< T >](#)

- [basic\\_multi\\_iterator](#) (void)  
*Default constructor.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), size\_t [k](#))  
*Construct a multi\_iterator with upper limit `N`, lower limit `B` and size `k`.*
- [basic\\_multi\\_iterator](#) (T [B](#), T [N](#), const std::vector< T > &vv)  
*Construct from a vector.*
- virtual [~basic\\_multi\\_iterator](#) ()

*Destructor.*

- `size_t size` (void) const  
*Returns the size of a multi\_iterator.*
- `bool overflow` (void) const  
*Return the overflow flag.*
- `const std::vector< T > & get_vector` (void) const  
*Returns a reference to the vector v.*
- `T operator[]` (size\_t i) const  
*Subscription via [].*
- `T & operator[]` (size\_t i)  
*Subscription via [].*
- `T operator()` (size\_t i) const  
*Subscription via ().*
- `T & operator()` (size\_t i)  
*Subscription via ().*

## Friends

- `template<class TT >`  
`std::ostream & operator<< (std::ostream &os, const multi_iterator_shuffle_prime< TT > &v)`

## Additional Inherited Members

### Protected Attributes inherited from `GiNaC::multi_iterator_shuffle< T >`

- `size_t N_internal`
- `std::vector< size_t > v_internal`
- `std::vector< T > v_orig`

### Protected Attributes inherited from `GiNaC::basic_multi_iterator< T >`

- `T N`
- `T B`
- `std::vector< T > v`
- `bool flag_overflow`

## 6.101.1 Detailed Description

```
template<class T>
class GiNaC::multi_iterator_shuffle_prime< T >
```

The class `multi_iterator_shuffle_prime` defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).

## 6.101.2 Constructor & Destructor Documentation

### 6.101.2.1 multi\_iterator\_shuffle\_prime() [1/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
 void) [inline]
```

Default constructor.

### 6.101.2.2 multi\_iterator\_shuffle\_prime() [2/2]

```
template<class T >
GiNaC::multi_iterator_shuffle_prime< T >::multi_iterator_shuffle_prime (
 const std::vector< T > & a,
 const std::vector< T > & b) [inline], [explicit]
```

Construct from a vector.

## 6.101.3 Member Function Documentation

### 6.101.3.1 init()

```
template<class T >
basic_multi_iterator< T > & GiNaC::multi_iterator_shuffle_prime< T >::init (
 void) [inline], [virtual]
```

Initialize the multi-index to the first shuffle.

Reimplemented from [GiNaC::multi\\_iterator\\_shuffle< T >](#).

## 6.101.4 Friends And Related Symbol Documentation

### 6.101.4.1 operator<<

```
template<class T >
template<class TT >
std::ostream & operator<< (
 std::ostream & os,
 const multi_iterator_shuffle_prime< TT > & v) [friend]
```

The documentation for this class was generated from the following file:

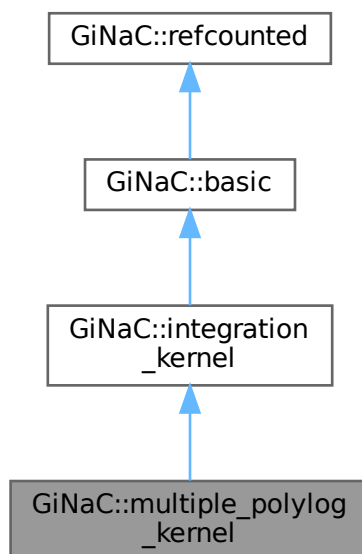
- [utils\\_multi\\_iterator.h](#)

## 6.102 GiNaC::multiple\_polylog\_kernel Class Reference

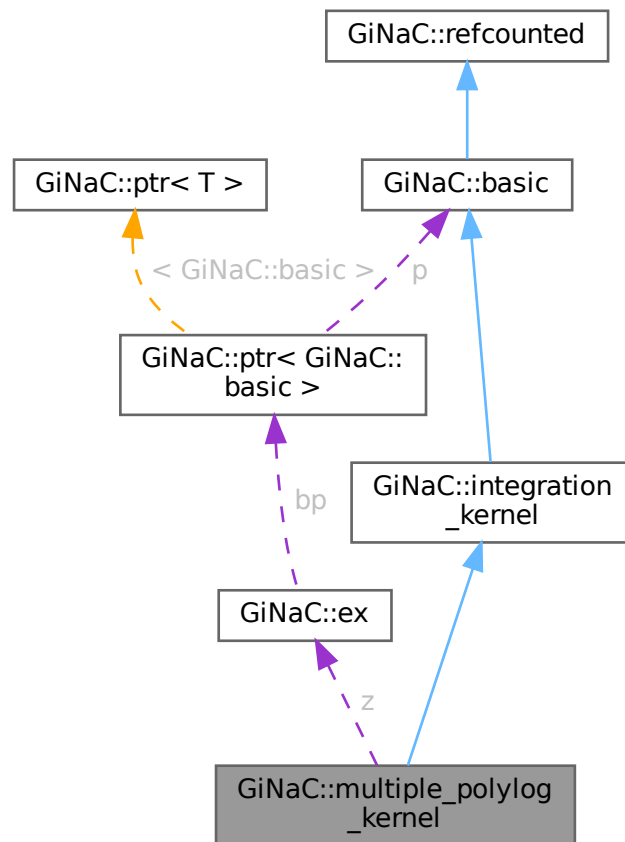
The integration kernel for multiple polylogarithms.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::multiple\_polylog\_kernel:



Collaboration diagram for GiNaC::multiple\_polylog\_kernel:



### Public Member Functions

- `multiple_polylog_kernel` (const `ex` &`z`)
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex & let_op` (size\_t `i`) override  
*Return modifiable operand/member at position `i`.*
- `bool is_numeric` (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*

### Public Member Functions inherited from `GiNaC::integration_kernel`

- `ex series` (const `relational` &`r`, int `order`, unsigned `options`=0) const override  
*Default implementation of `ex::series()`.*
- virtual `bool has_trailing_zero` (void) const

- *This routine returns true, if the integration kernel has a trailing zero.*
- virtual `ex Laurent_series` (const `ex` &`x`, int `order`) const  
*Returns the Laurent series, starting possibly with the pole term.*
- virtual `ex get_numerical_value` (const `ex` &`lambda`, int `N_trunc`=0) const  
*Evaluates the integrand at lambda.*
- `size_t get_cache_size` (void) const  
*Returns the current size of the cache.*
- void `set_cache_step` (int `cache_steps`) const  
*Sets the step size by which the cache is increased.*
- `ex get_series_coeff` (int `i`) const  
*Wrapper around series\_coeff(i), converts cl\_N to numeric.*
- `cln::cl_N series_coeff` (int `i`) const  
*Subclasses have either to implement series\_coeff\_impl or the two methods Laurent\_series and uses\_Laurent\_series.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &`c`, unsigned `level`=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned `inf`) const  
*Information about the object.*
- virtual `ex operator[]` (const `ex` &`index`) const
- virtual `ex operator[]` (size\_t `i`) const
- virtual `ex & operator[]` (const `ex` &`index`)
- virtual `ex & operator[]` (size\_t `i`)
- virtual bool `has` (const `ex` &`other`, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &`pattern`, `exmap` &`repls`) const

- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `cln::cl_N series_coeff_impl` (int i) const override  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- virtual bool `uses_Laurent_series` () const  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int shift, int N\_trunc) const  
*The actual implementation for computing a numerical value for the integrand.*
- void `do_print` (const `print_context` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const

- Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- `ex z`

### Protected Attributes inherited from `GiNaC::integration_kernel`

- int `cache_step_size`
- `std::vector< cln::cl_N >` `series_vec`

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.102.1 Detailed Description

The integration kernel for multiple polylogarithms.

This class represents the differential one-form

$$\omega^{\text{mpl}}(z) = \frac{d\lambda}{\lambda - z}$$

For the case  $z = 0$  the class `basic_log_kernel` should be used.

## 6.102.2 Constructor & Destructor Documentation

### 6.102.2.1 `multiple_polylog_kernel()`

```
GiNaC::multiple_polylog_kernel::multiple_polylog_kernel (
 const ex & z)
```

### 6.102.3 Member Function Documentation

#### 6.102.3.1 nops()

```
size_t GiNaC::multiple_polylog_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.102.3.2 op()

```
ex GiNaC::multiple_polylog_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [z](#).

#### 6.102.3.3 let\_op()

```
ex & GiNaC::multiple_polylog_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), and [z](#).

#### 6.102.3.4 is\_numeric()

```
bool GiNaC::multiple_polylog_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), and [z](#).

#### 6.102.3.5 series\_coeff\_impl()

```
cln::cl_N GiNaC::multiple_polylog_kernel::series_coeff_impl (
 int i) const [override], [protected], [virtual]
```

For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.

The i-th coefficient corresponds to the power  $\lambda^{i-1}$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), and [z](#).

### 6.102.3.6 do\_print()

```
void GiNaC::multiple_polylog_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::ex::print\(\)](#), and [z](#).

## 6.102.4 Member Data Documentation

### 6.102.4.1 z

```
ex GiNaC::multiple_polylog_kernel::z [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [let\\_op\(\)](#), [op\(\)](#), and [series\\_coeff\\_impl\(\)](#).

The documentation for this class was generated from the following files:

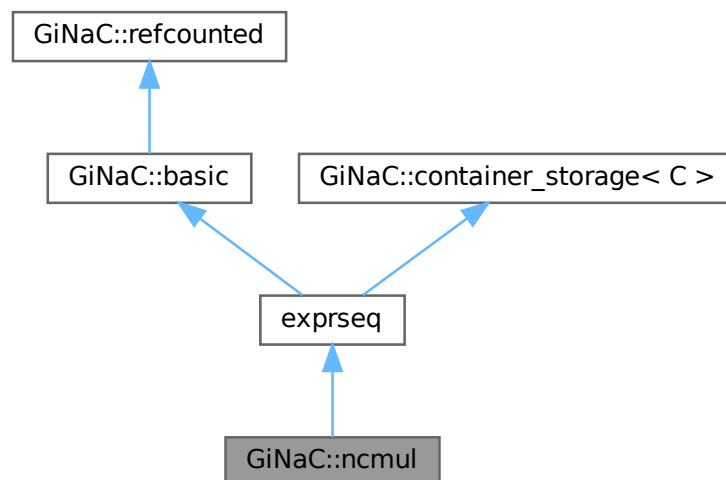
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.103 GiNaC::ncmul Class Reference

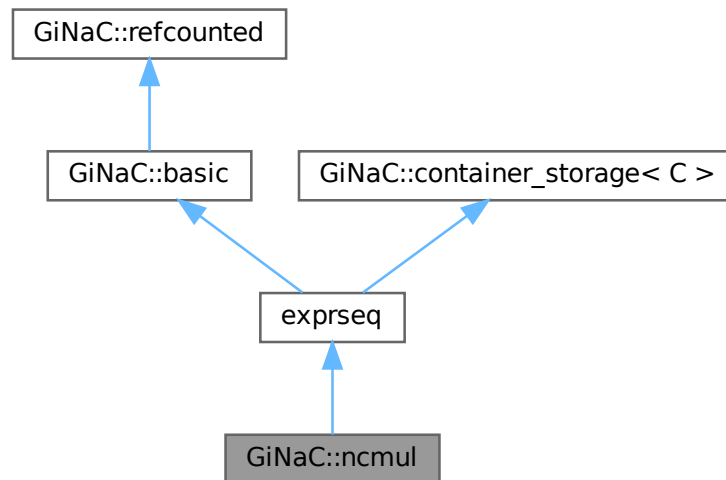
Non-commutative product of expressions.

```
#include <ncmul.h>
```

Inheritance diagram for GiNaC::ncmul:



Collaboration diagram for GiNaC::ncmul:



## Public Member Functions

- `ncmul` (const `ex` &lh, const `ex` &rh)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5)
- `ncmul` (const `ex` &f1, const `ex` &f2, const `ex` &f3, const `ex` &f4, const `ex` &f5, const `ex` &f6)
- `ncmul` (const `exvector` &v)
- `ncmul` (`exvector` &&v)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- `ex coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- `ex eval` () const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalm` () const override  
*Evaluate sums, products and integer powers of matrices.*
- `exvector get_free_indices` () const override  
*Return a vector containing the free indices of an expression.*
- `ex thiscontainer` (const `exvector` &v) const override

- [ex thiscontainer](#) ([exvector](#) &&v) const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- const [exvector](#) & [get\\_factors](#) () const

## Public Member Functions inherited from [GiNaC::container< C >](#)

- [container](#) (STLT const &s)
- [container](#) (STLT &&v)
- [container](#) (exvector::const\_iterator b, exvector::const\_iterator e)
- [container](#) (std::initializer\_list< [ex](#) > il)
- [size\\_t nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &sym\_lst) override  
*Load (deserialize) the object from an archive node.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Archive the object.*
- [container & prepend](#) (const [ex](#) &b)  
*Add element at front.*
- [container & append](#) (const [ex](#) &b)  
*Add element at back.*
- [container & remove\\_first](#) ()  
*Remove first element.*
- [container & remove\\_last](#) ()  
*Remove last element.*
- [container & remove\\_all](#) ()  
*Remove all elements.*
- [container & sort](#) ()  
*Sort elements.*
- [container & unique](#) ()  
*Remove adjacent duplicate elements.*
- [const\\_iterator begin](#) () const
- [const\\_iterator end](#) () const
- [const\\_reverse\\_iterator rbegin](#) () const
- [const\\_reverse\\_iterator rend](#) () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial (exmap &repl) const`
- virtual `numeric integer_content () const`
- virtual `ex smod (const numeric &xi) const`  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient () const`  
*Implementation `ex::max_coefficient()`.*
- virtual `ex add_indexed (const ex &self, const ex &other) const`  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other) const`

- *Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a non-commutative product.*
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_csrf` (const `print_context` &c, unsigned level) const
- size\_t `count_factors` (const `ex` &e) const
- void `append_factors` (`exvector` &v, const `ex` &e) const
- `exvector expandchildren` (unsigned `options`) const

## Protected Member Functions inherited from [GiNaC::container< C >](#)

- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- virtual [ex thiscontainer](#) (const [STLT](#) &v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence.*
- virtual [ex thiscontainer](#) ([STLT](#) &&v) const  
*Similar to [duplicate\(\)](#), but with a preset sequence (which gets pilfered).*
- virtual void [printseq](#) (const [print\\_context](#) &c, char openbracket, char delim, char closebracket, unsigned this←\_precedence, unsigned upper\_precedence=0) const  
*Print sequence of contained elements.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python](#) (const [print\\_python](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
- [STLT](#) [subchildren](#) (const [exmap](#) &m, unsigned [options](#)=0) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- [container\\_storage](#) ()
- [container\\_storage](#) (size\_t n, const [ex](#) &e)
- [container\\_storage](#) (std::initializer\_list< [ex](#) > il)
- template<class In >  
  [container\\_storage](#) (In b, In e)
- void [reserve](#) (size\_t)
- [~container\\_storage](#) ()
- void [reserve](#) (size\_t n)
- void [reserve](#) (std::vector< [ex](#) > &v, size\_t n)

## Friends

- class [power](#)
- [ex reeval\\_ncmul](#) (const [exvector](#) &v)
- [ex hold\\_ncmul](#) (const [exvector](#) &v)

## Additional Inherited Members

### Public Types inherited from [GiNaC::container< C >](#)

- typedef [STL::const\\_iterator](#) [const\\_iterator](#)
- typedef [STL::const\\_reverse\\_iterator](#) [const\\_reverse\\_iterator](#)

### Protected Types inherited from [GiNaC::container< C >](#)

- typedef [container\\_storage< C >::STLT](#) [STLT](#)

### Protected Types inherited from [GiNaC::container\\_storage< C >](#)

- typedef [C< ex >](#) [STLT](#)

### Static Protected Member Functions inherited from [GiNaC::container< C >](#)

- static unsigned [get\\_default\\_flags](#) ()  
*Specialization of [container::get\\_default\\_flags\(\)](#) for [lst](#).*
- static char [get\\_open\\_delim](#) ()  
*Specialization of [container::get\\_open\\_delim\(\)](#) for [lst](#).*
- static char [get\\_close\\_delim](#) ()  
*Specialization of [container::get\\_close\\_delim\(\)](#) for [lst](#).*

### Static Protected Member Functions inherited from [GiNaC::container\\_storage< C >](#)

- static void [reserve](#) ([STLT](#) &, [size\\_t](#))

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### Protected Attributes inherited from [GiNaC::container\\_storage< C >](#)

- [STLT seq](#)

### 6.103.1 Detailed Description

Non-commutative product of expressions.

### 6.103.2 Constructor & Destructor Documentation

#### 6.103.2.1 `ncmul()` [1/7]

```
GiNaC::ncmul::ncmul (
 const ex & lh,
 const ex & rh)
```

#### 6.103.2.2 `ncmul()` [2/7]

```
GiNaC::ncmul::ncmul (
 const ex & f1,
 const ex & f2,
 const ex & f3)
```

#### 6.103.2.3 `ncmul()` [3/7]

```
GiNaC::ncmul::ncmul (
 const ex & f1,
 const ex & f2,
 const ex & f3,
 const ex & f4)
```

#### 6.103.2.4 `ncmul()` [4/7]

```
GiNaC::ncmul::ncmul (
 const ex & f1,
 const ex & f2,
 const ex & f3,
 const ex & f4,
 const ex & f5)
```

#### 6.103.2.5 `ncmul()` [5/7]

```
GiNaC::ncmul::ncmul (
 const ex & f1,
 const ex & f2,
 const ex & f3,
 const ex & f4,
 const ex & f5,
 const ex & f6)
```

**6.103.2.6 ncmul()** [6/7]

```
GiNaC::ncmul::ncmul (
 const exvector & v)
```

**6.103.2.7 ncmul()** [7/7]

```
GiNaC::ncmul::ncmul (
 exvector && v)
```

**6.103.3 Member Function Documentation****6.103.3.1 precedence()**

```
unsigned GiNaC::ncmul::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::container< C >](#).

Referenced by [do\\_print\(\)](#), and [do\\_print\\_csrc\(\)](#).

**6.103.3.2 info()**

```
bool GiNaC::ncmul::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::container< C >](#).

**6.103.3.3 degree()**

```
int GiNaC::ncmul::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.103.3.4 ldegree()

```
int GiNaC::ncmul::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

#### 6.103.3.5 expand()

```
ex GiNaC::ncmul::expand (
 unsigned options = 0) const [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [expandchildren\(\)](#), [GiNaC::status\\_flags::expanded](#), [k](#), [GiNaC::container< C >::op\(\)](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::basic::setflag\(\)](#).

#### 6.103.3.6 coeff()

```
ex GiNaC::ncmul::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [n](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [GiNaC::add::coeff\(\)](#).

#### 6.103.3.7 eval()

```
ex GiNaC::ncmul::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following x, x1, x2,... stand for a symbolic variables of type [ex](#) and c, c1, c2... stand for such expressions that contain a plain number.

- [ncmul](#)(...,\*(x1,x2),...,[ncmul](#)(x3,x4),...) -> [ncmul](#)(...,x1,x2,...,x3,x4,...) (associativity)
- [ncmul](#)(x) -> x
- [ncmul](#)() -> 1
- [ncmul](#)(...,c1,...,c2,...) -> \*(c1,c2,[ncmul](#)(...)) (pull out commutative elements)
- [ncmul](#)(x1,y1,x2,y2) -> \*([ncmul](#)(x1,x2),[ncmul](#)(y1,y2)) (collect elements of same type)
- [ncmul](#)(x1,x2,x3,...) -> x::eval\_[ncmul](#)(x1,x2,x3,...)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::factor\(\)](#), [factors](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), [GiNaC::container\\_storage< C >::reserve](#) and [GiNaC::container\\_storage< C >::seq](#).

**6.103.3.8 evalm()**

```
ex GiNaC::ncmul::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::begin\(\)](#), [GiNaC::matrix::mul\(\)](#), and [GiNaC::container\\_storage< C >::seq](#).

**6.103.3.9 get\_free\_indices()**

```
exvector GiNaC::ncmul::get_free_indices () const [override], [virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::ex::get\\_free\\_indices\(\)](#), [GiNaC::container< C >::nops\(\)](#), and [GiNaC::container< C >::op\(\)](#).

**6.103.3.10 thiscontainer() [1/2]**

```
ex GiNaC::ncmul::thiscontainer (
 const exvector & v) const [override]
```

**6.103.3.11 thiscontainer() [2/2]**

```
ex GiNaC::ncmul::thiscontainer (
 exvector && v) const [override]
```

**6.103.3.12 conjugate()**

```
ex GiNaC::ncmul::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::container< C >::begin\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::container< C >::nops\(\)](#), [return\\_type\(\)](#), and [return\\_type\\_tinfo\(\)](#).

**6.103.3.13 real\_part()**

```
ex GiNaC::ncmul::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::real\\_part\(\)](#).

**6.103.3.14 imag\_part()**

```
ex GiNaC::ncmul::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::container< C >](#).

References [GiNaC::basic::imag\\_part\(\)](#).

**6.103.3.15 derivative()**

```
ex GiNaC::ncmul::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a non-commutative product.

It applies the product rule.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::diff\(\)](#), [GiNaC::container\\_storage< C >::seq](#), and [GiNaC::ex::swap\(\)](#).

**6.103.3.16 return\_type()**

```
unsigned GiNaC::ncmul::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#), [GiNaC::container< C >::end\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::return\\_types::noncommutative](#), [GiNaC::return\\_types::noncommutative\\_composite](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.17 return\_type\_tinfo()**

```
return_type_t GiNaC::ncmul::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [conjugate\(\)](#).

**6.103.3.18 do\_print()**

```
void GiNaC::ncmul::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.19 do\_print\_csrc()**

```
void GiNaC::ncmul::do_print_csrc (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [precedence\(\)](#), and [GiNaC::container< C >::printseq\(\)](#).

**6.103.3.20 count\_factors()**

```
size_t GiNaC::ncmul::count_factors (
 const ex & e) const [protected]
```

References [GiNaC::return\\_types::commutative](#), [count\\_factors\(\)](#), [factors](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [count\\_factors\(\)](#), and [eval\(\)](#).

**6.103.3.21 append\_factors()**

```
void GiNaC::ncmul::append_factors (
 exvector & v,
 const ex & e) const [protected]
```

References [append\\_factors\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [GiNaC::ex::return\\_type\(\)](#).

Referenced by [append\\_factors\(\)](#), and [eval\(\)](#).

**6.103.3.22 expandchildren()**

```
exvector GiNaC::ncmul::expandchildren (
 unsigned options) const [protected]
```

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::container< C >::end\(\)](#), [GiNaC::ex::expand\(\)](#), [options](#), and [GiNaC::container\\_storage< C >::seq](#).

Referenced by [expand\(\)](#).

**6.103.3.23 get\_factors()**

```
const exvector & GiNaC::ncmul::get_factors () const
```

References [GiNaC::container\\_storage< C >::seq](#).

**6.103.4 Friends And Related Symbol Documentation****6.103.4.1 power**

```
friend class power [friend]
```

#### 6.103.4.2 reeval\_ncmul

```
ex reeval_ncmul (
 const exvector & v) [friend]
```

#### 6.103.4.3 hold\_ncmul

```
ex hold_ncmul (
 const exvector & v) [friend]
```

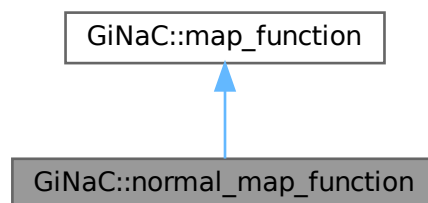
The documentation for this class was generated from the following files:

- [ncmul.h](#)
- [indexed.cpp](#)
- [ncmul.cpp](#)

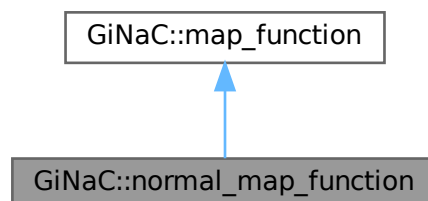
### 6.104 GiNaC::normal\_map\_function Struct Reference

Function object to be applied by [basic::normal\(\)](#).

Inheritance diagram for GiNaC::normal\_map\_function:



Collaboration diagram for GiNaC::normal\_map\_function:



## Public Member Functions

- [ex operator\(\)](#) (const [ex](#) &*e*) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

### 6.104.1 Detailed Description

Function object to be applied by [basic::normal\(\)](#).

### 6.104.2 Member Function Documentation

#### 6.104.2.1 [operator\(\)](#)()

```
ex GiNaC::normal_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::normal\(\)](#).

The documentation for this struct was generated from the following file:

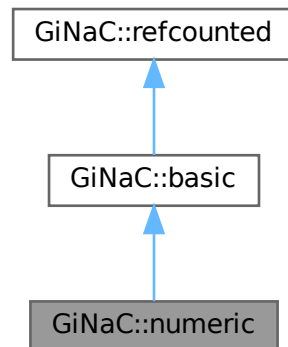
- [normal.cpp](#)

## 6.105 GiNaC::numeric Class Reference

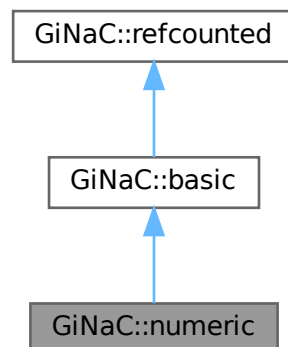
This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::numeric:



Collaboration diagram for GiNaC::numeric:



### Public Member Functions

- [numeric](#) (int i)
- [numeric](#) (unsigned int i)
- [numeric](#) (long i)
- [numeric](#) (unsigned long i)

- `numeric` (long long i)
- `numeric` (unsigned long long i)
- `numeric` (long `numer`, long `denom`)  
*Constructor for rational numerics a/b.*
- `numeric` (double d)
- `numeric` (const char \*)  
*ctor from C-style string.*
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*
- int `ldegree` (const `ex` &s) const override  
*Return degree of lowest power in object s.*
- `ex coeff` (const `ex` &s, int `n`=1) const override  
*Return coefficient of degree n in object s.*
- bool `has` (const `ex` &other, unsigned `options`=0) const override  
*Disassemble real part and imaginary part to scan for the occurrence of a single number.*
- `ex eval` () const override  
*Evaluation of numbers doesn't do anything at all.*
- `ex evalf` () const override  
*Cast numeric into a floating-point object.*
- `ex subs` (const `exmap` &m, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const override  
*Implementation of `ex::normal()` for a numeric.*
- `ex to_rational` (`exmap` &repl) const override  
*Implementation of `ex::to_rational()` for a numeric.*
- `ex to_polynomial` (`exmap` &repl) const override  
*Implementation of `ex::to_polynomial()` for a numeric.*
- `numeric integer_content` () const override
- `ex smod` (const `numeric` &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- `numeric max_coefficient` () const override  
*Implementation `ex::max_coefficient()`.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*
- const `numeric add` (const `numeric` &other) const  
*Numerical addition method.*
- const `numeric sub` (const `numeric` &other) const  
*Numerical subtraction method.*
- const `numeric mul` (const `numeric` &other) const  
*Numerical multiplication method.*

- const [numeric div](#) (const [numeric](#) &other) const  
*Numerical division method.*
- const [numeric power](#) (const [numeric](#) &other) const  
*Numerical exponentiation.*
- const [numeric & add\\_dyn](#) (const [numeric](#) &other) const  
*Numerical addition method.*
- const [numeric & sub\\_dyn](#) (const [numeric](#) &other) const  
*Numerical subtraction method.*
- const [numeric & mul\\_dyn](#) (const [numeric](#) &other) const  
*Numerical multiplication method.*
- const [numeric & div\\_dyn](#) (const [numeric](#) &other) const  
*Numerical division method.*
- const [numeric & power\\_dyn](#) (const [numeric](#) &other) const  
*Numerical exponentiation.*
- const [numeric & operator=](#) (int i)
- const [numeric & operator=](#) (unsigned int i)
- const [numeric & operator=](#) (long i)
- const [numeric & operator=](#) (unsigned long i)
- const [numeric & operator=](#) (double d)
- const [numeric & operator=](#) (const char \*s)
- const [numeric inverse](#) () const  
*Inverse of a number.*
- [numeric step](#) () const  
*Return the step function of a numeric.*
- int [csgn](#) () const  
*Return the complex half-plane (left or right) in which the number lies.*
- int [compare](#) (const [numeric](#) &other) const  
*This method establishes a canonical order on all numbers.*
- bool [is\\_equal](#) (const [numeric](#) &other) const
- bool [is\\_zero](#) () const  
*True if object is zero.*
- bool [is\\_positive](#) () const  
*True if object is not complex and greater than zero.*
- bool [is\\_negative](#) () const  
*True if object is not complex and less than zero.*
- bool [is\\_integer](#) () const  
*True if object is a non-complex integer.*
- bool [is\\_pos\\_integer](#) () const  
*True if object is an exact integer greater than zero.*
- bool [is\\_nonneg\\_integer](#) () const  
*True if object is an exact integer greater or equal zero.*
- bool [is\\_even](#) () const  
*True if object is an exact even integer.*
- bool [is\\_odd](#) () const  
*True if object is an exact odd integer.*
- bool [is\\_prime](#) () const  
*Probabilistic primality test.*
- bool [is\\_rational](#) () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool [is\\_real](#) () const  
*True if object is a real integer, rational or float (but not complex).*

- bool `is_cinteger` () const  
*True if object is element of the domain of integers extended by I, i.e.*
- bool `is_crational` () const  
*True if object is an exact rational number, may even be complex (denominator may be unity).*
- bool `operator==` (const `numeric` &other) const
- bool `operator!=` (const `numeric` &other) const
- bool `operator<` (const `numeric` &other) const  
*Numerical comparison: less.*
- bool `operator<=` (const `numeric` &other) const  
*Numerical comparison: less or equal.*
- bool `operator>` (const `numeric` &other) const  
*Numerical comparison: greater.*
- bool `operator>=` (const `numeric` &other) const  
*Numerical comparison: greater or equal.*
- int `to_int` () const  
*Converts numeric types to machine's int.*
- long `to_long` () const  
*Converts numeric types to machine's long.*
- double `to_double` () const  
*Converts numeric types to machine's double.*
- `cln::cl_N to_cl_N` () const  
*Returns a new CLN object of type cl\_N, representing the value of \*this.*
- const `numeric real` () const  
*Real part of a number.*
- const `numeric imag` () const  
*Imaginary part of a number.*
- const `numeric numer` () const  
*Numerator.*
- const `numeric denom` () const  
*Denominator.*
- int `int_length` () const  
*Size in binary notation.*
- `numeric` (const `cln::cl_N` &z)  
*Ctor from CLN types.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprnttree` () const  
*Little wrapper around prnttree to be called within a debugger.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic & hold` () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const

*Set some [status\\_flags](#).*

- const [basic](#) & [clearflag](#) (unsigned f) const

*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff](#) for a numeric always returns 0.*
- bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_numeric](#) (const [print\\_context](#) &c, const char \*par\_open, const char \*par\_close, const char \*imag↔  
\_sym, const char \*mul\_sym, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_csrc\\_cl\\_N](#) (const [print\\_csrc\\_cl\\_N](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- `cln::cl_N` [value](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 6.105.1 Detailed Description

This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.

Objects of this type may directly be created by the user.

### 6.105.2 Constructor & Destructor Documentation

#### 6.105.2.1 `numeric()` [1/10]

```
GiNaC::numeric::numeric (
 int i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

Referenced by [add\(\)](#), [conjugate\(\)](#), [denom\(\)](#), [div\(\)](#), [evalf\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [inverse\(\)](#), [mul\(\)](#), [numer\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [power\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [step\(\)](#), and [sub\(\)](#).

#### 6.105.2.2 `numeric()` [2/10]

```
GiNaC::numeric::numeric (
 unsigned int i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.2.3 `numeric()` [3/10]

```
GiNaC::numeric::numeric (
 long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.2.4 `numeric()` [4/10]

```
GiNaC::numeric::numeric (
 unsigned long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.5 numeric()** [5/10]

```
GiNaC::numeric::numeric (
 long long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.6 numeric()** [6/10]

```
GiNaC::numeric::numeric (
 unsigned long long i)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.7 numeric()** [7/10]

```
GiNaC::numeric::numeric (
 long numer,
 long denom)
```

Constructor for rational numerics a/b.

**Exceptions**

|                       |                    |
|-----------------------|--------------------|
| <i>overflow_error</i> | (division by zero) |
|-----------------------|--------------------|

References [denom\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [numer\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.8 numeric()** [8/10]

```
GiNaC::numeric::numeric (
 double d)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.2.9 numeric()** [9/10]

```
GiNaC::numeric::numeric (
 const char * s)
```

ctor from C-style string.

It also accepts complex numbers in [GiNaC](#) notation like "2+5\*I".

References [GiNaC::Digits](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

### 6.105.2.10 numeric() [10/10]

```
GiNaC::numeric::numeric (
 const cln::cl_N & z) [explicit]
```

Ctor from CLN types.

This is for the initiated user or internal use only.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

## 6.105.3 Member Function Documentation

### 6.105.3.1 precedence()

```
unsigned GiNaC::numeric::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_numeric\(\)](#).

### 6.105.3.2 info()

```
bool GiNaC::numeric::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::info\\_flags::expanded](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::info\\_flags::odd](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::posint](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::prime](#), [GiNaC::info\\_flags::rational](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), and [GiNaC::info\\_flags::real](#).

Referenced by [GiNaC::abs\\_power\(\)](#), [GiNaC::csgn\\_power\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

### 6.105.3.3 is\_polynomial()

```
bool GiNaC::numeric::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.4 degree()

```
int GiNaC::numeric::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.5 ldegree()

```
int GiNaC::numeric::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

#### 6.105.3.6 coeff()

```
ex GiNaC::numeric::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), and [n](#).

Referenced by [GiNaC::pseries::mul\\_const\(\)](#).

#### 6.105.3.7 has()

```
bool GiNaC::numeric::has (
 const ex & other,
 unsigned options = 0) const [override], [virtual]
```

Disassemble real part and imaginary part to scan for the occurrence of a single number.

Also handles the imaginary unit. It ignores the sign on both this and the argument, which may lead to what might appear as funny results:  $(2+i).has(-2) \rightarrow \text{true}$ . But this is consistent, since we also would like to have  $(-2+i).has(2) \rightarrow \text{true}$  and we want to think about the sign as a multiplicative factor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_num0\\_p](#), [GiNaC::i](#), [imag\(\)](#), [is\\_equal\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), and [real\(\)](#).

### 6.105.3.8 eval()

```
ex GiNaC::numeric::eval () const [override], [virtual]
```

Evaluation of numbers doesn't do anything at all.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

### 6.105.3.9 evalf()

```
ex GiNaC::numeric::evalf () const [override], [virtual]
```

Cast numeric into a floating-point object.

For example `exact numeric(1)` is returned as a `1.0000000000000000000000` and so on according to how `Digits` is currently set. In case the object already was a floating point number the precision is trimmed to match the currently set default.

#### Returns

an ex-handle to a numeric.

Reimplemented from [GiNaC::basic](#).

References [numeric\(\)](#), and [value](#).

### 6.105.3.10 subs()

```
ex GiNaC::numeric::subs (
 const exmap & m,
 unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), [options](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

### 6.105.3.11 normal()

```
ex GiNaC::numeric::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for a numeric.

It splits complex numbers into `re+I*im` and replaces `I` and non-rational real numbers with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [denom\(\)](#), [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [numer\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.12 to\_rational()**

```
ex GiNaC::numeric::to_rational (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-rational real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.13 to\_polynomial()**

```
ex GiNaC::numeric::to_polynomial (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for a numeric.

It splits complex numbers into  $re+I*im$  and replaces  $I$  and non-integer real numbers with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::I](#), [imag\(\)](#), [is\\_integer\(\)](#), [is\\_real\(\)](#), [real\(\)](#), and [GiNaC::replace\\_with\\_symbol\(\)](#).

**6.105.3.14 integer\_content()**

```
numeric GiNaC::numeric::integer_content () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

Referenced by [GiNaC::mul::eval\(\)](#), and [GiNaC::ex::integer\\_content\(\)](#).

**6.105.3.15 smod()**

```
ex GiNaC::numeric::smod (
 const numeric & xi) const [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

Parameters

|      |         |
|------|---------|
| $xi$ | modulus |
|------|---------|

**Returns**

mapped polynomial

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::smod\(\)](#).

**6.105.3.16 max\_coefficient()**

```
numeric GiNaC::numeric::max_coefficient () const [override], [virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

**See also**

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::abs\(\)](#).

Referenced by [GiNaC::ex::max\\_coefficient\(\)](#).

**6.105.3.17 conjugate()**

```
ex GiNaC::numeric::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [is\\_real\(\)](#), [numeric\(\)](#), and [value](#).

**6.105.3.18 real\_part()**

```
ex GiNaC::numeric::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [numeric\(\)](#), and [value](#).

**6.105.3.19 imag\_part()**

```
ex GiNaC::numeric::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [numeric\(\)](#), and [value](#).

**6.105.3.20 archive()**

```
void GiNaC::numeric::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [value](#), and [GiNaC::write\\_real\\_float\(\)](#).

**6.105.3.21 read\_archive()**

```
void GiNaC::numeric::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [c](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [n](#), [GiNaC::read\\_real\\_float\(\)](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

**6.105.3.22 derivative()**

```
ex GiNaC::numeric::derivative (
 const symbol & s) const [inline], [override], [protected], [virtual]
```

Implementation of [ex::diff](#) for a numeric always returns 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

**6.105.3.23 is\_equal\_same\_type()**

```
bool GiNaC::numeric::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [is\\_equal\(\)](#).

#### 6.105.3.24 calchash()

```
unsigned GiNaC::numeric::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class `basic` computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::basic::setflag\(\)](#), and [value](#).

#### 6.105.3.25 add()

```
const numeric GiNaC::numeric::add (
 const numeric & other) const
```

Numerical addition method.

Adds argument to `*this` and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator+\(\)](#), [GiNaC::operator++\(\)](#), [GiNaC::operator++\(\)](#), [GiNaC::operator+=\(\)](#), [GiNaC::operator--\(\)](#), and [GiNaC::operator--\(\)](#).

#### 6.105.3.26 sub()

```
const numeric GiNaC::numeric::sub (
 const numeric & other) const
```

Numerical subtraction method.

Subtracts argument from `*this` and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::operator-\(\)](#), [GiNaC::operator-=\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

#### 6.105.3.27 mul()

```
const numeric GiNaC::numeric::mul (
 const numeric & other) const
```

Numerical multiplication method.

Multiplies `*this` and argument and returns result as a numeric object.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator\\*\(\)](#), [GiNaC::operator\\*=\(\)](#), and [GiNaC::operator-\(\)](#).

#### 6.105.3.28 div()

```
const numeric GiNaC::numeric::div (
 const numeric & other) const
```

Numerical division method.

Divides `*this` by argument and returns result as a numeric object.

## Exceptions

|                             |                    |
|-----------------------------|--------------------|
| <code>overflow_error</code> | (division by zero) |
|-----------------------------|--------------------|

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::operator/\(\)](#), [GiNaC::operator/=\(\(\)\)](#), [GiNaC::basic::series\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

**6.105.3.29 power()**

```
const numeric GiNaC::numeric::power (
 const numeric & other) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::binomial\(\)](#), and [GiNaC::power::eval\(\)](#).

**6.105.3.30 add\_dyn()**

```
const numeric & GiNaC::numeric::add_dyn (
 const numeric & other) const
```

Numerical addition method.

Adds argument to \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), and [value](#).

**6.105.3.31 sub\_dyn()**

```
const numeric & GiNaC::numeric::sub_dyn (
 const numeric & other) const
```

Numerical subtraction method.

Subtracts argument from \*this and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), and [value](#).

**6.105.3.32 mul\_dyn()**

```
const numeric & GiNaC::numeric::mul_dyn (
 const numeric & other) const
```

Numerical multiplication method.

Multiplies \*this and argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num1\\_p](#), and [value](#).

Referenced by [GiNaC::power::expand\\_add\\_2\(\)](#).

**6.105.3.33 div\_dyn()**

```
const numeric & GiNaC::numeric::div_dyn (
 const numeric & other) const
```

Numerical division method.

Divides \*this by argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

**Exceptions**

|                                |                    |
|--------------------------------|--------------------|
| <a href="#">overflow_error</a> | (division by zero) |
|--------------------------------|--------------------|

References [GiNaC::\\_num1\\_p](#), and [value](#).

**6.105.3.34 power\_dyn()**

```
const numeric & GiNaC::numeric::power_dyn (
 const numeric & other) const
```

Numerical exponentiation.

Raises \*this to the power given as argument and returns result as a numeric object on the heap. Use internally only for direct wrapping into an ex object, where the result would end up on the heap anyways.

References [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), and [value](#).

**6.105.3.35 operator=() [1/6]**

```
const numeric & GiNaC::numeric::operator= (
 int i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

Referenced by [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), [operator=\(\)](#), and [operator=\(\)](#).

**6.105.3.36 operator=()** [2/6]

```
const numeric & GiNaC::numeric::operator= (
 unsigned int i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**6.105.3.37 operator=()** [3/6]

```
const numeric & GiNaC::numeric::operator= (
 long i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**6.105.3.38 operator=()** [4/6]

```
const numeric & GiNaC::numeric::operator= (
 unsigned long i)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**6.105.3.39 operator=()** [5/6]

```
const numeric & GiNaC::numeric::operator= (
 double d)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**6.105.3.40 operator=()** [6/6]

```
const numeric & GiNaC::numeric::operator= (
 const char * s)
```

References [numeric\(\)](#), and [operator=\(\)](#).

**6.105.3.41 inverse()**

```
const numeric GiNaC::numeric::inverse () const
```

Inverse of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::interpolate\(\)](#), and [GiNaC::psi1\\_eval\(\)](#).

#### 6.105.3.42 `step()`

```
numeric GiNaC::numeric::step () const
```

Return the step function of a numeric.

The imaginary part of it is ignored because the step function is generally considered real but a numeric may develop a small imaginary part due to rounding errors.

References [numeric\(\)](#), [r](#), and [value](#).

#### 6.105.3.43 `csgn()`

```
int GiNaC::numeric::csgn () const
```

Return the complex half-plane (left or right) in which the number lies.

$\text{csgn}(x) == 0$  for  $x == 0$ ,  $\text{csgn}(x) == 1$  for  $\text{Re}(x) > 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) > 0$ ,  $\text{csgn}(x) == -1$  for  $\text{Re}(x) < 0$  or  $\text{Re}(x) = 0$  and  $\text{Im}(x) < 0$ .

See also

`numeric::compare(const numeric &other)`

References [r](#), and [value](#).

#### 6.105.3.44 `compare()`

```
int GiNaC::numeric::compare (
 const numeric & other) const
```

This method establishes a canonical order on all numbers.

For complex numbers this is not possible in a mathematically consistent way but we need to establish some order and it ought to be fast. So we simply define it to be compatible with our method `csgn`.

Returns

`csgn(*this-other)`

See also

[numeric::csgn\(\)](#)

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#).

**6.105.3.45 is\_equal()**

```
bool GiNaC::numeric::is_equal (
 const numeric & other) const
```

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [has\(\)](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::tan\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

**6.105.3.46 is\_zero()**

```
bool GiNaC::numeric::is_zero () const
```

True if object is zero.

References [value](#).

Referenced by [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::expairseq::construct\\_from\\_2\\_expairseq\(\)](#), [GiNaC::expairseq::construct\\_from\\_expairseq\\_ex\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::gcd\(\)](#), [has\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::step\\_eval\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

**6.105.3.47 is\_positive()**

```
bool GiNaC::numeric::is_positive () const
```

True if object is not complex and greater than zero.

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

**6.105.3.48 is\_negative()**

```
bool GiNaC::numeric::is_negative () const
```

True if object is not complex and less than zero.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [info\(\)](#), and [GiNaC::pseries::power\\_const\(\)](#).

**6.105.3.49 is\_integer()**

```
bool GiNaC::numeric::is_integer () const
```

True if object is a non-complex integer.

References [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::gcd\(\)](#), [info\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::mod\(\)](#), [normal\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [to\\_int\(\)](#), [to\\_long\(\)](#), [to\\_polynomial\(\)](#), and [GiNaC::zeta1\\_eval\(\)](#).

#### 6.105.3.50 `is_pos_integer()`

```
bool GiNaC::numeric::is_pos_integer () const
```

True if object is an exact integer greater than zero.

References [value](#).

Referenced by [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [info\(\)](#), and [GiNaC::pseries::power\\_const\(\)](#).

#### 6.105.3.51 `is_nonneg_integer()`

```
bool GiNaC::numeric::is_nonneg_integer () const
```

True if object is an exact integer greater or equal zero.

References [value](#).

Referenced by [GiNaC::binomial\\_sym\(\)](#), [info\(\)](#), and [print\\_numeric\(\)](#).

#### 6.105.3.52 `is_even()`

```
bool GiNaC::numeric::is_even () const
```

True if object is an exact even integer.

References [value](#).

Referenced by [info\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), and [GiNaC::tgamma\\_eval\(\)](#).

#### 6.105.3.53 `is_odd()`

```
bool GiNaC::numeric::is_odd () const
```

True if object is an exact odd integer.

References [value](#).

Referenced by [info\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), and [GiNaC::matrix::pow\(\)](#).

#### 6.105.3.54 `is_prime()`

```
bool GiNaC::numeric::is_prime () const
```

Probabilistic primality test.

##### Returns

true if object is exact integer and prime.

References [value](#).

Referenced by [info\(\)](#).

**6.105.3.55 is\_rational()**

```
bool GiNaC::numeric::is_rational () const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [info\(\)](#), [GiNaC::multiply\\_lcm\(\)](#), [normal\(\)](#), and [to\\_rational\(\)](#).

**6.105.3.56 is\_real()**

```
bool GiNaC::numeric::is_real () const
```

True if object is a real integer, rational or float (but not complex).

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::beta\\_eval\(\)](#), [conjugate\(\)](#), [GiNaC::csgn\\_eval\(\)](#), [denom\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::fsolve\(\)](#), [has\(\)](#), [info\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [normal\(\)](#), [number\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_double\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

**6.105.3.57 is\_cinteger()**

```
bool GiNaC::numeric::is_cinteger () const
```

True if object is element of the domain of integers extended by  $i$ , i.e.

is of the form  $a+bi$ , where  $a$  and  $b$  are integers.

References [is\\_real\(\)](#), and [value](#).

Referenced by [info\(\)](#).

**6.105.3.58 is\_crational()**

```
bool GiNaC::numeric::is_crational () const
```

True if object is an exact rational number, may even be complex (denominator may be unity).

References [is\\_real\(\)](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), and [info\(\)](#).

**6.105.3.59 operator==( )**

```
bool GiNaC::numeric::operator== (
 const numeric & other) const
```

References [value](#).

**6.105.3.60 operator"!="()**

```
bool GiNaC::numeric::operator!= (
 const numeric & other) const
```

References [value](#).

**6.105.3.61 operator<()**

```
bool GiNaC::numeric::operator< (
 const numeric & other) const
```

Numerical comparison: less.

**Exceptions**

|                         |                      |
|-------------------------|----------------------|
| <i>invalid_argument</i> | (complex inequality) |
|-------------------------|----------------------|

References [is\\_real\(\)](#), and [value](#).

**6.105.3.62 operator<=()**

```
bool GiNaC::numeric::operator<= (
 const numeric & other) const
```

Numerical comparison: less or equal.

**Exceptions**

|                         |                      |
|-------------------------|----------------------|
| <i>invalid_argument</i> | (complex inequality) |
|-------------------------|----------------------|

References [is\\_real\(\)](#), and [value](#).

**6.105.3.63 operator>()**

```
bool GiNaC::numeric::operator> (
 const numeric & other) const
```

Numerical comparison: greater.

**Exceptions**

|                         |                      |
|-------------------------|----------------------|
| <i>invalid_argument</i> | (complex inequality) |
|-------------------------|----------------------|

References [is\\_real\(\)](#), and [value](#).

**6.105.3.64 operator>=()**

```
bool GiNaC::numeric::operator>= (
 const numeric & other) const
```

Numerical comparison: greater or equal.

**Exceptions**

|                         |                      |
|-------------------------|----------------------|
| <i>invalid_argument</i> | (complex inequality) |
|-------------------------|----------------------|

References [is\\_real\(\)](#), and [value](#).

**6.105.3.65 to\_int()**

```
int GiNaC::numeric::to_int () const
```

Converts numeric types to machine's int.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#) and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

**6.105.3.66 to\_long()**

```
long GiNaC::numeric::to_long () const
```

Converts numeric types to machine's long.

You should check with [is\\_integer\(\)](#) if the number is really an integer before calling this method. You may also consider checking the range first.

References [GINAC\\_ASSERT](#), [is\\_integer\(\)](#), and [value](#).

Referenced by [GiNaC::power::expand\(\)](#), and [GiNaC::power::expand\\_add\(\)](#).

**6.105.3.67 to\_double()**

```
double GiNaC::numeric::to_double () const
```

Converts numeric types to machine's double.

You should check with [is\\_real\(\)](#) if the number is really not complex before calling this method.

References [GINAC\\_ASSERT](#), [is\\_real\(\)](#), and [value](#).

**6.105.3.68 to\_cl\_N()**

```
cln::cl_N GiNaC::numeric::to_cl_N () const
```

Returns a new CLN object of type `cl_N`, representing the value of `*this`.

This method may be used when mixing [GiNaC](#) and CLN in one project.

References [value](#).

Referenced by [GiNaC::atan\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::irem\(\)](#), [GiNaC::lcm\(\)](#), [GiNaC::mod\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::smod\(\)](#).

### 6.105.3.69 real()

```
const numeric GiNaC::numeric::real () const
```

Real part of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::csgn\\_eval\(\)](#), [GiNaC::power::eval\(\)](#), [has\(\)](#), [normal\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.105.3.70 imag()

```
const numeric GiNaC::numeric::imag () const
```

Imaginary part of a number.

References [numeric\(\)](#), and [value](#).

Referenced by [GiNaC::csgn\\_eval\(\)](#), [has\(\)](#), [normal\(\)](#), [GiNaC::step\\_eval\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.105.3.71 numer()

```
const numeric GiNaC::numeric::numer () const
```

Numerator.

Computes the numerator of rational numbers, rationalized numerator of complex if real and imaginary part are both rational numbers (i.e `numer(4/3+5/6*I) == 8+5*I`), the number carrying the sign in all other cases.

References [is\\_real\(\)](#), [numeric\(\)](#), [r](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), and [numeric\(\)](#).

### 6.105.3.72 denom()

```
const numeric GiNaC::numeric::denom () const
```

Denominator.

Computes the denominator of rational numbers, common integer denominator of complex if real and imaginary part are both rational numbers (i.e `denom(4/3+5/6*I) == 6`), one in all other cases.

References [GiNaC::\\_num1\\_p](#), [is\\_real\(\)](#), [numeric\(\)](#), [r](#), and [value](#).

Referenced by [GiNaC::power::eval\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [normal\(\)](#), and [numeric\(\)](#).

### 6.105.3.73 int\_length()

```
int GiNaC::numeric::int_length () const
```

Size in binary notation.

For integers, this is the smallest  $n \geq 0$  such that  $-2^n \leq x < 2^n$ . If  $x > 0$ , this is the unique  $n > 0$  such that  $2^{(n-1)} \leq x < 2^n$ .

#### Returns

number of bits (excluding sign) needed to represent that number in two's complement if it is an integer, 0 otherwise.

References [value](#).

Referenced by [GiNaC::heur\\_gcd\\_z\(\)](#).

### 6.105.3.74 print\_numeric()

```
void GiNaC::numeric::print_numeric (
 const print_context & c,
 const char * par_open,
 const char * par_close,
 const char * imag_sym,
 const char * mul_sym,
 unsigned level) const [protected]
```

References [c](#), [is\\_nonneg\\_integer\(\)](#), [precedence\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [r](#), [GiNaC::print\\_context::s](#), and [value](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\\_repr\(\)](#).

### 6.105.3.75 do\_print()

```
void GiNaC::numeric::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

### 6.105.3.76 do\_print\_latex()

```
void GiNaC::numeric::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

**6.105.3.77 do\_print\_csrc()**

```
void GiNaC::numeric::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), and [value](#).

**6.105.3.78 do\_print\_csrc\_cl\_N()**

```
void GiNaC::numeric::do_print_csrc_cl_N (
 const print_csrc_cl_N & c,
 unsigned level) const [protected]
```

References [c](#), [is\\_real\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), and [value](#).

**6.105.3.79 do\_print\_tree()**

```
void GiNaC::numeric::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [value](#).

**6.105.3.80 do\_print\_python\_repr()**

```
void GiNaC::numeric::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_numeric\(\)](#).

**6.105.4 Member Data Documentation****6.105.4.1 value**

```
cln::cl_N GiNaC::numeric::value [protected]
```

Referenced by [add\(\)](#), [add\\_dyn\(\)](#), [archive\(\)](#), [calchash\(\)](#), [compare\(\)](#), [conjugate\(\)](#), [csgn\(\)](#), [denom\(\)](#), [div\(\)](#), [div\\_dyn\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_tree\(\)](#), [evalf\(\)](#), [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(\)](#), [imag\(\)](#), [imag\\_part\(\)](#), [int\\_length\(\)](#), [inverse\(\)](#), [is\\_cinteger\(\)](#), [is\\_crational\(\)](#), [is\\_equal\(\)](#), [is\\_even\(\)](#), [is\\_integer\(\)](#), [is\\_negative\(\)](#), [is\\_nonneg\\_integer\(\)](#), [is\\_odd\(\)](#), [is\\_pos\\_integer\(\)](#), [is\\_positive\(\)](#), [is\\_prime\(\)](#), [is\\_rational\(\)](#), [is\\_real\(\)](#), [is\\_zero\(\)](#), [mul\(\)](#), [mul\\_dyn\(\)](#), [numer\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [numeric\(\)](#), [operator!=\(\)](#), [operator<\(\)](#), [operator<=\(\)](#), [operator==\(\)](#), [operator>\(\)](#), [operator>=\(\)](#), [power\(\)](#), [power\\_dyn\(\)](#), [print\\_numeric\(\)](#), [read\\_archive\(\)](#), [real\(\)](#), [real\\_part\(\)](#), [step\(\)](#), [sub\(\)](#), [sub\\_dyn\(\)](#), [to\\_cl\\_N\(\)](#), [to\\_double\(\)](#), [to\\_int\(\)](#), and [to\\_long\(\)](#).

The documentation for this class was generated from the following files:

- [numeric.h](#)
- [normal.cpp](#)
- [numeric.cpp](#)

## 6.106 GiNaC::op0\_is\_equal Struct Reference

```
#include <ex.h>
```

### Public Member Functions

- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### 6.106.1 Member Function Documentation

#### 6.106.1.1 [operator\(\)](#)()

```
bool GiNaC::op0_is_equal::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::op\(\)](#).

The documentation for this struct was generated from the following file:

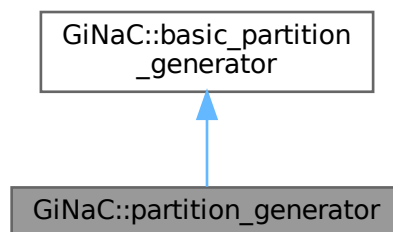
- [ex.h](#)

## 6.107 GiNaC::partition\_generator Class Reference

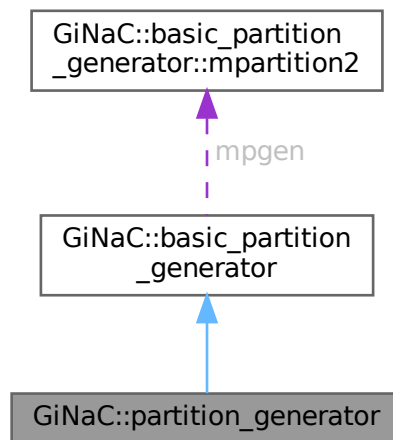
Generate all bounded combinatorial partitions of an integer n with exactly m parts (not including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::partition\_generator:



Collaboration diagram for `GiNaC::partition_generator`:



### Public Member Functions

- [partition\\_generator](#) (unsigned n\_, unsigned m\_)
- const std::vector< unsigned > & [get](#) () const
- bool [next](#) ()

### Private Attributes

- std::vector< unsigned > [partition](#)
- bool [current\\_updated](#)

### Additional Inherited Members

### Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)

- [basic\\_partition\\_generator](#) (unsigned n\_, unsigned m\_)

### Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)

- [mpartition2](#) [mpgen](#)

## 6.107.1 Detailed Description

Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.

## 6.107.2 Constructor & Destructor Documentation

### 6.107.2.1 partition\_generator()

```
GiNaC::partition_generator::partition_generator (
 unsigned n_,
 unsigned m_) [inline]
```

## 6.107.3 Member Function Documentation

### 6.107.3.1 get()

```
const std::vector< unsigned > & GiNaC::partition_generator::get () const [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [partition](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::x](#).

### 6.107.3.2 next()

```
bool GiNaC::partition_generator::next () [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::next\\_pa](#)

## 6.107.4 Member Data Documentation

### 6.107.4.1 partition

```
std::vector<unsigned> GiNaC::partition_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#).

### 6.107.4.2 current\_updated

```
bool GiNaC::partition_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

The documentation for this class was generated from the following file:

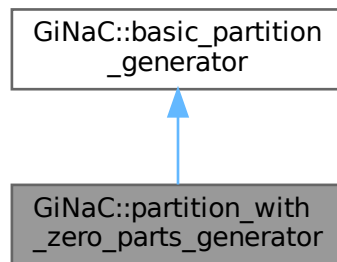
- [utils.h](#)

## 6.108 GiNaC::partition\_with\_zero\_parts\_generator Class Reference

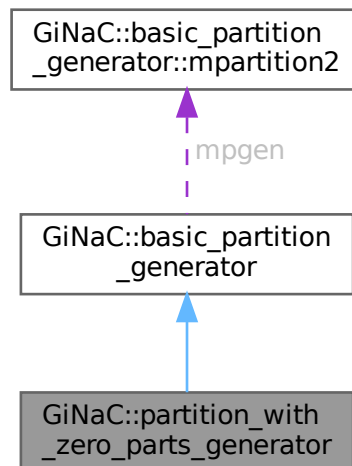
Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.

```
#include <utils.h>
```

Inheritance diagram for GiNaC::partition\_with\_zero\_parts\_generator:



Collaboration diagram for GiNaC::partition\_with\_zero\_parts\_generator:



### Public Member Functions

- [partition\\_with\\_zero\\_parts\\_generator](#) (unsigned  $n$ \_, unsigned  $m$ \_)
- `const std::vector< unsigned > & get () const`
- `bool next ()`

**Private Attributes**

- unsigned [m](#)
- std::vector< unsigned > [partition](#)
- bool [current\\_updated](#)

**Additional Inherited Members****Protected Member Functions inherited from [GiNaC::basic\\_partition\\_generator](#)**

- [basic\\_partition\\_generator](#) (unsigned *n\_*, unsigned *m\_*)

**Protected Attributes inherited from [GiNaC::basic\\_partition\\_generator](#)**

- [mpartition2](#) [mpgen](#)

**6.108.1 Detailed Description**

Generate all bounded combinatorial partitions of an integer *n* with exactly *m* parts (including zero parts) in non-decreasing order.

**6.108.2 Constructor & Destructor Documentation****6.108.2.1 [partition\\_with\\_zero\\_parts\\_generator\(\)](#)**

```
GiNaC::partition_with_zero_parts_generator::partition_with_zero_parts_generator (
 unsigned n_,
 unsigned m_) [inline]
```

**6.108.3 Member Function Documentation****6.108.3.1 [get\(\)](#)**

```
const std::vector< unsigned > & GiNaC::partition_with_zero_parts_generator::get () const
[inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [partition](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::x](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

**6.108.3.2 [next\(\)](#)**

```
bool GiNaC::partition_with_zero_parts_generator::next () [inline]
```

References [current\\_updated](#), [GiNaC::basic\\_partition\\_generator::mpartition2::m](#), [m](#), [GiNaC::basic\\_partition\\_generator::mpgen](#), [GiNaC::basic\\_partition\\_generator::mpartition2::n](#), and [GiNaC::basic\\_partition\\_generator::mpartition2::next\\_partition\(\)](#).

Referenced by [GiNaC::power::expand\\_add\(\)](#).

## 6.108.4 Member Data Documentation

### 6.108.4.1 m

```
unsigned GiNaC::partition_with_zero_parts_generator::m [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

### 6.108.4.2 partition

```
std::vector<unsigned> GiNaC::partition_with_zero_parts_generator::partition [mutable], [private]
```

Referenced by [get\(\)](#).

### 6.108.4.3 current\_updated

```
bool GiNaC::partition_with_zero_parts_generator::current_updated [mutable], [private]
```

Referenced by [get\(\)](#), and [next\(\)](#).

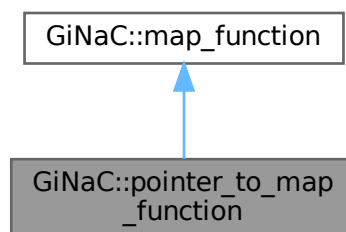
The documentation for this class was generated from the following file:

- [utils.h](#)

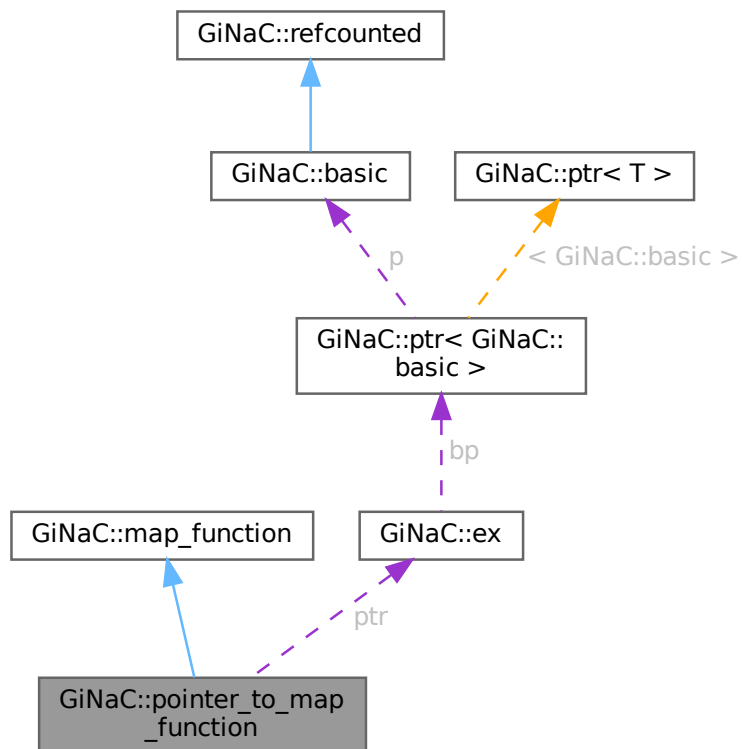
## 6.109 GiNaC::pointer\_to\_map\_function Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function:



Collaboration diagram for GiNaC::pointer\_to\_map\_function:



### Public Member Functions

- [pointer\\_to\\_map\\_function](#) ([ex](#) x(const [ex](#) &))
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &)

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.109.1 Constructor & Destructor Documentation

### 6.109.1.1 `pointer_to_map_function()`

```
GiNaC::pointer_to_map_function::pointer_to_map_function (
 ex xconst ex &) [inline], [explicit]
```

## 6.109.2 Member Function Documentation

### 6.109.2.1 `operator>()()`

```
ex GiNaC::pointer_to_map_function::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [ptr](#).

## 6.109.3 Member Data Documentation

### 6.109.3.1 `ptr`

```
ex(* GiNaC::pointer_to_map_function::ptr) (const ex &) [protected]
```

Referenced by [operator>\(\)\(\)](#).

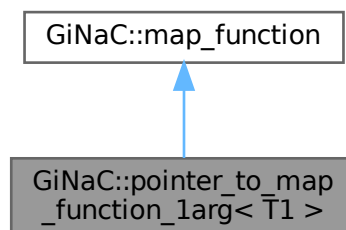
The documentation for this class was generated from the following file:

- [ex.h](#)

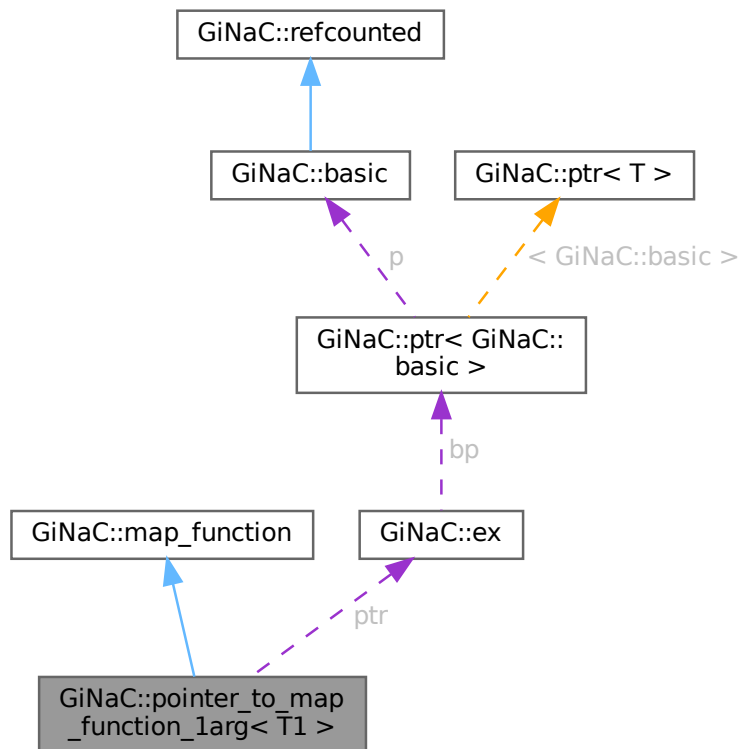
## 6.110 `GiNaC::pointer_to_map_function_1arg< T1 >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_map_function_1arg< T1 >`:



Collaboration diagram for GiNaC::pointer\_to\_map\_function\_1arg< T1 >:



### Public Member Functions

- [pointer\\_to\\_map\\_function\\_1arg](#) ([ex](#) x(const [ex](#) &, T1), T1 a1)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &, T1)
- T1 [arg1](#)

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.110.1 Constructor & Destructor Documentation

### 6.110.1.1 `pointer_to_map_function_1arg()`

```
template<class T1 >
GiNaC::pointer_to_map_function_1arg< T1 >::pointer_to_map_function_1arg (
 ex xconst ex &, T1,
 T1 a1) [inline], [explicit]
```

## 6.110.2 Member Function Documentation

### 6.110.2.1 `operator>()()`

```
template<class T1 >
ex GiNaC::pointer_to_map_function_1arg< T1 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::arg1](#), and [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::ptr](#).

## 6.110.3 Member Data Documentation

### 6.110.3.1 `ptr`

```
template<class T1 >
ex(* GiNaC::pointer_to_map_function_1arg< T1 >::ptr) (const ex &, T1) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::operator>\(\)\(\)](#).

### 6.110.3.2 `arg1`

```
template<class T1 >
T1 GiNaC::pointer_to_map_function_1arg< T1 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >::operator>\(\)\(\)](#).

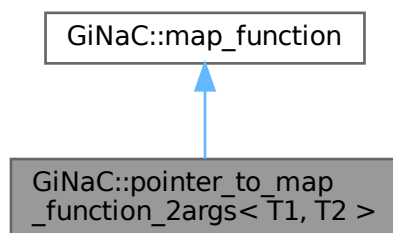
The documentation for this class was generated from the following file:

- [ex.h](#)

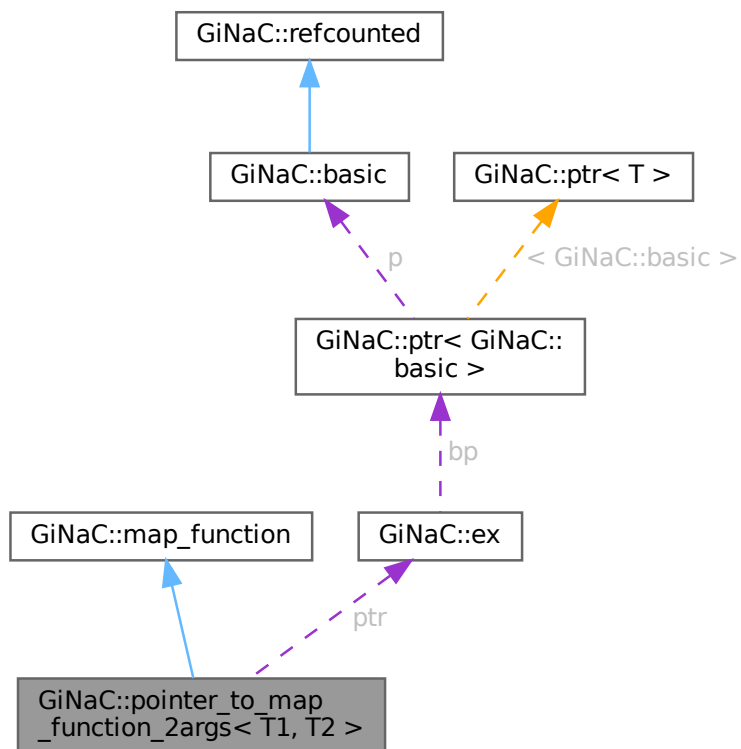
## 6.111 GiNaC::pointer\_to\_map\_function\_2args< T1, T2 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



Collaboration diagram for GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >:



## Public Member Functions

- [pointer\\_to\\_map\\_function\\_2args](#) ([ex](#) x(const [ex](#) &, T1, T2), T1 a1, T2 a2)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Protected Attributes

- [ex](#)(\* [ptr](#) )(const [ex](#) &, T1, T2)
- T1 [arg1](#)
- T2 [arg2](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.111.1 Constructor & Destructor Documentation

### 6.111.1.1 [pointer\\_to\\_map\\_function\\_2args\(\)](#)

```
template<class T1 , class T2 >
GiNaC::pointer_to_map_function_2args< T1, T2 >::pointer_to_map_function_2args (
 ex xconst ex &, T1, T2,
 T1 a1,
 T2 a2) [inline], [explicit]
```

## 6.111.2 Member Function Documentation

### 6.111.2.1 [operator>\(\)\(\)](#)

```
template<class T1 , class T2 >
ex GiNaC::pointer_to_map_function_2args< T1, T2 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::arg2](#), and [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::ptr](#).

### 6.111.3 Member Data Documentation

#### 6.111.3.1 ptr

```
template<class T1 , class T2 >
ex(* GiNaC::pointer_to_map_function_2args< T1, T2 >::ptr) (const ex &, T1, T2) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

#### 6.111.3.2 arg1

```
template<class T1 , class T2 >
T1 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

#### 6.111.3.3 arg2

```
template<class T1 , class T2 >
T2 GiNaC::pointer_to_map_function_2args< T1, T2 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >::operator\(\)\(\)](#).

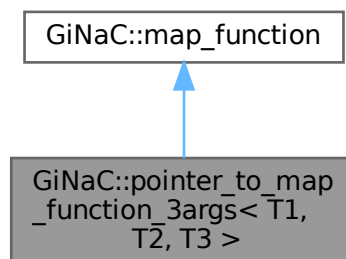
The documentation for this class was generated from the following file:

- [ex.h](#)

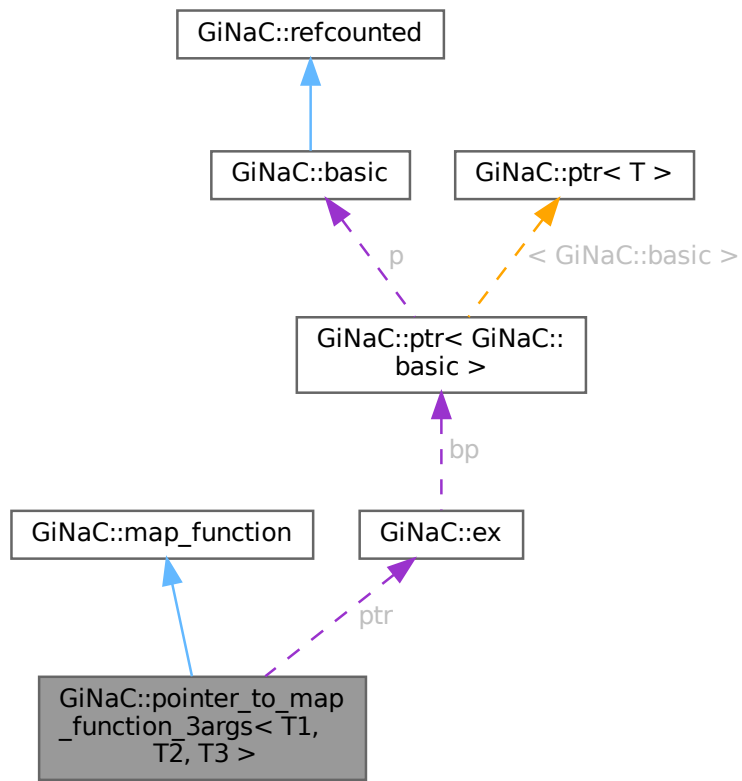
## 6.112 GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >:



Collaboration diagram for `GiNaC::pointer_to_map_function_3args< T1, T2, T3 >`:



### Public Member Functions

- `pointer_to_map_function_3args` (`ex x(const ex &, T1, T2, T3), T1 a1, T2 a2, T3 a3`)
- `ex operator()` (`const ex &e`) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`

### Protected Attributes

- `ex(* ptr)(const ex &, T1, T2, T3)`
- `T1 arg1`
- `T2 arg2`
- `T3 arg3`

### Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef `const ex & argument_type`
- typedef `ex result_type`

## 6.112.1 Constructor & Destructor Documentation

### 6.112.1.1 pointer\_to\_map\_function\_3args()

```
template<class T1 , class T2 , class T3 >
GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::pointer_to_map_function_3args (
 ex xconst ex &, T1, T2, T3,
 T1 a1,
 T2 a2,
 T3 a3) [inline], [explicit]
```

## 6.112.2 Member Function Documentation

### 6.112.2.1 operator>()()

```
template<class T1 , class T2 , class T3 >
ex GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::a](#), [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::ptr](#).

## 6.112.3 Member Data Documentation

### 6.112.3.1 ptr

```
template<class T1 , class T2 , class T3 >
ex(* GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::ptr) (const ex &, T1, T2, T3) [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator>\(\)\(\)](#).

### 6.112.3.2 arg1

```
template<class T1 , class T2 , class T3 >
T1 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator>\(\)\(\)](#).

### 6.112.3.3 arg2

```
template<class T1 , class T2 , class T3 >
T2 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator>\(\)\(\)](#).

#### 6.112.3.4 `arg3`

```
template<class T1 , class T2 , class T3 >
T3 GiNaC::pointer_to_map_function_3args< T1, T2, T3 >::arg3 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >::operator\(\)](#).

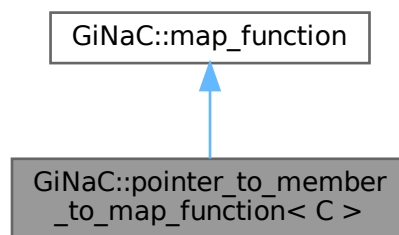
The documentation for this class was generated from the following file:

- [ex.h](#)

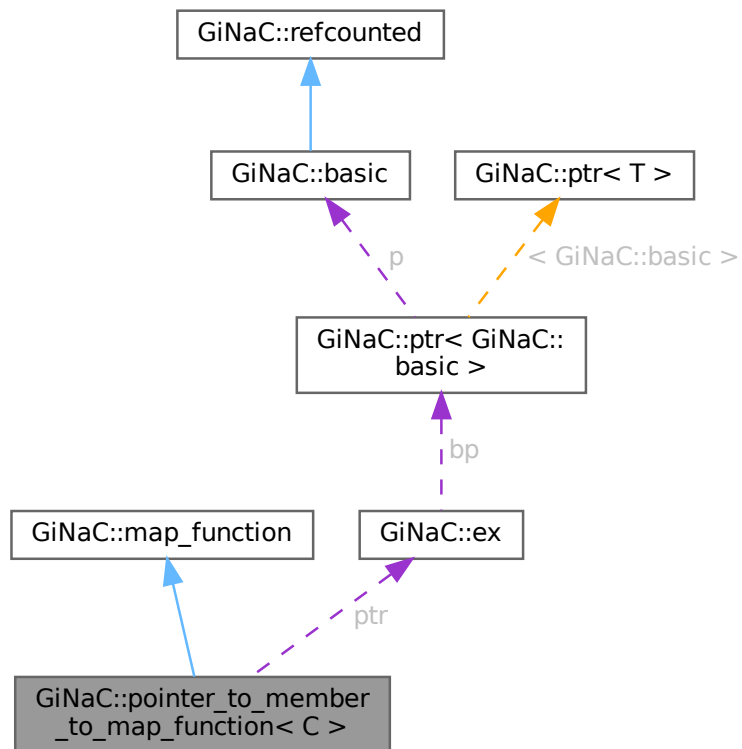
### 6.113 `GiNaC::pointer_to_member_to_map_function< C >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function< C >`:



Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function< C >:



### Public Member Functions

- `pointer_to_member_to_map_function` (`ex(C::*member)(const ex &), C &obj`)
- `ex operator()` (`const ex &e`) override

### Public Member Functions inherited from `GiNaC::map_function`

- virtual `~map_function()`

### Protected Attributes

- `ex(C::* ptr)(const ex &)`
- `C & c`

### Additional Inherited Members

### Public Types inherited from `GiNaC::map_function`

- typedef const `ex` & `argument_type`
- typedef `ex` `result_type`

## 6.113.1 Constructor & Destructor Documentation

### 6.113.1.1 `pointer_to_member_to_map_function()`

```
template<class C >
GiNaC::pointer_to_member_to_map_function< C >::pointer_to_member_to_map_function (
 ex(C::*)(const ex &) member,
 C & obj) [inline], [explicit]
```

## 6.113.2 Member Function Documentation

### 6.113.2.1 `operator>()()`

```
template<class C >
ex GiNaC::pointer_to_member_to_map_function< C >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::c](#).

## 6.113.3 Member Data Documentation

### 6.113.3.1 `ptr`

```
template<class C >
ex(C::* GiNaC::pointer_to_member_to_map_function< C >::ptr) (const ex &) [protected]
```

### 6.113.3.2 `c`

```
template<class C >
C& GiNaC::pointer_to_member_to_map_function< C >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >::operator>\(\)\(\)](#).

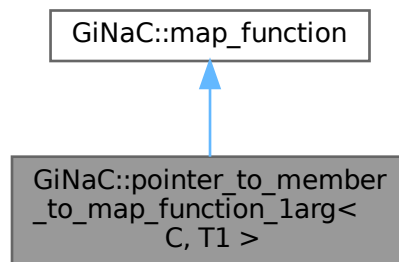
The documentation for this class was generated from the following file:

- [ex.h](#)

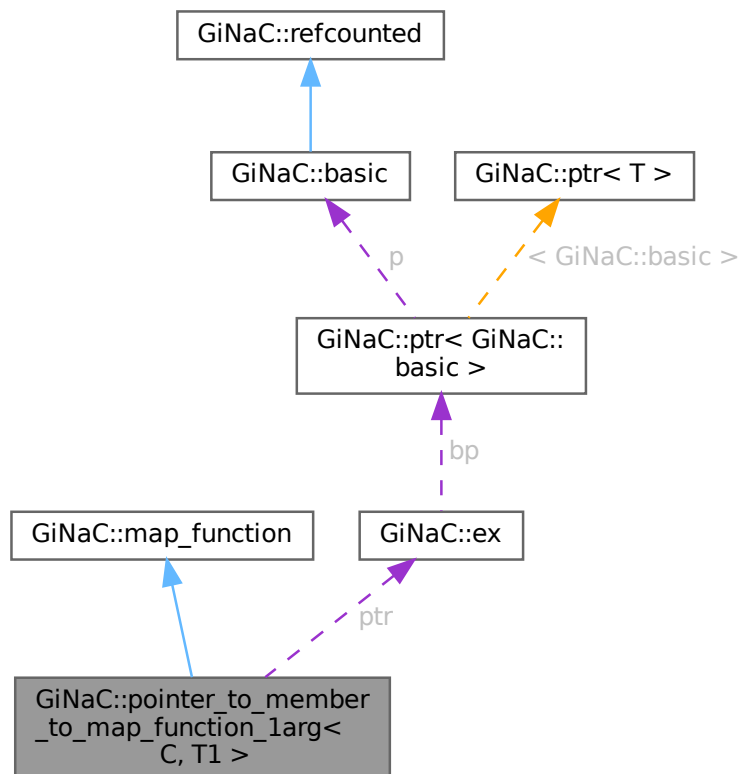
## 6.114 GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >:



Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >:



## Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#) ([ex](#)(C::[\\*member](#))(const [ex](#) &, T1), C &obj, T1 a1)
- [ex operator\(\)](#) (const [ex](#) &e) override

## Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

## Protected Attributes

- [ex](#)(C::[\\* ptr](#))(const [ex](#) &, T1)
- C & [c](#)
- T1 [arg1](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.114.1 Constructor & Destructor Documentation

### 6.114.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_1arg\(\)](#)

```
template<class C , class T1 >
GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::pointer_to_member_to_map_function_1arg
(
 ex(C::* member)(const ex &, T1) member,
 C & obj,
 T1 a1) [inline], [explicit]
```

## 6.114.2 Member Function Documentation

### 6.114.2.1 [operator\(\)](#)()

```
template<class C , class T1 >
ex GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::arg1](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg](#).

### 6.114.3 Member Data Documentation

#### 6.114.3.1 ptr

```
template<class C , class T1 >
ex(C::* GiNaC::pointer_to_member_to_map_function_larg< C, T1 >::ptr) (const ex &, T1) [protected]
```

#### 6.114.3.2 c

```
template<class C , class T1 >
C& GiNaC::pointer_to_member_to_map_function_larg< C, T1 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::operator\(\)\(\)](#).

#### 6.114.3.3 arg1

```
template<class C , class T1 >
T1 GiNaC::pointer_to_member_to_map_function_larg< C, T1 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >::operator\(\)\(\)](#).

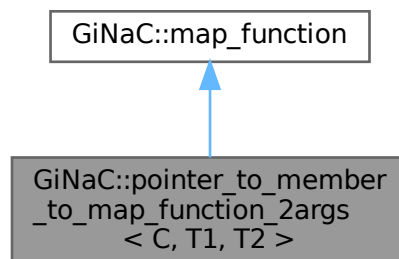
The documentation for this class was generated from the following file:

- [ex.h](#)

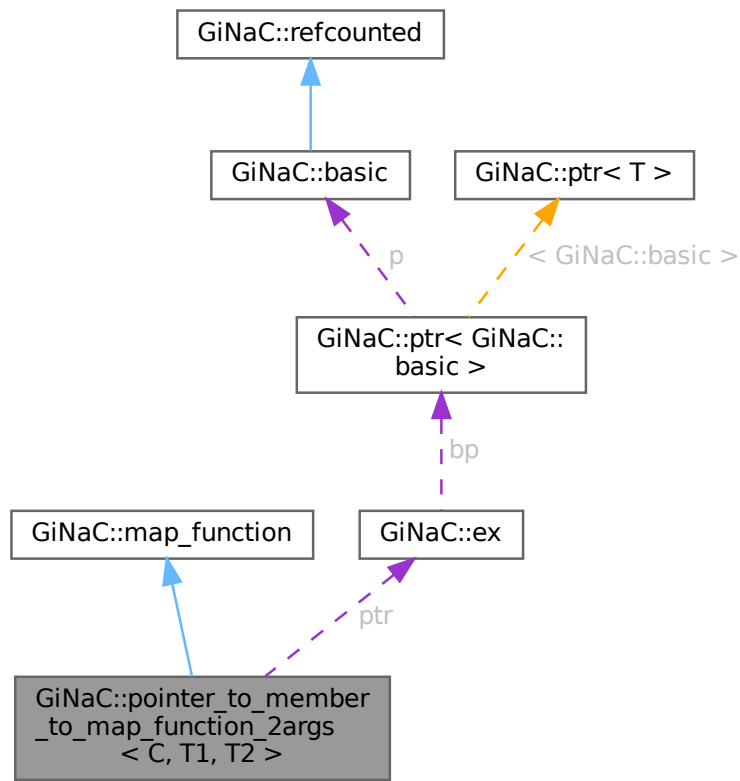
## 6.115 GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 > Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >:



Collaboration diagram for `GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >`:



### Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_2args](#) (`ex(C::*member)(const ex &, T1, T2), C &obj, T1 a1, T2 a2`)
- [ex operator\(\)](#) (`const ex &e`) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- `ex(C::* ptr)` (`const ex &, T1, T2`)
- `C & c`
- `T1 arg1`
- `T2 arg2`

### Additional Inherited Members

### Public Types inherited from [GiNaC::map\\_function](#)

- typedef `const ex & argument_type`
- typedef `ex result_type`

## 6.115.1 Constructor & Destructor Documentation

### 6.115.1.1 pointer\_to\_member\_to\_map\_function\_2args()

```
template<class C , class T1 , class T2 >
GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::pointer_to_member_to_map_function_2args (
 ex(C::*)(const ex &, T1, T2) member,
 C & obj,
 T1 a1,
 T2 a2) [inline], [explicit]
```

## 6.115.2 Member Function Documentation

### 6.115.2.1 operator>()()

```
template<class C , class T1 , class T2 >
ex GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::c](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::c](#).

## 6.115.3 Member Data Documentation

### 6.115.3.1 ptr

```
template<class C , class T1 , class T2 >
ex(C::* GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::ptr) (const ex &, T1, T2)
[protected]
```

### 6.115.3.2 c

```
template<class C , class T1 , class T2 >
C& GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator>\(\)](#).

### 6.115.3.3 arg1

```
template<class C , class T1 , class T2 >
T1 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator>\(\)](#).

#### 6.115.3.4 `arg2`

```
template<class C , class T1 , class T2 >
T2 GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >::operator\(\)](#).

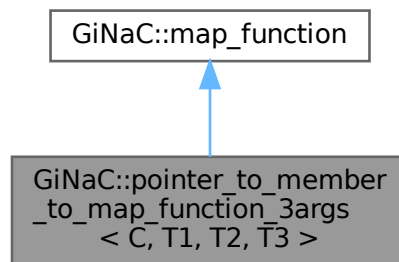
The documentation for this class was generated from the following file:

- [ex.h](#)

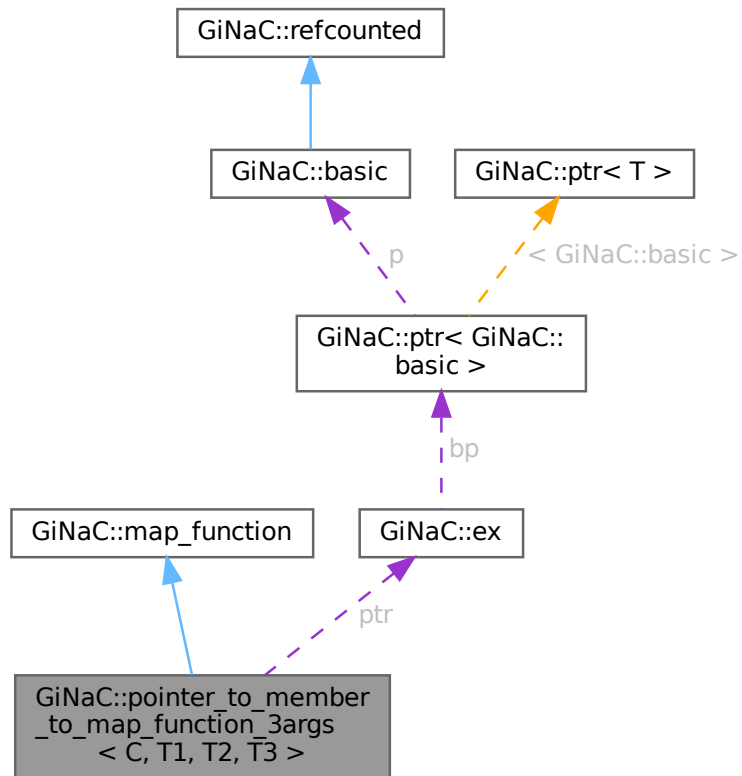
### 6.116 `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >` Class Template Reference

```
#include <ex.h>
```

Inheritance diagram for `GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >`:



Collaboration diagram for GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >:



### Public Member Functions

- [pointer\\_to\\_member\\_to\\_map\\_function\\_3args](#) ([ex](#)(C::\*member)(const [ex](#) &, T1, T2, T3), C &obj, T1 a1, T2 a2, T3 a3)
- [ex operator\(\)](#) (const [ex](#) &e) override

### Public Member Functions inherited from [GiNaC::map\\_function](#)

- virtual [~map\\_function](#) ()

### Protected Attributes

- [ex](#)(C::\* [ptr](#) )(const [ex](#) &, T1, T2, T3)
- C & [c](#)
- T1 [arg1](#)
- T2 [arg2](#)
- T3 [arg3](#)

## Additional Inherited Members

## Public Types inherited from [GiNaC::map\\_function](#)

- typedef const [ex](#) & [argument\\_type](#)
- typedef [ex](#) [result\\_type](#)

## 6.116.1 Constructor & Destructor Documentation

### 6.116.1.1 [pointer\\_to\\_member\\_to\\_map\\_function\\_3args\(\)](#)

```
template<class C , class T1 , class T2 , class T3 >
GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::pointer_to_member_to_map_←
function_3args (
 ex(C::*)(const ex &, T1, T2, T3) member,
 C & obj,
 T1 a1,
 T2 a2,
 T3 a3) [inline], [explicit]
```

## 6.116.2 Member Function Documentation

### 6.116.2.1 [operator>\(\)\(\)](#)

```
template<class C , class T1 , class T2 , class T3 >
ex GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::operator() (
 const ex & e) [inline], [override], [virtual]
```

Implements [GiNaC::map\\_function](#).

References [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg1](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg2](#), [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::arg3](#), and [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)](#).

## 6.116.3 Member Data Documentation

### 6.116.3.1 [ptr](#)

```
template<class C , class T1 , class T2 , class T3 >
ex(C::* GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::ptr) (const ex &, T1,
T2, T3) [protected]
```

### 6.116.3.2 [c](#)

```
template<class C , class T1 , class T2 , class T3 >
C& GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::c [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator>\(\)\(\)](#).

### 6.116.3.3 arg1

```
template<class C , class T1 , class T2 , class T3 >
T1 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg1 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

### 6.116.3.4 arg2

```
template<class C , class T1 , class T2 , class T3 >
T2 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg2 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

### 6.116.3.5 arg3

```
template<class C , class T1 , class T2 , class T3 >
T3 GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >::arg3 [protected]
```

Referenced by [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >::operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

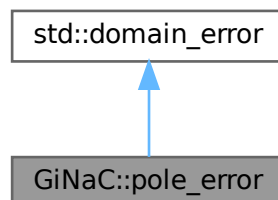
- [ex.h](#)

## 6.117 GiNaC::pole\_error Class Reference

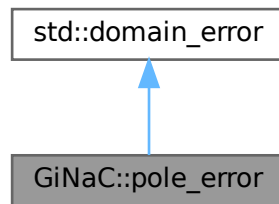
Exception class thrown when a singularity is encountered.

```
#include <numeric.h>
```

Inheritance diagram for GiNaC::pole\_error:



Collaboration diagram for `GiNaC::pole_error`:



### Public Member Functions

- `pole_error` (const std::string &what\_arg, int degree)  
*ctor for `pole_error` exception class.*
- int `degree` () const  
*Return the degree of the `pole_error` exception class.*

### Private Attributes

- int `deg`

## 6.117.1 Detailed Description

Exception class thrown when a singularity is encountered.

## 6.117.2 Constructor & Destructor Documentation

### 6.117.2.1 pole\_error()

```
GiNaC::pole_error::pole_error (
 const std::string & what_arg,
 int degree) [explicit]
```

ctor for `pole_error` exception class.

## 6.117.3 Member Function Documentation

### 6.117.3.1 degree()

```
int GiNaC::pole_error::degree () const
```

Return the degree of the `pole_error` exception class.

References `deg`.

## 6.117.4 Member Data Documentation

### 6.117.4.1 deg

```
int GiNaC::pole_error::deg [private]
```

Referenced by [degree\(\)](#).

The documentation for this class was generated from the following files:

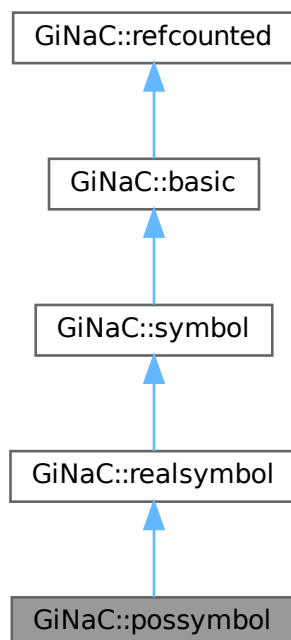
- [numeric.h](#)
- [utils.cpp](#)

## 6.118 GiNaC::possymbol Class Reference

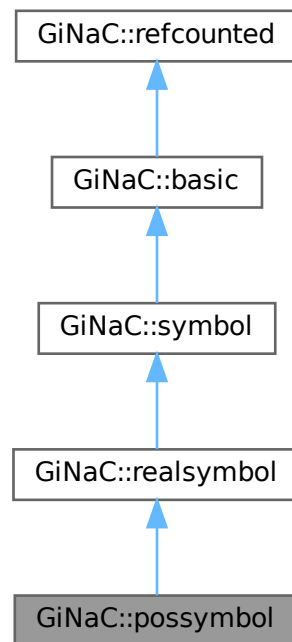
Specialization of symbol to real positive domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::possymbol:



Collaboration diagram for GiNaC::possymbol:



### Public Member Functions

- [possymbol](#) ()
- [possymbol](#) (const std::string &initname)
- [possymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [possymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

### Public Member Functions inherited from [GiNaC::realsymbol](#)

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

## Public Member Functions inherited from GiNaC::symbol

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

## Public Member Functions inherited from GiNaC::basic

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*

- virtual void `dbgprinttree` () const  
*Little wrapper around `printtree` to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*

- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Additional Inherited Members

## Protected Member Functions inherited from `GiNaC::symbol`

- `ex derivative` (const `symbol` &`s`) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- `bool is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- `unsigned calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- `void do_print` (const `print_context` &`c`, unsigned `level`) const
- `void do_print_latex` (const `print_latex` &`c`, unsigned `level`) const
- `void do_print_tree` (const `print_tree` &`c`, unsigned `level`) const
- `void do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &`v`) const
- `virtual bool match_same_type` (const `basic` &`other`) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual int compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- `void ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- `void do_print` (const `print_context` &`c`, unsigned `level`) const  
*Default output to stream.*
- `void do_print_tree` (const `print_tree` &`c`, unsigned `level`) const  
*Tree output to stream.*
- `void do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const  
*Python parsable output to stream.*

## Protected Attributes inherited from [GiNaC::symbol](#)

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.118.1 Detailed Description

Specialization of symbol to real positive domain.

### 6.118.2 Constructor & Destructor Documentation

#### 6.118.2.1 `possymbol()` [1/3]

```
GiNaC::possymbol::possymbol ()
```

Referenced by [duplicate\(\)](#).

#### 6.118.2.2 `possymbol()` [2/3]

```
GiNaC::possymbol::possymbol (
 const std::string & initname) [explicit]
```

#### 6.118.2.3 `possymbol()` [3/3]

```
GiNaC::possymbol::possymbol (
 const std::string & initname,
 const std::string & texname)
```

### 6.118.3 Member Function Documentation

#### 6.118.3.1 `get_domain()`

```
unsigned GiNaC::possymbol::get_domain () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

References [GiNaC::domain::positive](#).

### 6.118.3.2 duplicate()

```
possymbol * GiNaC::possymbol::duplicate () const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::dynallocated](#), [possymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

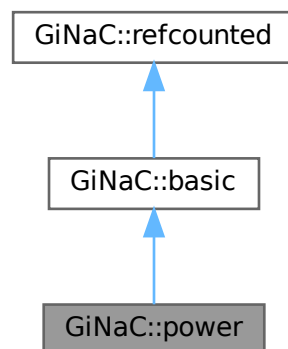
- [symbol.h](#)
- [symbol.cpp](#)

## 6.119 GiNaC::power Class Reference

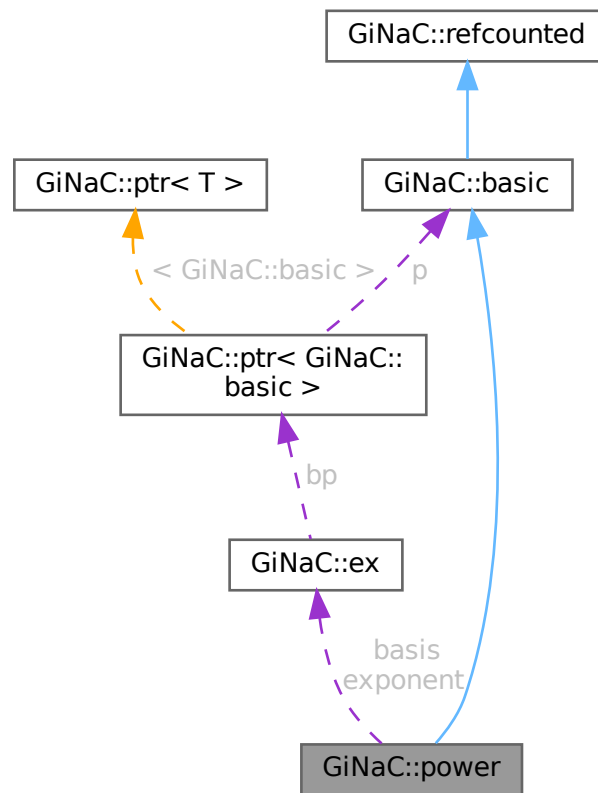
This class holds a two-component object, a basis and an exponent representing exponentiation.

```
#include <power.h>
```

Inheritance diagram for GiNaC::power:



Collaboration diagram for `GiNaC::power`:



## Public Member Functions

- `power` (const `ex` &lh, const `ex` &rh)
- `template<typename T>`  
`power` (const `ex` &lh, const T &rh)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t i) const override  
*Return operand/member at position i.*
- `ex map` (`map_function` &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- bool `is_polynomial` (const `ex` &var) const override  
*Check whether this is a polynomial in the given variables.*
- int `degree` (const `ex` &s) const override  
*Return degree of highest power in object s.*

- `int ldegree (const ex &s) const` override  
*Return degree of lowest power in object s.*
- `ex coeff (const ex &s, int n=1) const` override  
*Return coefficient of degree n in object s.*
- `ex eval ()` const override  
*Perform automatic term rewriting rules in this class.*
- `ex evalf ()` const override  
*Evaluate object numerically.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `ex series (const relational &s, int order, unsigned options=0) const` override  
*Implementation of `ex::series()` for powers.*
- `ex subs (const exmap &m, unsigned options=0) const` override  
*Substitute a set of objects by arbitrary expressions.*
- `bool has (const ex &other, unsigned options=0) const` override  
*Test for occurrence of a pattern.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const` override  
*Implementation of `ex::normal()` for powers.*
- `ex to_rational (exmap &repl) const` override  
*Implementation of `ex::to_rational()` for powers.*
- `ex to_polynomial (exmap &repl) const` override  
*Implementation of `ex::to_polynomial()` for powers.*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `void archive (archive_node &n) const` override  
*Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &syms) const` override  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- `virtual void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- `virtual ex operator[] (const ex &index) const`

- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- template<class T >  
 void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned options) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for a power.*
- `ex eval_ncmul` (const `exvector` &v) const override
- unsigned `return_type` () const override
- `return_type_t return_type_tinfo` () const override
- `ex expand` (unsigned `options`=0) const override  
*Expand expression, i.e.*
- void `print_power` (const `print_context` &c, const char \*powersymbol, const char \*openbrace, const char \*closebrace, unsigned level) const
- void `do_print_dflt` (const `print_dflt` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_python` (const `print_python` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const
- void `do_print_csrc_cl_N` (const `print_csrc_cl_N` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Static Protected Member Functions

- static `ex expand_add` (const `add` &a, long n, unsigned `options`)  
*expand  $a^n$  where  $a$  is an add and  $n$  is a positive integer.*
- static `ex expand_add_2` (const `add` &a, unsigned `options`)  
*Special case of `power::expand_add`.*
- static `ex expand_mul` (const `mul` &m, const `numeric` &n, unsigned `options`, bool from\_expand=false)  
*Expand factors of  $m$  in  $m^n$  where  $m$  is a mul and  $n$  is an integer.*

## Protected Attributes

- `ex basis`
- `ex exponent`

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Friends

- class [mul](#)

## 6.119.1 Detailed Description

This class holds a two-component object, a basis and an exponent representing exponentiation.

## 6.119.2 Constructor & Destructor Documentation

### 6.119.2.1 `power()` [1/2]

```
GiNaC::power::power (
 const ex & lh,
 const ex & rh) [inline]
```

Referenced by [do\\_print\\_latex\(\)](#).

### 6.119.2.2 `power()` [2/2]

```
template<typename T >
GiNaC::power::power (
 const ex & lh,
 const T & rh) [inline]
```

## 6.119.3 Member Function Documentation

### 6.119.3.1 `precedence()`

```
unsigned GiNaC::power::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_power\(\)](#).

### 6.119.3.2 info()

```
bool GiNaC::power::info (
 unsigned int) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::even](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::info\\_flags::expanded](#), [exponent](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::has\\_indices](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::status\\_flags::has\\_no\\_indices](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::info\\_flags::real](#), and [GiNaC::basic::setflag\(\)](#).

### 6.119.3.3 nops()

```
size_t GiNaC::power::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.119.3.4 op()

```
ex GiNaC::power::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [GINAC\\_ASSERT](#).

### 6.119.3.5 map()

```
ex GiNaC::power::map (
 map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), and [exponent](#).

### 6.119.3.6 is\_polynomial()

```
bool GiNaC::power::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_polynomial\(\)](#), and [GiNaC::info\\_flags::nonnegint](#).

### 6.119.3.7 degree()

```
int GiNaC::power::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::degree\(\)](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::is\\_integer\(\)](#).

### 6.119.3.8 ldegree()

```
int GiNaC::power::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::has\(\)](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), and [GiNaC::ex::ldegree\(\)](#).

### 6.119.3.9 coeff()

```
ex GiNaC::power::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::basic::is\\_equal\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), and [n](#).

Referenced by [expand\(\)](#), and [expand\\_add\(\)](#).

## 6.119.3.10 eval()

```
ex GiNaC::power::eval () const [override], [virtual]
```

Perform automatic term rewriting rules in this class.

In the following  $x, x_1, x_2, \dots$  stand for a symbolic variables of type `ex` and  $c, c_1, c_2, \dots$  stand for such expressions that contain a plain number.

- $x^0 \rightarrow 1$  (also handles  $0^0$ )
- $x^1 \rightarrow x$
- $0^c \rightarrow 0$  or exception (depending on the real part of  $c$ )
- $1^x \rightarrow 1$
- $c_1^{c_2} \rightarrow (c_1^n c_1^{c_2-n})$  (so that  $0 < (c_2-n) < 1$ , try to evaluate roots, possibly in numerator and denominator of  $c_1$ )
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_1 c_2)}$  if  $x$  is positive and  $c_1$  is real.
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_1 c_2)}$  ( $c_2$  integer or  $-1 < c_1 \leq 1$  or ( $c_1 = -1$  and  $c_2 > 0$ ), case  $c_1 = 1$  should not happen, see below!)
- $x^{(x, y, z), c} \rightarrow (x^c y^c z^c)$  (if  $c$  integer)
- $x^{(x, c_1), c_2} \rightarrow x^{(x, c_2)} c_1^{c_2}$  ( $c_1 > 0$ )
- $x^{(x, c_1), c_2} \rightarrow x^{(-x, c_2)} c_1^{c_2}$  ( $c_1 < 0$ )

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex0](#), [GiNaC::\\_ex1](#), [GiNaC::ex\\_1](#), [GiNaC::\\_num0\\_p](#), [GiNaC::\\_num1\\_p](#), [GiNaC::\\_num\\_1\\_p](#), [GiNaC::abs\(\)](#), [basis](#), [c](#), [GiNaC::basic::clearflag\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::return\\_types::commutative](#), [GiNaC::numeric::compare\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::numeric::div\(\)](#), [GiNaC::status\\_flags::evaluated](#), [expand\\_mul\(\)](#), [exponent](#), [GiNaC::basic::flags](#), [GINAC\\_ASSERT](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hold\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::integer\\_content\(\)](#), [GiNaC::numeric::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::numeric::is\\_crational\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::numeric::is\\_equal\(\)](#), [GiNaC::numeric::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::numeric::is\\_pos\\_integ](#), [GiNaC::numeric::is\\_positive\(\)](#), [GiNaC::numeric::is\\_rational\(\)](#), [GiNaC::numeric::is\\_real\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [likely](#), [m](#), [GiNaC::numeric::mul\(\)](#), [n](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::numeric::power\(\)](#), [r](#), [GiNaC::info\\_flags::real](#), [GiNaC::numeric::real\(\)](#), [GiNaC::ex::return\\_type\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

## 6.119.3.11 evalf()

```
ex GiNaC::power::evalf () const [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalf\(\)](#), and [exponent](#).

**6.119.3.12 evalm()**

```
ex GiNaC::power::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [GiNaC::ex::evalm\(\)](#), [exponent](#), and [GiNaC::pow\(\)](#).

**6.119.3.13 series()**

```
ex GiNaC::power::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for powers.

This performs Laurent expansion of reciprocals of series at singularities.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [GiNaC::exp\(\)](#), [GiNaC::ex::expand\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::ex::lddegree\(\)](#), [GiNaC::ex::lhs\(\)](#), [GiNaC::log\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [order](#), [r](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::ex::rhs\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::to\\_int\(\)](#).

Referenced by [GiNaC::psi1\\_series\(\)](#).

**6.119.3.14 subs()**

```
ex GiNaC::power::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::subs\\_options::algebraic](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [exponent](#), [m](#), [GiNaC::subs\\_options::no\\_pattern](#), [options](#), [GiNaC::pow\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), and [GiNaC::tryfactsubs\(\)](#).

**6.119.3.15 has()**

```
bool GiNaC::power::has (
 const ex & pattern,
 unsigned options = 0) const [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::has\\_options::algebraic](#), [basis](#), [exponent](#), [GiNaC::basic::has\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::match\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::ex::op\(\)](#), [options](#), and [GiNaC::info\\_flags::posint](#).

**6.119.3.16 normal()**

```
ex GiNaC::power::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for powers.

It normalizes the basis, distributes integer exponents to numerator and denominator, and replaces non-integer powers by temporary symbols.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::container< C >::nops\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::ex::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::subs\(\)](#).

**6.119.3.17 to\_rational()**

```
ex GiNaC::power::to_rational (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for powers.

It replaces non-integer powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::to\\_rational\(\)](#).

**6.119.3.18 to\_polynomial()**

```
ex GiNaC::power::to_polynomial (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for powers.

It replaces non-posint powers by temporary symbols.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::collect\\_common\\_factors\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::negint](#), [GiNaC::info\\_flags::posint](#), [GiNaC::pow\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), and [GiNaC::ex::to\\_polynomial\(\)](#).

**6.119.3.19 conjugate()**

```
ex GiNaC::power::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [GiNaC::ex::conjugate\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), and [GiNaC::info\\_flags::positive](#).

**6.119.3.20 real\_part()**

```
ex GiNaC::power::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::cos\(\)](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), and [GiNaC::ex::real\\_part\(\)](#).

**6.119.3.21 imag\_part()**

```
ex GiNaC::power::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::abs\(\)](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::imag\\_part\(\)](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::log\(\)](#), [n](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::numer\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::ex::real\\_part\(\)](#), and [GiNaC::sin\(\)](#).

**6.119.3.22 archive()**

```
void GiNaC::power::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

### 6.119.3.23 read\_archive()

```
void GiNaC::power::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [basis](#), [exponent](#), and [n](#).

### 6.119.3.24 derivative()

```
ex GiNaC::power::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [basis](#), [GiNaC::ex::diff\(\)](#), [exponent](#), [GiNaC::log\(\)](#), and [GiNaC::pow\(\)](#).

### 6.119.3.25 eval\_ncmul()

```
ex GiNaC::power::eval_ncmul (
 const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.119.3.26 return\_type()

```
unsigned GiNaC::power::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\(\)](#).

### 6.119.3.27 return\_type\_tinfo()

```
return_type_t GiNaC::power::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [basis](#), and [GiNaC::ex::return\\_type\\_tinfo\(\)](#).

### 6.119.3.28 `expand()`

```
ex GiNaC::power::expand (
 unsigned options = 0) const [override], [protected], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [basis](#), [coeff\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::ex::expand\(\)](#), [expand\\_add\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::indefinite](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_integer\(\)](#), [m](#), [GiNaC::info\\_flags::negative](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [GiNaC::status\\_flags::purely\\_indefinite](#), [r](#), [GiNaC::add::recombine\\_pair\\_to\\_ex\(\)](#), [GiNaC::expairseq::seq](#), [GiNaC::basic::setflag\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

### 6.119.3.29 `print_power()`

```
void GiNaC::power::print_power (
 const print_context & c,
 const char * powersymbol,
 const char * openbrace,
 const char * closebrace,
 unsigned level) const [protected]
```

References [basis](#), [c](#), [exponent](#), [precedence\(\)](#), and [GiNaC::ex::print\(\)](#).

Referenced by [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

### 6.119.3.30 `do_print_dflt()`

```
void GiNaC::power::do_print_dflt (
 const print_dflt & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::print\(\)](#), and [print\\_power\(\)](#).

### 6.119.3.31 `do_print_latex()`

```
void GiNaC::power::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex1\\_2](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_negative\(\)](#), [power\(\)](#), [GiNaC::ex::print\(\)](#), and [print\\_power\(\)](#).

**6.119.3.32 do\_print\_csrc()**

```
void GiNaC::power::do_print_csrc (
 const print_csrc & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [c](#), [GiNaC::exp\(\)](#), [exponent](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::integer](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), and [GiNaC::numeric::to\\_int\(\)](#).

**6.119.3.33 do\_print\_python()**

```
void GiNaC::power::do_print_python (
 const print_python & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_power\(\)](#).

**6.119.3.34 do\_print\_python\_repr()**

```
void GiNaC::power::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [basis](#), [c](#), [exponent](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.35 do\_print\_csrc\_cl\_N()**

```
void GiNaC::power::do_print_csrc_cl_N (
 const print_csrc_cl_N & c,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex\\_1](#), [basis](#), [c](#), [exponent](#), [GiNaC::ex::is\\_equal\(\)](#), and [GiNaC::ex::print\(\)](#).

**6.119.3.36 expand\_add()**

```
ex GiNaC::power::expand_add (
 const add & a,
 long n,
 unsigned options) [static], [protected]
```

expand  $a^n$  where  $a$  is an [add](#) and  $n$  is a positive integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_num1\\_p](#), [basis](#), [GiNaC::binomial\(\)](#), [c](#), [coeff\(\)](#), [expand\(\)](#), [expand\\_add\\_2\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GiNaC::factor\(\)](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::get\(\)](#), [GiNaC::composition\\_generator::get\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [k](#), [mul](#), [GiNaC::multinomial\\_coefficient\(\)](#), [n](#), [GiNaC::partition\\_with\\_zero\\_parts\\_generator::next\(\)](#), [GiNaC::composition\\_generator::next\(\)](#), [GiNaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [GiNaC::pow\(\)](#), [r](#), [GiNaC::expairseq::seq](#), and [GiNaC::numeric::to\\_long\(\)](#).

Referenced by [expand\(\)](#).

### 6.119.3.37 `expand_add_2()`

```
ex GiNaC::power::expand_add_2 (
 const add & a,
 unsigned options) [static], [protected]
```

Special case of [power::expand\\_add](#).

Expands  $a^2$  where  $a$  is an `add`.

See also

[power::expand\\_add](#)

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_num2\\_p](#), [basis](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair](#), [expand\(\)](#), [expand\\_mul\(\)](#), [GiNaC::status\\_flags::expanded](#), [exponent](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [last](#), [GiNaC::numeric::mul\(\)](#), [mul](#), [GiNaC::numeric::mul\\_dyn\(\)](#), [GiNaC::expairseq::nops\(\)](#), [options](#), [GiNaC::expairseq::overall\\_coeff](#), [r](#), and [GiNaC::expairseq::seq](#).

Referenced by [expand\\_add\(\)](#).

### 6.119.3.38 `expand_mul()`

```
ex GiNaC::power::expand_mul (
 const mul & m,
 const numeric & n,
 unsigned options,
 bool from_expand = false) [static], [protected]
```

Expand factors of  $m$  in  $m^n$  where  $m$  is a `mul` and  $n$  is an integer.

See also

[power::expand](#)

References [GiNaC::\\_ex1](#), [GiNaC::expair::coeff](#), [GiNaC::basic::ex](#), [GiNaC::expand\\_options::expand\\_rename\\_idx](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::get\\_all\\_dummy\\_indices\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::has\\_indices](#), [m](#), [n](#), [options](#), [GiNaC::rename\\_dummy\\_indices\\_uniquely\(\)](#), and [GiNaC::basic::setflag\(\)](#).

Referenced by [eval\(\)](#), [expand\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.4 Friends And Related Symbol Documentation

### 6.119.4.1 `mul`

```
friend class mul [friend]
```

Referenced by [expand\\_add\(\)](#), and [expand\\_add\\_2\(\)](#).

## 6.119.5 Member Data Documentation

### 6.119.5.1 basis

ex GiNaC::power::basis [protected]

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

### 6.119.5.2 exponent

ex GiNaC::power::exponent [protected]

Referenced by [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_csrc\(\)](#), [do\\_print\\_csrc\\_cl\\_N\(\)](#), [do\\_print\\_dflt\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [expand\\_add\(\)](#), [expand\\_add\\_2\(\)](#), [has\(\)](#), [imag\\_part\(\)](#), [info\(\)](#), [is\\_polynomial\(\)](#), [ldegree\(\)](#), [map\(\)](#), [normal\(\)](#), [op\(\)](#), [print\\_power\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [GiNaC::mul::split\\_ex\\_to\\_pair\(\)](#), [subs\(\)](#), [to\\_polynomial\(\)](#), and [to\\_rational\(\)](#).

The documentation for this class was generated from the following files:

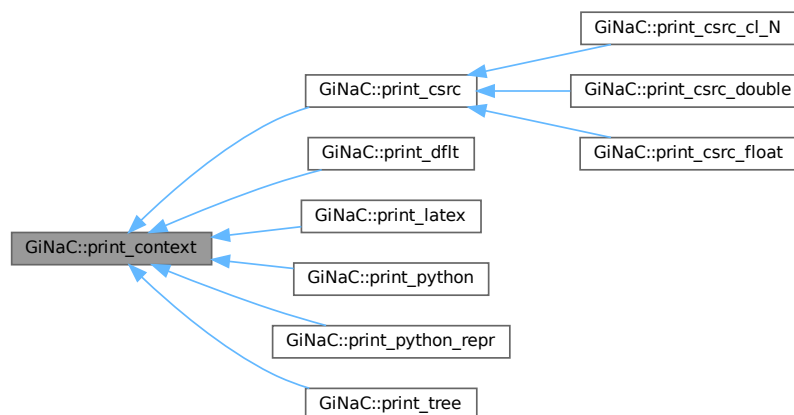
- [power.h](#)
- [normal.cpp](#)
- [power.cpp](#)
- [pseries.cpp](#)

## 6.120 GiNaC::print\_context Class Reference

Base class for print\_contexts.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_context:



## Public Member Functions

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Public Attributes

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 6.120.1 Detailed Description

Base class for print\_contexts.

## 6.120.2 Constructor & Destructor Documentation

### 6.120.2.1 print\_context()

```
GiNaC::print_context::print_context (
 std::ostream & os,
 unsigned options = 0)
```

### 6.120.2.2 ~print\_context()

```
virtual GiNaC::print_context::~~print_context () [inline], [virtual]
```

## 6.120.3 Member Data Documentation

### 6.120.3.1 s

```
std::ostream& GiNaC::print_context::s
```

stream to output to

Referenced by [GiNaC::function::print\(\)](#), and [GiNaC::numeric::print\\_numeric\(\)](#).

### 6.120.3.2 options

```
unsigned GiNaC::print_context::options
```

option flags

Referenced by [GiNaC::get\\_print\\_options\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), and [GiNaC::set\\_print\\_options\(\)](#).

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.121 GiNaC::print\_context\_options Class Reference

This class stores information about a registered [print\\_context](#) class.

```
#include <print.h>
```

### Public Member Functions

- [print\\_context\\_options](#) (const char \**n*, const char \**p*, unsigned *i*)
- const char \* [get\\_name](#) () const
- const char \* [get\\_parent\\_name](#) () const
- unsigned [get\\_id](#) () const

### Private Attributes

- const char \* [name](#)  
*Class name.*
- const char \* [parent\\_name](#)  
*Name of superclass.*
- unsigned [id](#)  
*ID number (assigned automatically).*

### 6.121.1 Detailed Description

This class stores information about a registered [print\\_context](#) class.

### 6.121.2 Constructor & Destructor Documentation

#### 6.121.2.1 print\_context\_options()

```
GiNaC::print_context_options::print_context_options (
 const char * n,
 const char * p,
 unsigned i) [inline]
```

### 6.121.3 Member Function Documentation

#### 6.121.3.1 get\_name()

```
const char * GiNaC::print_context_options::get_name () const [inline]
```

References [name](#).

### 6.121.3.2 `get_parent_name()`

```
const char * GiNaC::print_context_options::get_parent_name () const [inline]
```

References [parent\\_name](#).

### 6.121.3.3 `get_id()`

```
unsigned GiNaC::print_context_options::get_id () const [inline]
```

References [id](#).

## 6.121.4 Member Data Documentation

### 6.121.4.1 `name`

```
const char* GiNaC::print_context_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#).

### 6.121.4.2 `parent_name`

```
const char* GiNaC::print_context_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#).

### 6.121.4.3 `id`

```
unsigned GiNaC::print_context_options::id [private]
```

ID number (assigned automatically).

Referenced by [get\\_id\(\)](#).

The documentation for this class was generated from the following file:

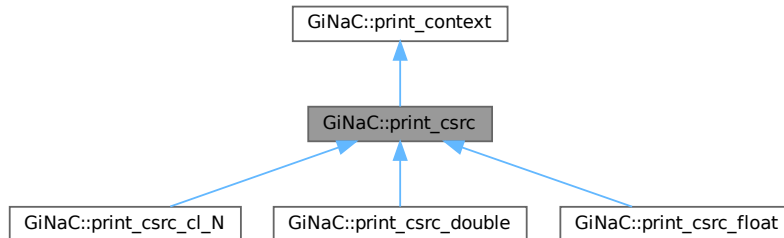
- [print.h](#)

## 6.122 GiNaC::print\_csrc Class Reference

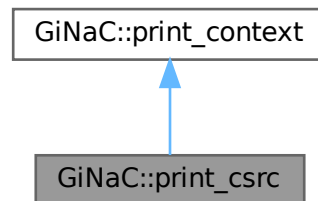
Base context for C source output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc:



Collaboration diagram for GiNaC::print\_csrc:



### Public Member Functions

- `print_csrc` (std::ostream &, unsigned `options`=0)

### Public Member Functions inherited from `GiNaC::print_context`

- `print_context` (std::ostream &, unsigned `options`=0)
- virtual `~print_context` ()

### Additional Inherited Members

### Public Attributes inherited from `GiNaC::print_context`

- std::ostream & `s`  
*stream to output to*
- unsigned `options`  
*option flags*

### 6.122.1 Detailed Description

Base context for C source output.

### 6.122.2 Constructor & Destructor Documentation

#### 6.122.2.1 `print_csrc()`

```
GiNaC::print_csrc::print_csrc (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

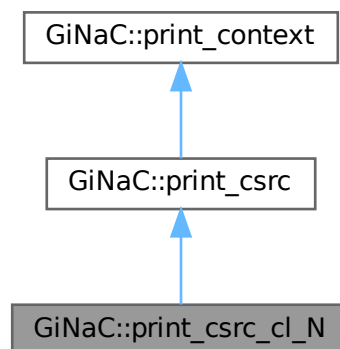
- [print.h](#)
- [print.cpp](#)

## 6.123 GiNaC::print\_csrc\_cl\_N Class Reference

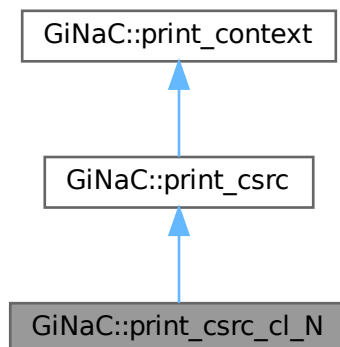
Context for C source output using CLN numbers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_cl\_N:



Collaboration diagram for GiNaC::print\_csrc\_cl\_N:



#### Public Member Functions

- [print\\_csrc\\_cl\\_N](#) (std::ostream &, unsigned [options](#)=0)

#### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

#### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

#### Additional Inherited Members

#### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.123.1 Detailed Description

Context for C source output using CLN numbers.

## 6.123.2 Constructor & Destructor Documentation

### 6.123.2.1 `print_csrc_cl_N()`

```
GiNaC::print_csrc_cl_N::print_csrc_cl_N (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

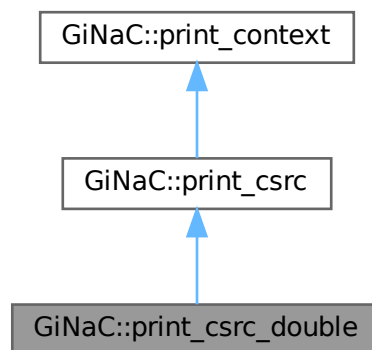
- [print.h](#)
- [print.cpp](#)

## 6.124 GiNaC::print\_csrc\_double Class Reference

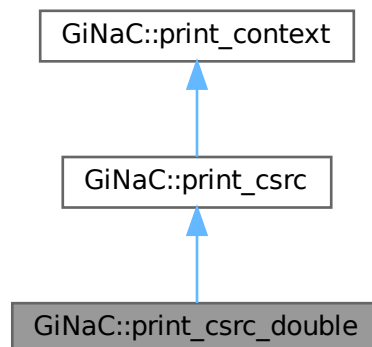
Context for C source output using double precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_double:



Collaboration diagram for GiNaC::print\_csrc\_double:



### Public Member Functions

- [print\\_csrc\\_double](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 6.124.1 Detailed Description

Context for C source output using double precision.

## 6.124.2 Constructor & Destructor Documentation

### 6.124.2.1 `print_csrc_double()`

```
GiNaC::print_csrc_double::print_csrc_double (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

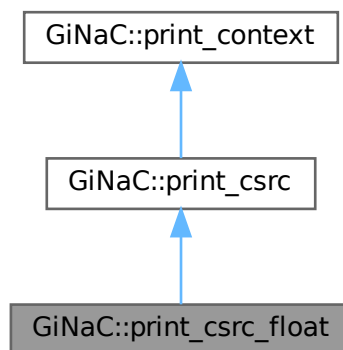
- [print.h](#)
- [print.cpp](#)

## 6.125 GiNaC::print\_csrc\_float Class Reference

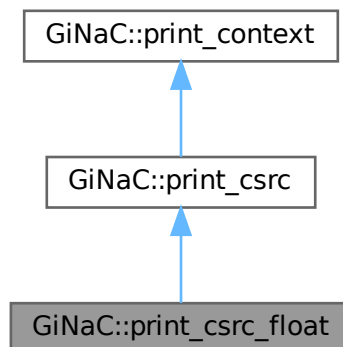
Context for C source output using float precision.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_csrc\_float:



Collaboration diagram for GiNaC::print\_csrc\_float:



### Public Member Functions

- [print\\_csrc\\_float](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_csrc](#)

- [print\\_csrc](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 6.125.1 Detailed Description

Context for C source output using float precision.

## 6.125.2 Constructor & Destructor Documentation

### 6.125.2.1 `print_csrc_float()`

```
GiNaC::print_csrc_float::print_csrc_float (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

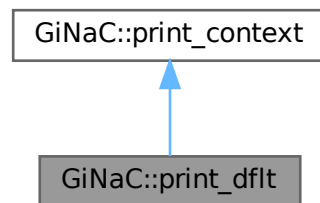
- [print.h](#)
- [print.cpp](#)

## 6.126 GiNaC::print\_dflt Class Reference

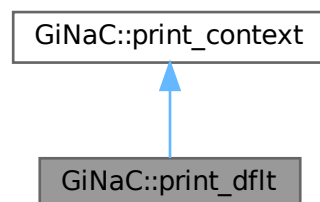
Context for default (ginsh-parsable) output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_dflt:



Collaboration diagram for GiNaC::print\_dflt:



## Public Member Functions

- [print\\_dflt](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.126.1 Detailed Description

Context for default (ginsh-parsable) output.

### 6.126.2 Constructor & Destructor Documentation

#### 6.126.2.1 [print\\_dflt\(\)](#)

```
GiNaC::print_dflt::print_dflt (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.127 GiNaC::print\_functor Class Reference

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

```
#include <print.h>
```

## Public Member Functions

- [print\\_functor](#) ()
- [print\\_functor](#) (const [print\\_functor](#) &other)
- [print\\_functor](#) (std::unique\_ptr< [print\\_functor\\_impl](#) > impl\_)
- template<class T , class C >  
  [print\\_functor](#) (void f(const T &, const C &, unsigned))
- template<class T , class C >  
  [print\\_functor](#) (void(T::\*f)(const C &, unsigned) const)
- [print\\_functor](#) & [operator=](#) (const [print\\_functor](#) &other)
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const
- bool [is\\_valid](#) () const

## Private Attributes

- std::unique\_ptr< [print\\_functor\\_impl](#) > impl

## 6.127.1 Detailed Description

This class represents a print method for a certain algebraic class and [print\\_context](#) type.

Its main purpose is to hide the difference between member functions and nonmember functions behind one unified [operator\(\)](#) interface. [print\\_functor](#) has value semantics and acts as a smart pointer (with deep copy) to a class derived from [print\\_functor\\_impl](#) which implements the actual function call.

## 6.127.2 Constructor & Destructor Documentation

### 6.127.2.1 [print\\_functor\(\)](#) [1/5]

```
GiNaC::print_functor::print_functor () [inline]
```

### 6.127.2.2 [print\\_functor\(\)](#) [2/5]

```
GiNaC::print_functor::print_functor (
 const print_functor & other) [inline]
```

### 6.127.2.3 [print\\_functor\(\)](#) [3/5]

```
GiNaC::print_functor::print_functor (
 std::unique_ptr< print_functor_impl > impl_) [inline]
```

### 6.127.2.4 [print\\_functor\(\)](#) [4/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
 void fconst T &, const C &, unsigned) [inline]
```

### 6.127.2.5 print\_functor() [5/5]

```
template<class T , class C >
GiNaC::print_functor::print_functor (
 void(T::*)(const C &, unsigned) const f) [inline]
```

## 6.127.3 Member Function Documentation

### 6.127.3.1 operator=()

```
print_functor & GiNaC::print_functor::operator= (
 const print_functor & other) [inline]
```

References [impl](#).

### 6.127.3.2 operator>()()

```
void GiNaC::print_functor::operator() (
 const basic & obj,
 const print_context & c,
 unsigned level) const [inline]
```

References [c](#).

### 6.127.3.3 is\_valid()

```
bool GiNaC::print_functor::is_valid () const [inline]
```

References [impl](#).

## 6.127.4 Member Data Documentation

### 6.127.4.1 impl

```
std::unique_ptr<print_functor_impl> GiNaC::print_functor::impl [private]
```

Referenced by [is\\_valid\(\)](#), and [operator=\(\)](#).

The documentation for this class was generated from the following file:

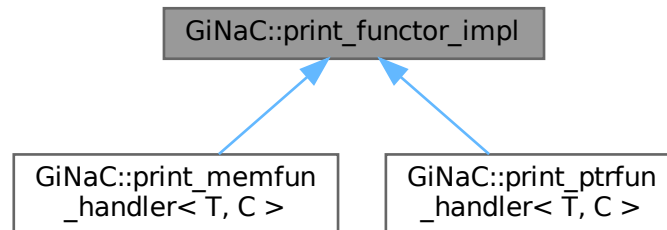
- [print.h](#)

## 6.128 GiNaC::print\_functor\_impl Class Reference

Base class for [print\\_functor](#) handlers.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_functor\_impl:



### Public Member Functions

- virtual [~print\\_functor\\_impl](#) ()
- virtual [print\\_functor\\_impl](#) \* [duplicate](#) () const =0
- virtual void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const =0

### 6.128.1 Detailed Description

Base class for [print\\_functor](#) handlers.

### 6.128.2 Constructor & Destructor Documentation

#### 6.128.2.1 ~print\_functor\_impl()

```
virtual GiNaC::print_functor_impl::~~print_functor_impl () [inline], [virtual]
```

### 6.128.3 Member Function Documentation

#### 6.128.3.1 duplicate()

```
virtual print_functor_impl * GiNaC::print_functor_impl::duplicate () const [pure virtual]
```

Implemented in [GiNaC::print\\_ptrfun\\_handler< T, C >](#), and [GiNaC::print\\_memfun\\_handler< T, C >](#).

### 6.128.3.2 operator>()

```
virtual void GiNaC::print_functor_impl::operator() (
 const basic & obj,
 const print_context & c,
 unsigned level) const [pure virtual]
```

Implemented in [GiNaC::print\\_ptrfun\\_handler< T, C >](#), and [GiNaC::print\\_memfun\\_handler< T, C >](#).

The documentation for this class was generated from the following file:

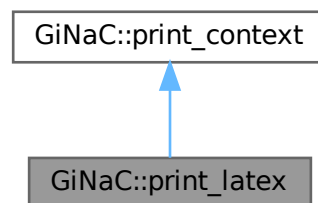
- [print.h](#)

## 6.129 GiNaC::print\_latex Class Reference

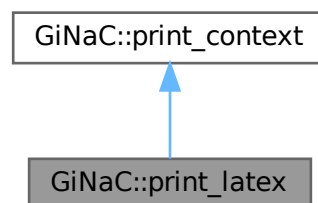
Context for latex-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_latex:



Collaboration diagram for GiNaC::print\_latex:



## Public Member Functions

- [print\\_latex](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.129.1 Detailed Description

Context for latex-parsable output.

### 6.129.2 Constructor & Destructor Documentation

#### 6.129.2.1 [print\\_latex\(\)](#)

```
GiNaC::print_latex::print_latex (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

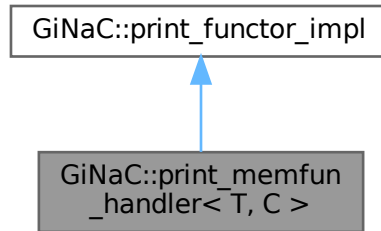
- [print.h](#)
- [print.cpp](#)

## 6.130 GiNaC::print\_memfun\_handler< T, C > Class Template Reference

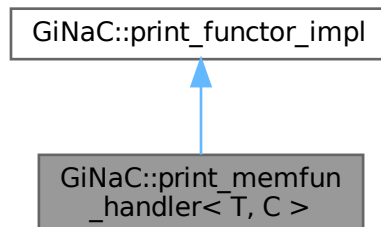
[print\\_functor](#) handler for member functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_memfun\_handler< T, C >:



Collaboration diagram for GiNaC::print\_memfun\_handler< T, C >:



### Public Types

- typedef void(T::\* [F](#)) (const C &[c](#), unsigned level) const

### Public Member Functions

- [print\\_memfun\\_handler](#) ([F](#) f\_)
- [print\\_memfun\\_handler](#) \* [duplicate](#) () const override
- void [operator](#)() (const [basic](#) &obj, const [print\\_context](#) &[c](#), unsigned level) const override

### Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()

## Private Attributes

- [F f](#)

### 6.130.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_memfun_handler< T, C >
```

[print\\_funcutor](#) handler for member functions of class T, context type C

### 6.130.2 Member Typedef Documentation

#### 6.130.2.1 F

```
template<class T , class C >
typedef void(T::* GiNaC::print_memfun_handler< T, C >::F) (const C &c, unsigned level) const
```

### 6.130.3 Constructor & Destructor Documentation

#### 6.130.3.1 print\_memfun\_handler()

```
template<class T , class C >
GiNaC::print_memfun_handler< T, C >::print_memfun_handler (
 F f_) [inline]
```

Referenced by [GiNaC::print\\_memfun\\_handler< T, C >::duplicate\(\)](#).

### 6.130.4 Member Function Documentation

#### 6.130.4.1 duplicate()

```
template<class T , class C >
print_memfun_handler * GiNaC::print_memfun_handler< T, C >::duplicate () const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_funcutor\\_impl](#).

References [GiNaC::print\\_memfun\\_handler< T, C >::print\\_memfun\\_handler\(\)](#).

#### 6.130.4.2 operator>()()

```
template<class T , class C >
void GiNaC::print_memfun_handler< T, C >::operator() (
 const basic & obj,
 const print_context & c,
 unsigned level) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_funcutor\\_impl](#).

References [c](#), and [GiNaC::print\\_memfun\\_handler< T, C >::f](#).

## 6.130.5 Member Data Documentation

### 6.130.5.1 f

```
template<class T , class C >
F GiNaC::print_memfun_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_memfun\\_handler< T, C >::operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [print.h](#)

## 6.131 GiNaC::print\_options Class Reference

Flags to control the behavior of a [print\\_context](#).

```
#include <print.h>
```

### Public Types

- enum { [print\\_index\\_dimensions](#) = 0x0001 }

### 6.131.1 Detailed Description

Flags to control the behavior of a [print\\_context](#).

## 6.131.2 Member Enumeration Documentation

### 6.131.2.1 anonymous enum

```
anonymous enum
```

#### Enumerator

|                                        |                                 |
|----------------------------------------|---------------------------------|
| <a href="#">print_index_dimensions</a> | print the dimensions of indices |
|----------------------------------------|---------------------------------|

The documentation for this class was generated from the following file:

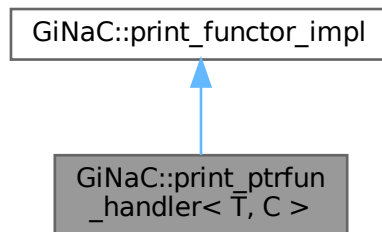
- [print.h](#)

## 6.132 GiNaC::print\_ptrfun\_handler< T, C > Class Template Reference

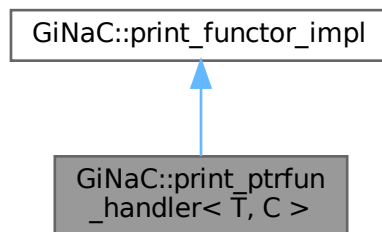
[print\\_functor](#) handler for pointer-to-functions of class T, context type C

```
#include <print.h>
```

Inheritance diagram for `GiNaC::print_ptrfun_handler< T, C >`:



Collaboration diagram for `GiNaC::print_ptrfun_handler< T, C >`:



## Public Types

- typedef void(\* [F](#)) (const T &, const C &, unsigned)

## Public Member Functions

- [print\\_ptrfun\\_handler](#) (F f\_)
- [print\\_ptrfun\\_handler](#) \* [duplicate](#) () const override
- void [operator\(\)](#) (const [basic](#) &obj, const [print\\_context](#) &c, unsigned level) const override

## Public Member Functions inherited from [GiNaC::print\\_functor\\_impl](#)

- virtual [~print\\_functor\\_impl](#) ()

## Private Attributes

- [F f](#)

### 6.132.1 Detailed Description

```
template<class T, class C>
class GiNaC::print_ptrfun_handler< T, C >
```

[print\\_functor](#) handler for pointer-to-functions of class T, context type C

### 6.132.2 Member Typedef Documentation

#### 6.132.2.1 F

```
template<class T , class C >
typedef void(* GiNaC::print_ptrfun_handler< T, C >::F) (const T &, const C &, unsigned)
```

### 6.132.3 Constructor & Destructor Documentation

#### 6.132.3.1 print\_ptrfun\_handler()

```
template<class T , class C >
GiNaC::print_ptrfun_handler< T, C >::print_ptrfun_handler (
 F f_) [inline]
```

Referenced by [GiNaC::print\\_ptrfun\\_handler< T, C >::duplicate\(\)](#).

### 6.132.4 Member Function Documentation

#### 6.132.4.1 duplicate()

```
template<class T , class C >
print_ptrfun_handler * GiNaC::print_ptrfun_handler< T, C >::duplicate () const [inline],
[override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [GiNaC::print\\_ptrfun\\_handler< T, C >::print\\_ptrfun\\_handler\(\)](#).

#### 6.132.4.2 operator>()()

```
template<class T , class C >
void GiNaC::print_ptrfun_handler< T, C >::operator() (
 const basic & obj,
 const print_context & c,
 unsigned level) const [inline], [override], [virtual]
```

Implements [GiNaC::print\\_functor\\_impl](#).

References [c](#), and [GiNaC::print\\_ptrfun\\_handler< T, C >::f](#).

## 6.132.5 Member Data Documentation

### 6.132.5.1 f

```
template<class T , class C >
F GiNaC::print_ptrfun_handler< T, C >::f [private]
```

Referenced by [GiNaC::print\\_ptrfun\\_handler< T, C >::operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

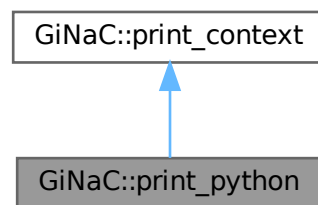
- [print.h](#)

## 6.133 GiNaC::print\_python Class Reference

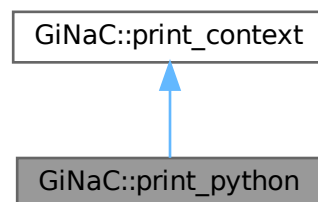
Context for python pretty-print output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python:



Collaboration diagram for GiNaC::print\_python:



## Public Member Functions

- [print\\_python](#) (std::ostream &, unsigned [options](#)=0)

## Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

## Additional Inherited Members

## Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.133.1 Detailed Description

Context for python pretty-print output.

### 6.133.2 Constructor & Destructor Documentation

#### 6.133.2.1 [print\\_python\(\)](#)

```
GiNaC::print_python::print_python (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

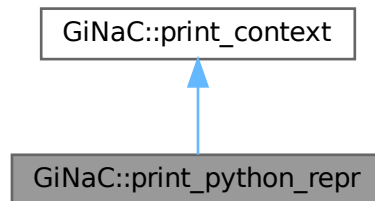
- [print.h](#)
- [print.cpp](#)

## 6.134 GiNaC::print\_python\_repr Class Reference

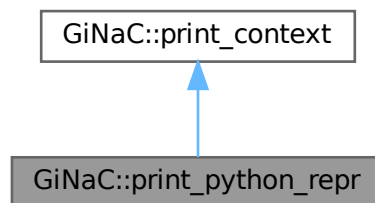
Context for python-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_python\_repr:



Collaboration diagram for GiNaC::print\_python\_repr:



### Public Member Functions

- [print\\_python\\_repr](#) (std::ostream &, unsigned [options](#)=0)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Additional Inherited Members

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

### 6.134.1 Detailed Description

Context for python-parsable output.

### 6.134.2 Constructor & Destructor Documentation

#### 6.134.2.1 print\_python\_repr()

```
GiNaC::print_python_repr::print_python_repr (
 std::ostream & os,
 unsigned options = 0)
```

The documentation for this class was generated from the following files:

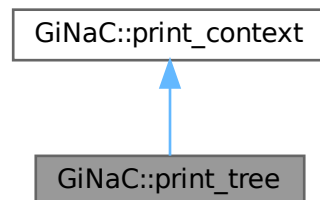
- [print.h](#)
- [print.cpp](#)

## 6.135 GiNaC::print\_tree Class Reference

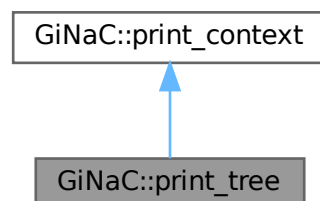
Context for tree-like output for debugging.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print\_tree:



Collaboration diagram for GiNaC::print\_tree:



### Public Member Functions

- [print\\_tree](#) (unsigned d)
- [print\\_tree](#) (std::ostream &, unsigned [options](#)=0, unsigned d=4)

### Public Member Functions inherited from [GiNaC::print\\_context](#)

- [print\\_context](#) (std::ostream &, unsigned [options](#)=0)
- virtual [~print\\_context](#) ()

### Public Attributes

- const unsigned [delta\\_indent](#)  
*size of indentation step*

### Public Attributes inherited from [GiNaC::print\\_context](#)

- std::ostream & [s](#)  
*stream to output to*
- unsigned [options](#)  
*option flags*

## 6.135.1 Detailed Description

Context for tree-like output for debugging.

## 6.135.2 Constructor & Destructor Documentation

### 6.135.2.1 [print\\_tree\(\)](#) [1/2]

```
GiNaC::print_tree::print_tree (
 unsigned d)
```

### 6.135.2.2 [print\\_tree\(\)](#) [2/2]

```
GiNaC::print_tree::print_tree (
 std::ostream & os,
 unsigned options = 0,
 unsigned d = 4)
```

### 6.135.3 Member Data Documentation

#### 6.135.3.1 delta\_indent

```
const unsigned GiNaC::print_tree::delta_indent
```

size of indentation step

The documentation for this class was generated from the following files:

- [print.h](#)
- [print.cpp](#)

## 6.136 GiNaC::archive\_node::property Struct Reference

Archived property (data type, name and associated data)

```
#include <archive.h>
```

### Public Member Functions

- [property](#) ()
- [property](#) ([archive\\_atom](#) n, [property\\_type](#) t, unsigned v)

### Public Attributes

- [property\\_type](#) type  
*Data type of property.*
- [archive\\_atom](#) name  
*Name of property.*
- unsigned [value](#)  
*Stored value.*

### 6.136.1 Detailed Description

Archived property (data type, name and associated data)

### 6.136.2 Constructor & Destructor Documentation

#### 6.136.2.1 [property](#)() [1/2]

```
GiNaC::archive_node::property::property () [inline]
```

### 6.136.2.2 `property()` [2/2]

```
GiNaC::archive_node::property::property (
 archive_atom n,
 property_type t,
 unsigned v) [inline]
```

## 6.136.3 Member Data Documentation

### 6.136.3.1 `type`

`property_type` GiNaC::archive\_node::property::type

Data type of property.

### 6.136.3.2 `name`

`archive_atom` GiNaC::archive\_node::property::name

Name of property.

### 6.136.3.3 `value`

`unsigned` GiNaC::archive\_node::property::value

Stored value.

The documentation for this struct was generated from the following file:

- [archive.h](#)

## 6.137 GiNaC::archive\_node::property\_info Struct Reference

Information about a stored property.

```
#include <archive.h>
```

### Public Member Functions

- [property\\_info](#) ()
- [property\\_info](#) ([property\\_type](#) t, const std::string &n, unsigned c=1)

## Public Attributes

- [property\\_type](#) `type`  
*Data type of property.*
- `std::string` [name](#)  
*Name of property.*
- `unsigned` [count](#)  
*Number of occurrences.*

## 6.137.1 Detailed Description

Information about a stored property.

A vector of these structures is returned by [get\\_properties\(\)](#).

See also

[get\\_properties](#)

## 6.137.2 Constructor & Destructor Documentation

### 6.137.2.1 `property_info()` [1/2]

```
GiNaC::archive_node::property_info::property_info () [inline]
```

### 6.137.2.2 `property_info()` [2/2]

```
GiNaC::archive_node::property_info::property_info (
 property_type t,
 const std::string & n,
 unsigned c = 1) [inline]
```

## 6.137.3 Member Data Documentation

### 6.137.3.1 `type`

[property\\_type](#) GiNaC::archive\_node::property\_info::type

Data type of property.

### 6.137.3.2 `name`

`std::string` GiNaC::archive\_node::property\_info::name

Name of property.

### 6.137.3.3 count

```
unsigned GiNaC::archive_node::property_info::count
```

Number of occurrences.

The documentation for this struct was generated from the following file:

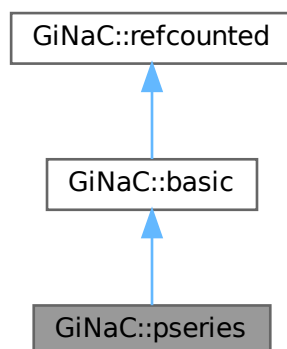
- [archive.h](#)

## 6.138 GiNaC::pseries Class Reference

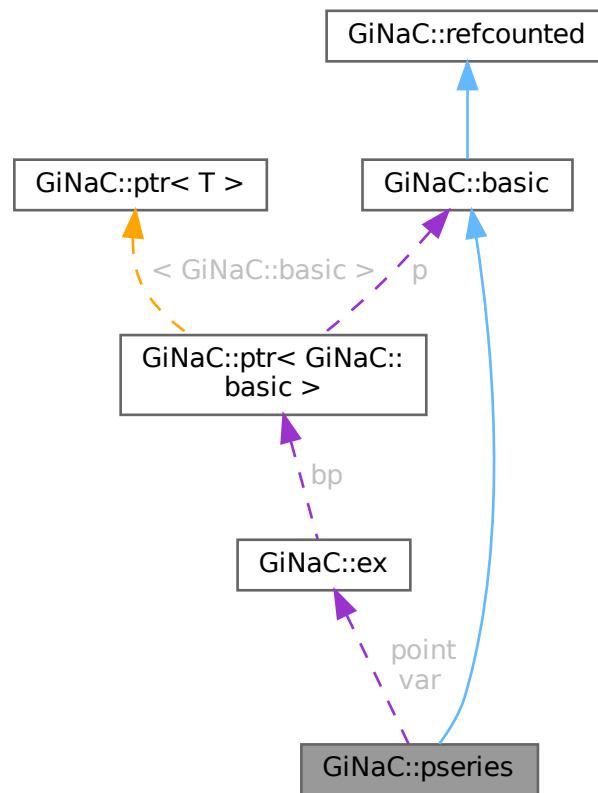
This class holds a extended truncated power series (positive and negative integer powers).

```
#include <pseries.h>
```

Inheritance diagram for GiNaC::pseries:



Collaboration diagram for GiNaC::pseries:



## Public Member Functions

- `pseries` (const `ex` &rel\_, const `epvector` &ops\_)  
Construct `pseries` from a vector of coefficients and powers.
- `pseries` (const `ex` &rel\_, `epvector` &&ops\_)
- unsigned `precedence` () const override  
Return relative operator precedence (for parenthezing output).
- `size_t nops` () const override  
Return the number of operands including a possible order term.
- `ex op` (size\_t i) const override  
Return the *i*th term in the series when represented as a sum.
- int `degree` (const `ex` &s) const override  
Return degree of highest power of the series.
- int `ldegree` (const `ex` &s) const override  
Return degree of lowest power of the series.
- `ex coeff` (const `ex` &s, int n=1) const override  
Return coefficient of degree *n* in power series if *s* is the expansion variable.
- `ex collect` (const `ex` &s, bool distributed=false) const override  
Does nothing.

- `ex eval ()` const override  
*Perform coefficient-wise automatic term rewriting rules in this class.*
- `ex evalf ()` const override  
*Evaluate coefficients numerically.*
- `ex series (const relational &r, int order, unsigned options=0)` const override  
*Re-expansion of a pseries object.*
- `ex subs (const exmap &m, unsigned options=0)` const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const override  
*Implementation of `ex::normal()` for pseries.*
- `ex expand (unsigned options=0)` const override  
*Implementation of `ex::expand()` for a power series.*
- `ex conjugate ()` const override
- `ex real_part ()` const override
- `ex imag_part ()` const override
- `ex eval_integ ()` const override  
*Evaluate integrals, if result is known.*
- `ex evalm ()` const override  
*Evaluate sums, products and integer powers of matrices.*
- `void archive (archive_node &n)` const override  
*Save (a.k.a.*
- `void read_archive (const archive_node &n, lst &syms)` override  
*Read (a.k.a.*
- `ex get_var ()` const  
*Get the expansion variable.*
- `ex get_point ()` const  
*Get the expansion point.*
- `ex convert_to_poly (bool no_order=false)` const  
*Convert the pseries object to an ordinary polynomial.*
- `bool is_compatible_to (const pseries &other)` const  
*Check whether series is compatible to another series (expansion variable and point are the same.*
- `bool is_zero ()` const  
*Check whether series has the value zero.*
- `bool is_terminating ()` const  
*Returns true if there is no order term, i.e.*
- `ex coeffop (size_t i)` const  
*Get coefficients and exponents.*
- `ex exponop (size_t i)` const
- `ex add_series (const pseries &other)` const  
*Add one series object to another, producing a pseries object that represents the sum.*
- `ex mul_const (const numeric &other)` const  
*Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.*
- `ex mul_series (const pseries &other)` const  
*Multiply one pseries object to another, producing a pseries object that represents the product.*
- `ex power_const (const numeric &p, int deg)` const  
*Compute the p-th power of a series.*
- `pseries shift_exponents (int deg)` const  
*Return a new pseries object with the powers shifted by deg.*

Public Member Functions inherited from **GiNaC::basic**

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex` & `let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex` & `operator[]` (const `ex` &index)
- virtual `ex` & `operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_info` () const

- `template<class T >`  
`void print_dispatch (const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `void print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level) const`  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options) const`  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1) const`  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare (const basic &other) const`  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal (const basic &other) const`  
*Test for syntactic equality.*
- `const basic & hold () const`  
*Stop further evaluation.*
- `unsigned gethash () const`
- `const basic & setflag (unsigned f) const`  
*Set some `status_flags`.*
- `const basic & clearflag (unsigned f) const`  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted () noexcept`
- `unsigned int add_reference () noexcept`
- `unsigned int remove_reference () noexcept`
- `unsigned int get_refcount () const noexcept`
- `void set_refcount (unsigned int r) noexcept`

## Protected Member Functions

- `ex derivative (const symbol &s) const override`  
*Implementation of `ex::diff()` for a power series.*
- `void print_series (const print_context &c, const char *openbrace, const char *closebrace, const char *mul←  
_sym, const char *pow_sym, unsigned level) const`
- `void do_print (const print_context &c, unsigned level) const`
- `void do_print_latex (const print_latex &c, unsigned level) const`
- `void do_print_tree (const print_tree &c, unsigned level) const`
- `void do_print_python (const print_python &c, unsigned level) const`
- `void do_print_python_repr (const print_python_repr &c, unsigned level) const`

## Protected Member Functions inherited from `GiNaC::basic`

- `basic ()`
- `virtual ex eval_ncmul (const exvector &v) const`
- `virtual bool match_same_type (const basic &other) const`  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual int compare_same_type (const basic &other) const`  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type (const basic &other) const`

- Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Protected Attributes

- [epvector seq](#)  
*Vector of {coefficient, power} pairs.*
- [ex var](#)  
*Series variable (holds a symbol)*
- [ex point](#)  
*Expansion point.*

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.138.1 Detailed Description

This class holds a extended truncated power series (positive and negative integer powers).

It consists of expression coefficients (only non-zero coefficients are stored), an expansion variable and an expansion point. Other classes must provide members to convert into this type.

## 6.138.2 Constructor & Destructor Documentation

### 6.138.2.1 [pseries\(\)](#) [1/2]

```
GiNaC::pseries::pseries (
 const ex & rel_,
 const epvector & ops_)
```

Construct pseries from a vector of coefficients and powers.

[expair.rest](#) holds the coefficient, [expair.coeff](#) holds the power. The powers must be integers (positive or negative) and in ascending order; the last coefficient can be `Order(_ex1)` to represent a truncated, non-terminating series.

## Parameters

|                            |                                                                     |
|----------------------------|---------------------------------------------------------------------|
| $rel \leftrightarrow$<br>— | expansion variable and point (must hold a relational)               |
| $ops \leftrightarrow$<br>— | vector of {coefficient, power} pairs (coefficient must not be zero) |

References [GINAC\\_ASSERT](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

Referenced by [add\\_series\(\)](#), [derivative\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [power\\_const\(\)](#), [series\(\)](#), and [shift\\_exponents\(\)](#).

### 6.138.2.2 pseries() [2/2]

```
GiNaC::pseries::pseries (
 const ex & rel_,
 epvector && ops_)
```

References [GINAC\\_ASSERT](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::lhs\(\)](#), [point](#), [GiNaC::ex::rhs\(\)](#), [seq](#), and [var](#).

## 6.138.3 Member Function Documentation

### 6.138.3.1 precedence()

```
unsigned GiNaC::pseries::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [print\\_series\(\)](#).

### 6.138.3.2 nops()

```
size_t GiNaC::pseries::nops () const [override], [virtual]
```

Return the number of operands including a possible order term.

Reimplemented from [GiNaC::basic](#).

References [seq](#).

Referenced by [coeffop\(\)](#), [exponop\(\)](#), and [GiNaC::log\\_series\(\)](#).

### 6.138.3.3 op()

```
ex GiNaC::pseries::op (
 size_t i) const [override], [virtual]
```

Return the ith term in the series when represented as a sum.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).

#### 6.138.3.4 degree()

```
int GiNaC::pseries::degree (
 const ex & s) const [override], [virtual]
```

Return degree of highest power of the series.

This is usually the exponent of the Order term. If s is not the expansion variable of the series, the series is examined termwise.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [mul\\_series\(\)](#), [power\\_const\(\)](#), and [series\(\)](#).

#### 6.138.3.5 ldegree()

```
int GiNaC::pseries::ldegree (
 const ex & s) const [override], [virtual]
```

Return degree of lowest power of the series.

This is usually the exponent of the leading term. If s is not the expansion variable of the series, the series is examined termwise. If s is the expansion variable but the expansion point is not zero the series is not expanded to find the degree. I.e.:  $(1-x) + (1-x)^2 + \text{Order}((1-x)^3)$  has `ldegree(x)` 1, not 0.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::is\\_equal\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), and [power\\_const\(\)](#).

#### 6.138.3.6 coeff()

```
ex GiNaC::pseries::coeff (
 const ex & s,
 int n = 1) const [override], [virtual]
```

Return coefficient of degree n in power series if s is the expansion variable.

If the expansion point is nonzero, by definition the n=1 coefficient in s of  $a+b*(s-z)+c*(s-z)^2+\text{Order}((s-z)^3)$  is b (assuming the expansion took place in the s in the first place). If s is not the expansion variable, an attempt is made to convert the series to a polynomial and return the corresponding coefficient from there.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::ex::coeff\(\)](#), [coeff\(\)](#), [convert\\_to\\_poly\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [n](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [GiNaC::log\\_series\(\)](#), [mul\\_series\(\)](#), [op\(\)](#), [power\\_const\(\)](#), and [read\\_archive\(\)](#).

**6.138.3.7 collect()**

```
ex GiNaC::pseries::collect (
 const ex & s,
 bool distributed = false) const [override], [virtual]
```

Does nothing.

Reimplemented from [GiNaC::basic](#).

**6.138.3.8 eval()**

```
ex GiNaC::pseries::eval () const [override], [virtual]
```

Perform coefficient-wise automatic term rewriting rules in this class.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.138.3.9 evalf()**

```
ex GiNaC::pseries::evalf () const [override], [virtual]
```

Evaluate coefficients numerically.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [point](#), [seq](#), and [var](#).

**6.138.3.10 series()**

```
ex GiNaC::pseries::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Re-expansion of a pseries object.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex1](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::is\\_equal\(\)](#), [options](#), [order](#), [point](#), [pseries\(\)](#), [r](#), [GiNaC::ex::rhs\(\)](#), [seq](#), [GiNaC::ex::series\(\)](#), and [var](#).

**6.138.3.11 subs()**

```
ex GiNaC::pseries::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [convert\\_to\\_poly\(\)](#), [m](#), [options](#), [point](#), [seq](#), [GiNaC::ex::subs\(\)](#), and [var](#).

**6.138.3.12 normal()**

```
ex GiNaC::pseries::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for pseries.

It normalizes each coefficient and replaces the series by a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [n](#), [GiNaC::ex::normal\(\)](#), [point](#), [pseries\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [seq](#), and [var](#).

**6.138.3.13 expand()**

```
ex GiNaC::pseries::expand (
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::expand\(\)](#) for a power series.

It expands all the terms individually and returns the resulting series as a new pseries.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::status\\_flags::expanded](#), [GiNaC::ex::is\\_zero\(\)](#), [options](#), [point](#), [seq](#), and [var](#).

**6.138.3.14 conjugate()**

```
ex GiNaC::pseries::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [seq](#), and [var](#).

**6.138.3.15 real\_part()**

```
ex GiNaC::pseries::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

**6.138.3.16 imag\_part()**

```
ex GiNaC::pseries::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::info\(\)](#), [point](#), [GiNaC::info\\_flags::real](#), [GiNaC::ex::real\\_part\(\)](#), [seq](#), and [var](#).

**6.138.3.17 eval\_integ()**

```
ex GiNaC::pseries::eval_integ () const [override], [virtual]
```

Evaluate integrals, if result is known.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::eval\\_integ\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.18 evalm()**

```
ex GiNaC::pseries::evalm () const [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::evalm\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [seq](#), and [var](#).

**6.138.3.19 archive()**

```
void GiNaC::pseries::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [point](#), [seq](#), and [var](#).

**6.138.3.20 read\_archive()**

```
void GiNaC::pseries::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [coeff\(\)](#), [n](#), [point](#), [seq](#), and [var](#).

**6.138.3.21 derivative()**

```
ex GiNaC::pseries::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for a power series.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::ex::diff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

**6.138.3.22 get\_var()**

```
ex GiNaC::pseries::get_var () const [inline]
```

Get the expansion variable.

References [var](#).

**6.138.3.23 get\_point()**

```
ex GiNaC::pseries::get_point () const [inline]
```

Get the expansion point.

References [point](#).

**6.138.3.24 convert\_to\_poly()**

```
ex GiNaC::pseries::convert_to_poly (
 bool no_order = false) const
```

Convert the pseries object to an ordinary polynomial.

Parameters

|                 |                                  |
|-----------------|----------------------------------|
| <i>no_order</i> | flag: discard higher order terms |
|-----------------|----------------------------------|

References [GiNaC::ex::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [GiNaC::pow\(\)](#), [seq](#), and [var](#).

Referenced by [coeff\(\)](#), [series\(\)](#), and [subs\(\)](#).

**6.138.3.25 is\_compatible\_to()**

```
bool GiNaC::pseries::is_compatible_to (
 const pseries & other) const [inline]
```

Check whether series is compatible to another series (expansion variable and point are the same).

References [GiNaC::ex::is\\_equal\(\)](#), [point](#), and [var](#).

Referenced by [add\\_series\(\)](#), and [mul\\_series\(\)](#).

#### 6.138.3.26 `is_zero()`

```
bool GiNaC::pseries::is_zero () const [inline]
```

Check whether series has the value zero.

References [seq](#).

Referenced by [power\\_const\(\)](#).

#### 6.138.3.27 `is_terminating()`

```
bool GiNaC::pseries::is_terminating () const
```

Returns true if there is no order term, i.e.

the series terminates and false otherwise.

References [GiNaC::is\\_order\\_function\(\)](#), and [seq](#).

Referenced by [GiNaC::is\\_terminating\(\)](#), and [GiNaC::log\\_series\(\)](#).

#### 6.138.3.28 `coeffop()`

```
ex GiNaC::pseries::coeffop (
 size_t i) const
```

Get coefficients and exponents.

References [nops\(\)](#), and [seq](#).

#### 6.138.3.29 `exponop()`

```
ex GiNaC::pseries::exponop (
 size_t i) const
```

References [nops\(\)](#), and [seq](#).

#### 6.138.3.30 `add_series()`

```
ex GiNaC::pseries::add_series (
 const pseries & other) const
```

Add one series object to another, producing a pseries object that represents the sum.

## Parameters

|              |                            |
|--------------|----------------------------|
| <i>other</i> | pseries object to add with |
|--------------|----------------------------|

## Returns

the sum as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [is\\_compatible\\_to\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::log\\_series\(\)](#).

**6.138.3.31 mul\_const()**

```
ex GiNaC::pseries::mul_const (
 const numeric & other) const
```

Multiply a pseries object with a numeric constant, producing a pseries object that represents the product.

## Parameters

|              |                           |
|--------------|---------------------------|
| <i>other</i> | constant to multiply with |
|--------------|---------------------------|

## Returns

the product as a pseries

References [GiNaC::numeric::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

**6.138.3.32 mul\_series()**

```
ex GiNaC::pseries::mul_series (
 const pseries & other) const
```

Multiply one pseries object to another, producing a pseries object that represents the product.

## Parameters

|              |                                 |
|--------------|---------------------------------|
| <i>other</i> | pseries object to multiply with |
|--------------|---------------------------------|

## Returns

the product as a pseries

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [coeff\(\)](#), [degree\(\)](#), [GiNaC::ex::find\(\)](#), [is\\_compatible\\_to\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [lddegree\(\)](#), [point](#), [pseries\(\)](#), [seq](#), and [var](#).

Referenced by [GiNaC::mul::series\(\)](#).

### 6.138.3.33 power\_const()

```
ex GiNaC::pseries::power_const (
 const numeric & p,
 int deg) const
```

Compute the p-th power of a series.

#### Parameters

|            |                                        |
|------------|----------------------------------------|
| <i>p</i>   | power to compute                       |
| <i>deg</i> | truncation order of series calculation |

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [c](#), [coeff\(\)](#), [degree\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::numeric::is\\_negative\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::numeric::is\\_pos\\_integer\(\)](#), [GiNaC::numeric::is\\_zero\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [point](#), [GiNaC::pow\(\)](#), [pseries\(\)](#), [GiNaC::numeric::real\(\)](#), [seq](#), [GiNaC::to\\_int\(\)](#), and [var](#).

### 6.138.3.34 shift\_exponents()

```
pseries GiNaC::pseries::shift_exponents (
 int deg) const
```

Return a new pseries object with the powers shifted by deg.

References [point](#), [pseries\(\)](#), [seq](#), and [var](#).

### 6.138.3.35 print\_series()

```
void GiNaC::pseries::print_series (
 const print_context & c,
 const char * openbrace,
 const char * closebrace,
 const char * mul_sym,
 const char * pow_sym,
 unsigned level) const [protected]
```

References [GiNaC::\\_ex1](#), [c](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::numeric](#), [point](#), [GiNaC::info\\_flags::positive](#), [GiNaC::pow\(\)](#), [precedence\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

Referenced by [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), and [do\\_print\\_python\(\)](#).

### 6.138.3.36 do\_print()

```
void GiNaC::pseries::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

**6.138.3.37 do\_print\_latex()**

```
void GiNaC::pseries::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

**6.138.3.38 do\_print\_tree()**

```
void GiNaC::pseries::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

**6.138.3.39 do\_print\_python()**

```
void GiNaC::pseries::do_print_python (
 const print_python & c,
 unsigned level) const [protected]
```

References [c](#), and [print\\_series\(\)](#).

**6.138.3.40 do\_print\_python\_repr()**

```
void GiNaC::pseries::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [point](#), [GiNaC::ex::print\(\)](#), [seq](#), and [var](#).

**6.138.4 Member Data Documentation****6.138.4.1 seq**

```
epvector GiNaC::pseries::seq [protected]
```

Vector of {coefficient, power} pairs.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [coeffop\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [exponop\(\)](#), [imag\\_part\(\)](#), [is\\_terminating\(\)](#), [is\\_zero\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [nops\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

### 6.138.4.2 var

```
ex GiNaC::pseries::var [protected]
```

Series variable (holds a symbol)

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [coeff\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [degree\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_var\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [ldegree\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

### 6.138.4.3 point

```
ex GiNaC::pseries::point [protected]
```

Expansion point.

Referenced by [add\\_series\(\)](#), [archive\(\)](#), [conjugate\(\)](#), [convert\\_to\\_poly\(\)](#), [derivative\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [eval\\_integ\(\)](#), [evalf\(\)](#), [evalm\(\)](#), [expand\(\)](#), [get\\_point\(\)](#), [imag\\_part\(\)](#), [is\\_compatible\\_to\(\)](#), [mul\\_const\(\)](#), [mul\\_series\(\)](#), [normal\(\)](#), [op\(\)](#), [power\\_const\(\)](#), [print\\_series\(\)](#), [pseries\(\)](#), [pseries\(\)](#), [read\\_archive\(\)](#), [real\\_part\(\)](#), [series\(\)](#), [shift\\_exponents\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [pseries.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)

## 6.139 GiNaC::psi1\_SERIAL Class Reference

Polylogarithm and multiple polylogarithm.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.139.1 Detailed Description

Polylogarithm and multiple polylogarithm.

Nielsen's generalized polylogarithm. Harmonic polylogarithm. Gamma-function. Beta-function. Psi-function (aka digamma-function).

## 6.139.2 Member Data Documentation

### 6.139.2.1 serial

unsigned GiNaC::psi1\_SERIAL::serial [static]

**Initial value:**

```
=
 function::register_new(function_options("psi", 1).
 eval_func(psi1_eval).
 evalf_func(psi1_evalf).
 derivative_func(psi1_deriv).
 series_func(psi1_series).
 latex_name("\\psi").
 overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 6.140 GiNaC::psi2\_SERIAL Class Reference

Derivatives of Psi-function (aka polygamma-functions).

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.140.1 Detailed Description

Derivatives of Psi-function (aka polygamma-functions).

## 6.140.2 Member Data Documentation

### 6.140.2.1 serial

unsigned GiNaC::psi2\_SERIAL::serial [static]

**Initial value:**

```
=
 function::register_new(function_options("psi", 2).
 eval_func(psi2_eval).
 evalf_func(psi2_evalf).
 derivative_func(psi2_deriv).
 series_func(psi2_series).
 latex_name("\\psi").
 overloaded(2))
```

Referenced by [GiNaC::psi\(\)](#).

The documentation for this class was generated from the following files:

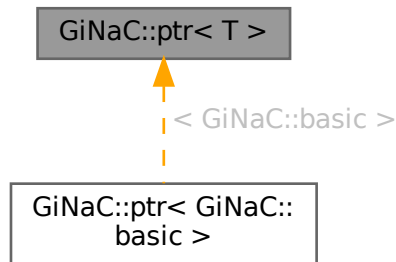
- [inifcns.h](#)
- [inifcns\\_gamma.cpp](#)

## 6.141 GiNaC::ptr< T > Class Template Reference

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

```
#include <ptr.h>
```

Inheritance diagram for GiNaC::ptr< T >:



### Public Member Functions

- `ptr` (T \*t) noexcept  
*Bind ptr to newly created object, start reference counting.*
- `ptr` (T &t) noexcept  
*Bind ptr to existing reference-counted object.*
- `ptr` (const `ptr` &other) noexcept
- `~ptr` ()
- `ptr` & `operator=` (const `ptr` &other)
- T & `operator*` () const noexcept
- T \* `operator->` () const noexcept
- void `makewritable` ()  
*Announce your intention to modify the object bound to this ptr.*
- void `swap` (`ptr` &other) noexcept  
*Swap the bound object of this ptr with another ptr.*
- template<class U >  
bool `operator==` (const `ptr`< U > &rhs) const noexcept
- template<class U >  
bool `operator!=` (const `ptr`< U > &rhs) const noexcept

### Private Attributes

- T \* `p`

## Friends

- struct [std::less< ptr< T > >](#)
- T \* [get\\_pointer](#) (const [ptr](#) &x) noexcept
- template<class U >  
bool [operator==](#) (const [ptr](#) &lhs, const U \*rhs) noexcept
- template<class U >  
bool [operator!=](#) (const [ptr](#) &lhs, const U \*rhs) noexcept
- template<class U >  
bool [operator==](#) (const U \*lhs, const [ptr](#) &rhs) noexcept
- template<class U >  
bool [operator!=](#) (const U \*lhs, const [ptr](#) &rhs) noexcept
- std::ostream & [operator<<](#) (std::ostream &os, const [ptr](#)< T > &rhs)

## 6.141.1 Detailed Description

**template<class T>**  
**class GiNaC::ptr< T >**

Class of (intrusively) reference-counted pointers that support copy-on-write semantics.

Requirements for T: must support the refcounted interface (usually by being derived from refcounted) T\* T↔::duplicate() member function (only if makewritable() is used)

## 6.141.2 Constructor & Destructor Documentation

### 6.141.2.1 ptr() [1/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
 T * t) [inline], [noexcept]
```

Bind ptr to newly created object, start reference counting.

References [GINAC\\_ASSERT](#), and [GiNaC::ptr< T >::p](#).

### 6.141.2.2 ptr() [2/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
 T & t) [inline], [explicit], [noexcept]
```

Bind ptr to existing reference-counted object.

References [GiNaC::ptr< T >::p](#).

### 6.141.2.3 ptr() [3/3]

```
template<class T >
GiNaC::ptr< T >::ptr (
 const ptr< T > & other) [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

**6.141.2.4 ~ptr()**

```
template<class T >
GiNaC::ptr< T >::~~ptr () [inline]
```

References [GiNaC::ptr< T >::p](#).

**6.141.3 Member Function Documentation****6.141.3.1 operator=()**

```
template<class T >
ptr & GiNaC::ptr< T >::operator= (
 const ptr< T > & other) [inline]
```

References [GiNaC::ptr< T >::p](#).

**6.141.3.2 operator\*()**

```
template<class T >
T & GiNaC::ptr< T >::operator* () const [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

**6.141.3.3 operator->()**

```
template<class T >
T * GiNaC::ptr< T >::operator-> () const [inline], [noexcept]
```

References [GiNaC::ptr< T >::p](#).

**6.141.3.4 makewritable()**

```
template<class T >
void GiNaC::ptr< T >::makewritable () [inline]
```

Announce your intention to modify the object bound to this ptr.

This ensures that the object is not shared by any other ptrs.

References [GiNaC::ptr< T >::p](#).

**6.141.3.5 swap()**

```
template<class T >
void GiNaC::ptr< T >::swap (
 ptr< T > & other) [inline], [noexcept]
```

Swap the bound object of this ptr with another ptr.

References [GiNaC::ptr< T >::p](#).

### 6.141.3.6 operator==( )

```
template<class T >
template<class U >
bool GiNaC::ptr< T >::operator==(
 const ptr< U > & rhs) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::get\\_pointer](#), [GiNaC::ptr< T >::p](#), and [GiNaC::rhs\(\)](#).

### 6.141.3.7 operator"!="()

```
template<class T >
template<class U >
bool GiNaC::ptr< T >::operator!=(
 const ptr< U > & rhs) const [inline], [noexcept]
```

References [GiNaC::ptr< T >::get\\_pointer](#), [GiNaC::ptr< T >::p](#), and [GiNaC::rhs\(\)](#).

## 6.141.4 Friends And Related Symbol Documentation

### 6.141.4.1 std::less< ptr< T > >

```
template<class T >
friend struct std::less< ptr< T > > [friend]
```

### 6.141.4.2 get\_pointer

```
template<class T >
T * get_pointer (
 const ptr< T > & x) [friend]
```

Referenced by [GiNaC::ptr< T >::operator!=\(\)](#), and [GiNaC::ptr< T >::operator==\( \)](#).

### 6.141.4.3 operator== [1/2]

```
template<class T >
template<class U >
bool operator==(
 const ptr< T > & lhs,
 const U * rhs) [friend]
```

### 6.141.4.4 operator"!=" [1/2]

```
template<class T >
template<class U >
bool operator!=(
 const ptr< T > & lhs,
 const U * rhs) [friend]
```

**6.141.4.5 operator== [2/2]**

```
template<class T >
template<class U >
bool operator== (
 const U * lhs,
 const ptr< T > & rhs) [friend]
```

**6.141.4.6 operator!= [2/2]**

```
template<class T >
template<class U >
bool operator!= (
 const U * lhs,
 const ptr< T > & rhs) [friend]
```

**6.141.4.7 operator<<**

```
template<class T >
std::ostream & operator<< (
 std::ostream & os,
 const ptr< T > & rhs) [friend]
```

**6.141.5 Member Data Documentation****6.141.5.1 p**

```
template<class T >
T* GiNaC::ptr< T >::p [private]
```

Referenced by [GiNaC::ptr< T >::makewritable\(\)](#), [GiNaC::ptr< T >::operator!=\(\)](#), [GiNaC::ptr< T >::operator\\*\(\)](#), [GiNaC::ptr< T >::operator->\(\)](#), [GiNaC::ptr< T >::operator=\(\)](#), [GiNaC::ptr< T >::operator==\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::ptr< T >::swap\(\)](#), and [GiNaC::ptr< T >::~~ptr\(\)](#).

The documentation for this class was generated from the following file:

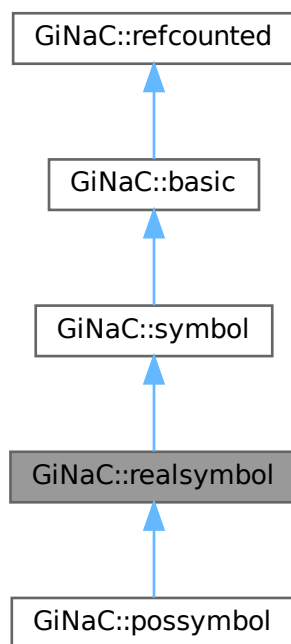
- [ptr.h](#)

## 6.142 GiNaC::realsymbol Class Reference

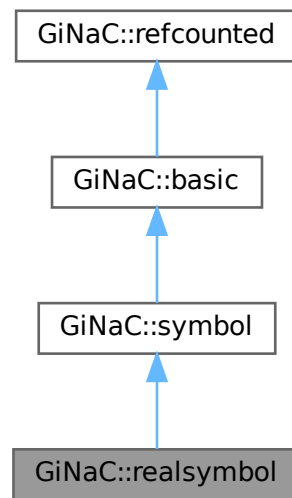
Specialization of symbol to real domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::realsymbol:



Collaboration diagram for `GiNaC::realsymbol`:



### Public Member Functions

- [realsymbol](#) ()
- [realsymbol](#) (const std::string &initname)
- [realsymbol](#) (const std::string &initname, const std::string &texname)
- unsigned [get\\_domain](#) () const override
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- [realsymbol](#) \* [duplicate](#) () const override

*Create a clone of this object on the heap.*

### Public Member Functions inherited from [GiNaC::symbol](#)

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override

- *Implementation of `ex::normal()` for symbols.*
- `ex to_rational` (`exmap` &`repl`) const override
- *Implementation of `ex::to_rational()` for symbols.*
- `ex to_polynomial` (`exmap` &`repl`) const override
- *Implementation of `ex::to_polynomial()` for symbols.*
- `ex conjugate` () const override
- `ex real_part` () const override
- `ex imag_part` () const override
- `bool is_polynomial` (const `ex` &`var`) const override
- *Check whether this is a polynomial in the given variables.*
- `void archive` (`archive_node` &`n`) const override
- *Save (a.k.a.*
- `void read_archive` (const `archive_node` &`n`, `lst` &`syms`) override
- *Read (a.k.a.*
- `void set_name` (const `std::string` &`n`)
- `void set_TeX_name` (const `std::string` &`n`)
- `std::string get_name` () const
- `std::string get_TeX_name` () const

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()
- *basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- `const basic & operator=` (const `basic` &`other`)
- *basic assignment operator: the other object might be of a derived class.*
- `virtual ex evalm` () const
- *Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () const
- *Evaluate integrals, if result is known.*
- `virtual ex eval_indexed` (const `basic` &`i`) const
- *Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print` (const `print_context` &`c`, unsigned `level=0`) const
- *Output to stream.*
- `virtual void dbgprint` () const
- *Little wrapper around `print` to be called within a debugger.*
- `virtual void dbgprinttree` () const
- *Little wrapper around `printtree` to be called within a debugger.*
- `virtual unsigned precedence` () const
- *Return relative operator precedence (for parenthezing output).*
- `virtual size_t nops` () const
- *Number of operands/members.*
- `virtual ex op` (size\_t `i`) const
- *Return operand/member at position `i`.*
- `virtual ex operator[]` (const `ex` &`index`) const
- `virtual ex operator[]` (size\_t `i`) const
- `virtual ex & let_op` (size\_t `i`)
- *Return modifiable operand/member at position `i`.*
- `virtual ex & operator[]` (const `ex` &`index`)
- `virtual ex & operator[]` (size\_t `i`)
- `virtual bool has` (const `ex` &`other`, unsigned `options=0`) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T >

void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const

*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*
- const `basic` & `hold` () const

*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const

*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const

*Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Additional Inherited Members

## Protected Member Functions inherited from GiNaC::symbol

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from GiNaC::basic

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes inherited from GiNaC::symbol

- unsigned `serial`  
*unique serial number for comparison*
- std::string `name`  
*printname of this symbol*
- std::string `TeX_name`  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

### 6.142.1 Detailed Description

Specialization of symbol to real domain.

### 6.142.2 Constructor & Destructor Documentation

#### 6.142.2.1 [realsymbol\(\)](#) [1/3]

```
GiNaC::realsymbol::realsymbol ()
```

Referenced by [duplicate\(\)](#).

#### 6.142.2.2 [realsymbol\(\)](#) [2/3]

```
GiNaC::realsymbol::realsymbol (
 const std::string & initname) [explicit]
```

#### 6.142.2.3 [realsymbol\(\)](#) [3/3]

```
GiNaC::realsymbol::realsymbol (
 const std::string & initname,
 const std::string & texname)
```

### 6.142.3 Member Function Documentation

#### 6.142.3.1 [get\\_domain\(\)](#)

```
unsigned GiNaC::realsymbol::get_domain () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::symbol](#).

References [GiNaC::domain::real](#).

#### 6.142.3.2 [conjugate\(\)](#)

```
ex GiNaC::realsymbol::conjugate () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.142.3.3 real\_part()

```
ex GiNaC::realsymbol::real_part () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.142.3.4 imag\_part()

```
ex GiNaC::realsymbol::imag_part () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.142.3.5 duplicate()

```
realsymbol * GiNaC::realsymbol::duplicate () const [inline], [override], [virtual]
```

Create a clone of this object on the heap.

One can think of this as simulating a virtual copy constructor which is needed for instance by the refcounted construction of an ex from a basic.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::dynallocated](#), [realsymbol\(\)](#), and [GiNaC::basic::setflag\(\)](#).

The documentation for this class was generated from the following files:

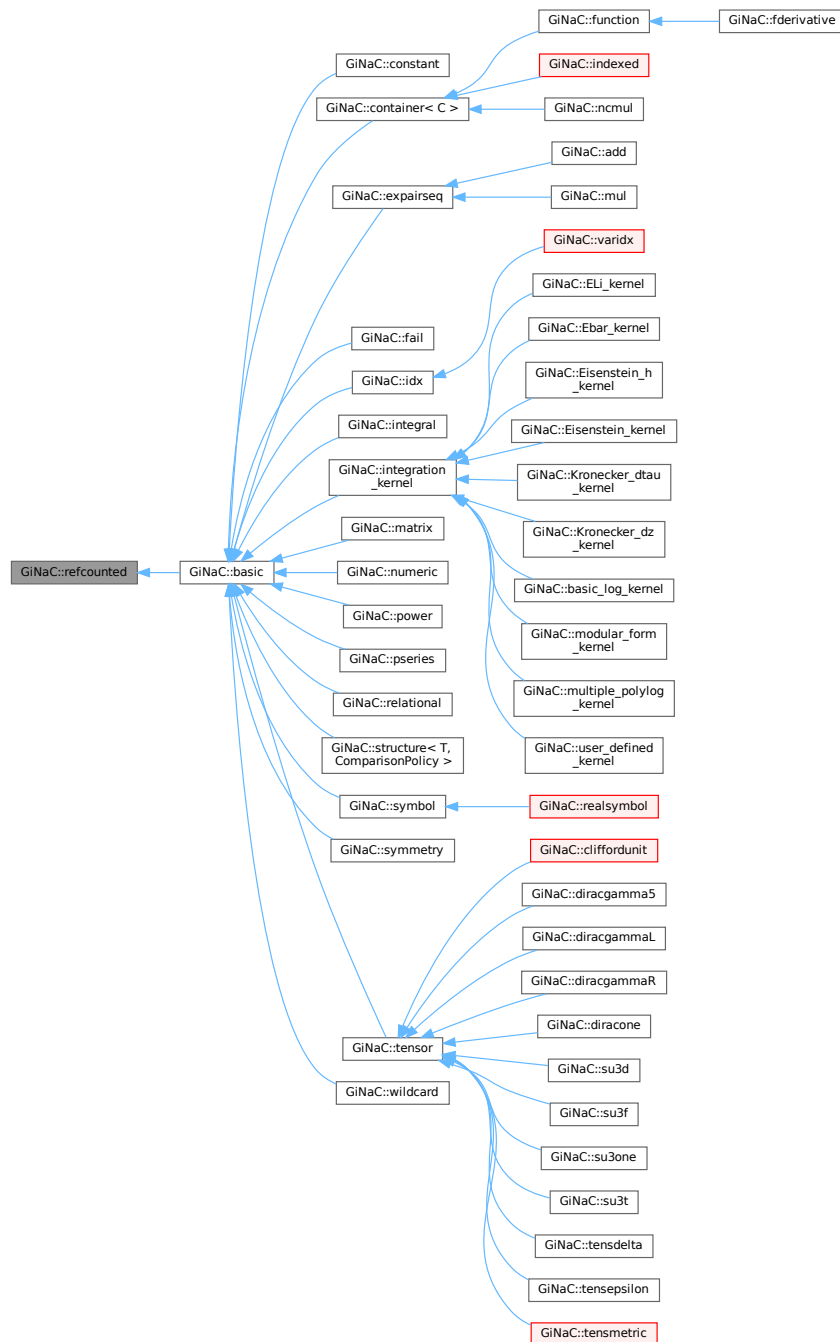
- [symbol.h](#)
- [symbol.cpp](#)

## 6.143 GiNaC::refcounted Class Reference

Base class for reference-counted objects.

```
#include <ptr.h>
```

Inheritance diagram for `GiNaC::refcounted`:



## Public Member Functions

- `refcounted()` noexcept
- unsigned int `add_reference()` noexcept
- unsigned int `remove_reference()` noexcept
- unsigned int `get_refcount()` const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

## Private Attributes

- unsigned int [refcount](#)  
*reference counter*

## 6.143.1 Detailed Description

Base class for reference-counted objects.

## 6.143.2 Constructor & Destructor Documentation

### 6.143.2.1 refcounted()

```
GiNaC::refcounted::refcounted () [inline], [noexcept]
```

## 6.143.3 Member Function Documentation

### 6.143.3.1 add\_reference()

```
unsigned int GiNaC::refcounted::add_reference () [inline], [noexcept]
```

References [refcount](#).

### 6.143.3.2 remove\_reference()

```
unsigned int GiNaC::refcounted::remove_reference () [inline], [noexcept]
```

References [refcount](#).

### 6.143.3.3 get\_refcount()

```
unsigned int GiNaC::refcounted::get_refcount () const [inline], [noexcept]
```

References [refcount](#).

Referenced by [GiNaC::ex::construct\\_from\\_basic\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

### 6.143.3.4 set\_refcount()

```
void GiNaC::refcounted::set_refcount (
 unsigned int r) [inline], [noexcept]
```

References [r](#), and [refcount](#).

## 6.143.4 Member Data Documentation

### 6.143.4.1 refcount

```
unsigned int GiNaC::refcounted::refcount [private]
```

reference counter

Referenced by [add\\_reference\(\)](#), [get\\_refcount\(\)](#), [remove\\_reference\(\)](#), and [set\\_refcount\(\)](#).

The documentation for this class was generated from the following file:

- [ptr.h](#)

## 6.144 GiNaC::registered\_class\_options Class Reference

This class stores information about a registered [GiNaC](#) class.

```
#include <registrar.h>
```

### Public Member Functions

- [registered\\_class\\_options](#) ([const char \\*n](#), [const char \\*p](#), [const std::type\\_info &ti](#))
- [const char \\* get\\_name](#) () [const](#)
- [const char \\* get\\_parent\\_name](#) () [const](#)
- [std::type\\_info const \\* get\\_id](#) () [const](#)
- [const std::vector< print\\_funcor > & get\\_print\\_dispatch\\_table](#) () [const](#)
- [template<class Ctx , class T , class C >](#)  
[registered\\_class\\_options & print\\_func](#) ([void f\(const T &, const C &c, unsigned\)](#))
- [template<class Ctx , class T , class C >](#)  
[registered\\_class\\_options & print\\_func](#) ([void\(T::\\*f\)\(const C &, unsigned\)](#))
- [template<class Ctx >](#)  
[registered\\_class\\_options & print\\_func](#) ([const print\\_funcor &f](#))
- [void set\\_print\\_func](#) ([unsigned id, const print\\_funcor &f](#))

### Private Attributes

- [const char \\* name](#)  
*Class name.*
- [const char \\* parent\\_name](#)  
*Name of superclass.*
- [std::type\\_info const \\* tinfo\\_key](#)  
*Type information key.*
- [std::vector< print\\_funcor > print\\_dispatch\\_table](#)  
*Method table for print() dispatch.*

### 6.144.1 Detailed Description

This class stores information about a registered [GiNaC](#) class.

## 6.144.2 Constructor & Destructor Documentation

### 6.144.2.1 registered\_class\_options()

```
GiNaC::registered_class_options::registered_class_options (
 const char * n,
 const char * p,
 const std::type_info & ti) [inline]
```

## 6.144.3 Member Function Documentation

### 6.144.3.1 get\_name()

```
const char * GiNaC::registered_class_options::get_name () const [inline]
```

References [name](#).

### 6.144.3.2 get\_parent\_name()

```
const char * GiNaC::registered_class_options::get_parent_name () const [inline]
```

References [parent\\_name](#).

### 6.144.3.3 get\_id()

```
std::type_info const * GiNaC::registered_class_options::get_id () const [inline]
```

References [tinfo\\_key](#).

### 6.144.3.4 get\_print\_dispatch\_table()

```
const std::vector< print_func_t > & GiNaC::registered_class_options::get_print_dispatch_table
() const [inline]
```

References [print\\_dispatch\\_table](#).

### 6.144.3.5 print\_func() [1/3]

```
template<class Ctx , class T , class C >
registered_class_options & GiNaC::registered_class_options::print_func (
 void fconst T &, const C &c, unsigned) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

**6.144.3.6 print\_func() [2/3]**

```
template<class Ctx , class T , class C >
registered_class_options & GiNaC::registered_class_options::print_func (
 void(T::*)(const C &, unsigned) f) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

**6.144.3.7 print\_func() [3/3]**

```
template<class Ctx >
registered_class_options & GiNaC::registered_class_options::print_func (
 const print_functor & f) [inline]
```

References [options](#), and [set\\_print\\_func\(\)](#).

**6.144.3.8 set\_print\_func()**

```
void GiNaC::registered_class_options::set_print_func (
 unsigned id,
 const print_functor & f) [inline]
```

References [print\\_dispatch\\_table](#).

Referenced by [print\\_func\(\)](#), [print\\_func\(\)](#), and [print\\_func\(\)](#).

**6.144.4 Member Data Documentation****6.144.4.1 name**

```
const char* GiNaC::registered_class_options::name [private]
```

Class name.

Referenced by [get\\_name\(\)](#).

**6.144.4.2 parent\_name**

```
const char* GiNaC::registered_class_options::parent_name [private]
```

Name of superclass.

Referenced by [get\\_parent\\_name\(\)](#).

**6.144.4.3 tinfo\_key**

```
std::type_info const* GiNaC::registered_class_options::tinfo_key [private]
```

Type information key.

Referenced by [get\\_id\(\)](#).

#### 6.144.4.4 print\_dispatch\_table

```
std::vector<print_functor> GiNaC::registered_class_options::print_dispatch_table [private]
```

Method table for print() dispatch.

Referenced by [get\\_print\\_dispatch\\_table\(\)](#), and [set\\_print\\_func\(\)](#).

The documentation for this class was generated from the following file:

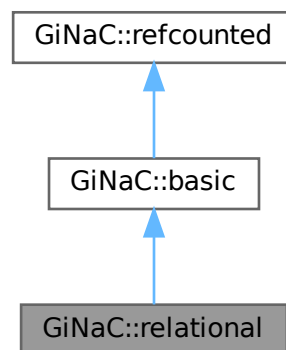
- [registrar.h](#)

## 6.145 GiNaC::relational Class Reference

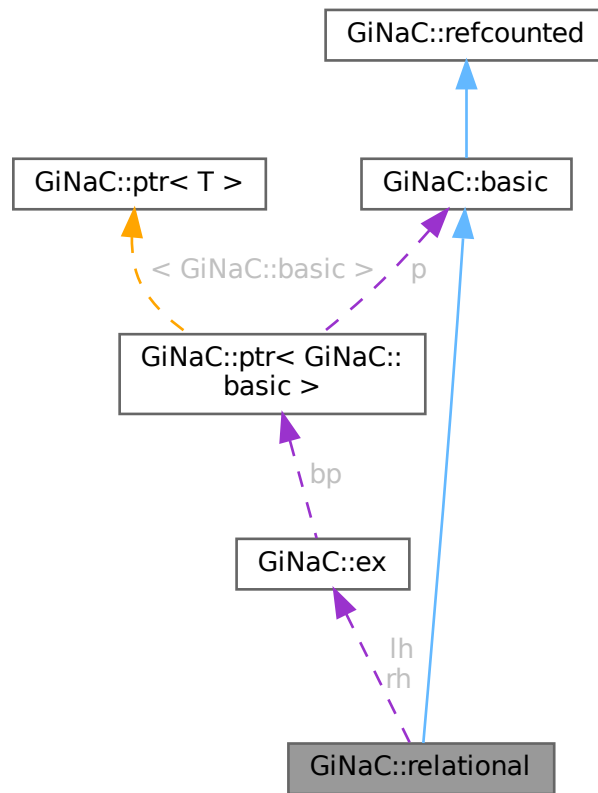
This class holds a relation consisting of two expressions and a logical relation between them.

```
#include <relational.h>
```

Inheritance diagram for GiNaC::relational:



Collaboration diagram for `GiNaC::relational`:



## Classes

- struct `safe_bool_helper`

## Public Types

- enum `operators` {  
`equal` , `not_equal` , `less` , `less_or_equal` ,  
`greater` , `greater_or_equal` }

## Public Member Functions

- `relational` (const `ex` &`lhs`, const `ex` &`rhs`, `operators` oper=`equal`)
- unsigned `precedence` () const override  
*Return relative operator precedence (for parenthezing output).*
- bool `info` (unsigned inf) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*

- `ex op` (`size_t i`) `const` override  
*Return operand/member at position  $i$ .*
- `ex map` (`map_function &f`) `const` override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex subs` (`const exmap &m`, unsigned `options=0`) `const` override  
*Substitute a set of objects by arbitrary expressions.*
- `void archive` (`archive_node &n`) `const` override  
*Save (a.k.a.*
- `void read_archive` (`const archive_node &n`, `lst &syms`) `override`  
*Read (a.k.a.*
- `ex canonical` () `const`  
*Returns an equivalent relational with zero right-hand side.*
- `ex lhs` () `const`
- `ex rhs` () `const`
- `operator safe_bool` () `const`  
*Cast the relational into a Boolean, mainly for evaluation within an if-statement.*
- `safe_bool operator!` () `const`

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (`const basic &other`)
- `const basic & operator=` (`const basic &other`)  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () `const`  
*Create a clone of this object on the heap.*
- `virtual ex eval` () `const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () `const`  
*Evaluate object numerically.*
- `virtual ex evalm` () `const`  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () `const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed` (`const basic &i`) `const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print` (`const print_context &c`, unsigned `level=0`) `const`  
*Output to stream.*
- `virtual void dbgprint` () `const`  
*Little wrapper around `print` to be called within a debugger.*
- `virtual void dbgprinttree` () `const`  
*Little wrapper around `printtree` to be called within a debugger.*
- `virtual ex operator[]` (`const ex &index`) `const`
- `virtual ex operator[]` (`size_t i`) `const`
- `virtual ex & let_op` (`size_t i`)  
*Return modifiable operand/member at position  $i$ .*
- `virtual ex & operator[]` (`const ex &index`)
- `virtual ex & operator[]` (`size_t i`)
- `virtual bool has` (`const ex &other`, unsigned `options=0`) `const`

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex` `coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex` `expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex` `collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex` `series` (const `relational` &r, int `order`, unsigned `options`=0) const
- Default implementation of `ex::series()`.*

  - virtual `ex` `normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex` `to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex` `to_polynomial` (`exmap` &repl) const
- virtual `numeric` `integer_content` () const
- virtual `ex` `smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric` `max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector` `get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex` `add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex` `scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual `ex` `conjugate` () const
- virtual `ex` `real_part` () const
- virtual `ex` `imag_part` () const
- template<class T >
- void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - `ex` `subs_one_level` (const `exmap` &m, unsigned `options`) const

*Helper function for `subs()`.*
- `ex` `diff` (const `symbol` &s, unsigned nth=1) const
- Default interface of nth derivative `ex::diff(s, n)`.*

  - int `compare` (const `basic` &other) const

*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const

*Test for syntactic equality.*

- const [basic](#) & [hold](#) () const

*Stop further evaluation.*

- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const

*Set some [status\\_flags](#).*

- const [basic](#) & [clearflag](#) (unsigned f) const

*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- [ex eval\\_ncmul](#) (const [exvector](#) &v) const override
- bool [match\\_same\\_type](#) (const [basic](#) &other) const override
 

*Returns true if the attributes of two objects are similar enough for a match.*
- unsigned [return\\_type](#) () const override
- [return\\_type\\_t](#) [return\\_type\\_tinfo](#) () const override
- unsigned [calchash](#) () const override
 

*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex derivative](#) (const [symbol](#) &s) const
 

*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const
 

*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const
 

*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const
 

*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
 

*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const
 

*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const
 

*Python parsable output to stream.*

## Protected Attributes

- [ex lh](#)
- [ex rh](#)
- [operators o](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## Private Types

- typedef void(safe\_bool\_helper::\* [safe\\_bool](#)) ()

## Private Member Functions

- [safe\\_bool make\\_safe\\_bool](#) (bool) const

## 6.145.1 Detailed Description

This class holds a relation consisting of two expressions and a logical relation between them.

## 6.145.2 Member Typedef Documentation

### 6.145.2.1 [safe\\_bool](#)

```
typedef void(safe_bool_helper::* GiNaC::relational::safe_bool) () [private]
```

## 6.145.3 Member Enumeration Documentation

### 6.145.3.1 [operators](#)

```
enum GiNaC::relational::operators
```

#### Enumerator

|                                  |  |
|----------------------------------|--|
| <a href="#">equal</a>            |  |
| <a href="#">not_equal</a>        |  |
| <a href="#">less</a>             |  |
| <a href="#">less_or_equal</a>    |  |
| <a href="#">greater</a>          |  |
| <a href="#">greater_or_equal</a> |  |

## 6.145.4 Constructor & Destructor Documentation

### 6.145.4.1 relational()

```
GiNaC::relational::relational (
 const ex & lhs,
 const ex & rhs,
 operators oper = equal)
```

Referenced by [canonical\(\)](#), and [subs\(\)](#).

## 6.145.5 Member Function Documentation

### 6.145.5.1 precedence()

```
unsigned GiNaC::relational::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

Referenced by [do\\_print\(\)](#).

### 6.145.5.2 info()

```
bool GiNaC::relational::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [equal](#), [greater](#), [greater\\_or\\_equal](#), [less](#), [less\\_or\\_equal](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::relation](#), [GiNaC::info\\_flags::relation\\_equal](#), [GiNaC::info\\_flags::relation\\_greater](#), [GiNaC::info\\_flags::relation\\_greater\\_or\\_equal](#), [GiNaC::info\\_flags::relation\\_less](#), [GiNaC::info\\_flags::relation\\_less\\_or\\_equal](#), and [GiNaC::info\\_flags::relation\\_not\\_equal](#).

Referenced by [operator safe\\_bool\(\)](#).

### 6.145.5.3 nops()

```
size_t GiNaC::relational::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

#### 6.145.5.4 op()

```
ex GiNaC::relational::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), and [rh](#).

#### 6.145.5.5 map()

```
ex GiNaC::relational::map (
 map_function & f) const [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [lh](#), [o](#), and [rh](#).

#### 6.145.5.6 subs()

```
ex GiNaC::relational::subs (
 const exmap & m,
 unsigned options = 0) const [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::are\\_ex\\_trivially\\_equal\(\)](#), [lh](#), [m](#), [o](#), [options](#), [relational\(\)](#), [rh](#), [GiNaC::ex::subs\(\)](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

#### 6.145.5.7 archive()

```
void GiNaC::relational::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [lh](#), [n](#), [o](#), and [rh](#).

### 6.145.5.8 read\_archive()

```
void GiNaC::relational::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [lh](#), [n](#), [o](#), and [rh](#).

### 6.145.5.9 canonical()

```
ex GiNaC::relational::canonical () const
```

Returns an equivalent relational with zero right-hand side.

References [GiNaC::\\_ex0](#), [lh](#), [o](#), [relational\(\)](#), and [rh](#).

### 6.145.5.10 eval\_ncmul()

```
ex GiNaC::relational::eval_ncmul (
 const exvector & v) const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::ex::eval\\_ncmul\(\)](#), and [lh](#).

### 6.145.5.11 match\_same\_type()

```
bool GiNaC::relational::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [o](#).

**6.145.5.12 return\_type()**

```
unsigned GiNaC::relational::return_type () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), [GiNaC::ex::return\\_type\(\)](#), and [rh](#).

**6.145.5.13 return\_type\_tinfo()**

```
return_type_t GiNaC::relational::return_type_tinfo () const [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), [lh](#), [GiNaC::ex::return\\_type\\_tinfo\(\)](#), and [rh](#).

**6.145.5.14 calchash()**

```
unsigned GiNaC::relational::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [equal](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::ex::gethash\(\)](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [less](#), [less\\_or\\_equal](#), [lh](#), [GiNaC::make\\_hash\\_seed\(\)](#), [not\\_equal](#), [o](#), [rh](#), [GiNaC::rotate\\_left\(\)](#), and [GiNaC::basic::setflag\(\)](#).

**6.145.5.15 do\_print()**

```
void GiNaC::relational::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [lh](#), [o](#), [precedence\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_operator\(\)](#), and [rh](#).

**6.145.5.16 do\_print\_python\_repr()**

```
void GiNaC::relational::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [lh](#), [o](#), [GiNaC::ex::print\(\)](#), [GiNaC::print\\_operator\(\)](#), and [rh](#).

**6.145.5.17 lhs()**

```
ex GiNaC::relational::lhs () const [inline]
```

References [lh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::tan\\_series\(\)](#), and [GiNaC::tanh\\_series\(\)](#).

**6.145.5.18 rhs()**

```
ex GiNaC::relational::rhs () const [inline]
```

References [rh](#).

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), and [GiNaC::log\\_series\(\)](#).

**6.145.5.19 make\_safe\_bool()**

```
relational::safe_bool GiNaC::relational::make_safe_bool (
 bool cond) const [private]
```

References [GiNaC::relational::safe\\_bool\\_helper::nonnull\(\)](#).

Referenced by [operator safe\\_bool\(\)](#), and [operator!\(\)](#).

**6.145.5.20 operator safe\_bool()**

```
GiNaC::relational::operator relational::safe_bool () const
```

Cast the relational into a Boolean, mainly for evaluation within an if-statement.

Note that  $(a < b) == \text{false}$  does not imply  $(a \geq b) == \text{true}$  in the general symbolic case. A false result means the comparison is either false or undecidable (except of course for  $!=$ , where true means either unequal or undecidable).

References [GiNaC::num0\\_p](#), [equal](#), [greater](#), [greater\\_or\\_equal](#), [GiNaC::ex::info\(\)](#), [info\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [GiNaC::is\\_zero\(\)](#), [less](#), [less\\_or\\_equal](#), [lh](#), [make\\_safe\\_bool\(\)](#), [GiNaC::info\\_flags::negative](#), [GiNaC::info\\_flags::nonnegative](#), [not\\_equal](#), [o](#), [GiNaC::info\\_flags::positive](#), and [rh](#).

**6.145.5.21 operator"!()**

```
relational::safe_bool GiNaC::relational::operator! () const [inline]
```

References [make\\_safe\\_bool\(\)](#).

## 6.145.6 Member Data Documentation

### 6.145.6.1 lh

`ex GiNaC::relational::lh` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [eval\\_ncmul\(\)](#), [lhs\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), and [subs\(\)](#).

### 6.145.6.2 rh

`ex GiNaC::relational::rh` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [map\(\)](#), [op\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), [return\\_type\(\)](#), [return\\_type\\_tinfo\(\)](#), [rhs\(\)](#), and [subs\(\)](#).

### 6.145.6.3 o

`operators GiNaC::relational::o` [protected]

Referenced by [archive\(\)](#), [calchash\(\)](#), [canonical\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [info\(\)](#), [map\(\)](#), [match\\_same\\_type\(\)](#), [operator safe\\_bool\(\)](#), [read\\_archive\(\)](#), and [subs\(\)](#).

The documentation for this class was generated from the following files:

- [relational.h](#)
- [relational.cpp](#)

## 6.146 GiNaC::remember\_strategies Class Reference

Strategies how to clean up the function remember cache.

```
#include <flags.h>
```

### Public Types

- enum { [delete\\_never](#) , [delete\\_lru](#) , [delete\\_lfu](#) , [delete\\_cyclic](#) }

### 6.146.1 Detailed Description

Strategies how to clean up the function remember cache.

See also

[remember\\_table](#)

## 6.146.2 Member Enumeration Documentation

### 6.146.2.1 anonymous enum

anonymous enum

## Enumerator

|               |                              |
|---------------|------------------------------|
| delete_never  | Let table grow indefinitely. |
| delete_lru    | Least recently used.         |
| delete_lfu    | Least frequently used.       |
| delete_cyclic | First (oldest) one in list.  |

The documentation for this class was generated from the following file:

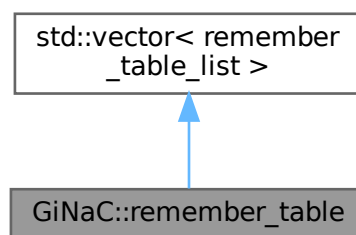
- [flags.h](#)

## 6.147 GiNaC::remember\_table Class Reference

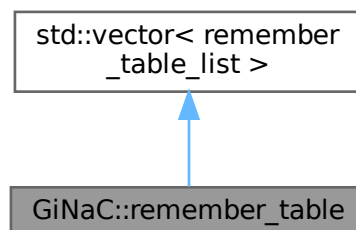
The remember table is organized like an n-fold associative cache in a microprocessor.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table:



Collaboration diagram for GiNaC::remember\_table:



### Public Member Functions

- [remember\\_table](#) ()
- [remember\\_table](#) (unsigned s, unsigned as, unsigned strat)
- bool [lookup\\_entry](#) (function const &f, [ex](#) &result) const
- void [add\\_entry](#) (function const &f, [ex](#) const &result)
- void [clear\\_all\\_entries](#) ()
- void [show\\_statistics](#) (std::ostream &os, unsigned level) const

### Static Public Member Functions

- static std::vector< [remember\\_table](#) > & [remember\\_tables](#) ()

### Protected Member Functions

- void [init\\_table](#) ()

### Protected Attributes

- unsigned [table\\_size](#)
- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

## 6.147.1 Detailed Description

The remember table is organized like an n-fold associative cache in a microprocessor.

The table has a width of 's' (which is rounded to [table\\_size](#), some power of 2 near 's', internally) and a depth of 'as' (unless you choose that entries are never discarded). The place where an entry is stored depends on the hashvalue of the parameters of the function (this corresponds to the address of byte to be cached). The '[log\\_2\(table\\_size\)](#)' least significant bits of this hashvalue give the slot in which the entry will be stored or looked up. Each slot can take up to 'as' entries. If a slot is full, an older entry is removed by one of the following strategies:

- oldest entry (the first one in the list)
- least recently used (the one with the lowest 'last\_access')
- least frequently used (the one with the lowest 'successful\_hits') or all entries are kept which means that the table grows indefinitely.

## 6.147.2 Constructor & Destructor Documentation

### 6.147.2.1 [remember\\_table](#)() [1/2]

```
GiNaC::remember_table::remember_table ()
```

References [GiNaC::remember\\_strategies::delete\\_never](#), [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

### 6.147.2.2 remember\_table() [2/2]

```
GiNaC::remember_table::remember_table (
 unsigned s,
 unsigned as,
 unsigned strat)
```

References [init\\_table\(\)](#), [GiNaC::log2\(\)](#), and [table\\_size](#).

## 6.147.3 Member Function Documentation

### 6.147.3.1 lookup\_entry()

```
bool GiNaC::remember_table::lookup_entry (
 function const & f,
 ex & result) const
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 6.147.3.2 add\_entry()

```
void GiNaC::remember_table::add_entry (
 function const & f,
 ex const & result)
```

References [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), and [table\\_size](#).

### 6.147.3.3 clear\_all\_entries()

```
void GiNaC::remember_table::clear_all_entries ()
```

References [init\\_table\(\)](#).

### 6.147.3.4 show\_statistics()

```
void GiNaC::remember_table::show_statistics (
 std::ostream & os,
 unsigned level) const
```

### 6.147.3.5 remember\_tables()

```
std::vector< remember_table > & GiNaC::remember_table::remember_tables () [static]
```

Referenced by [GiNaC::function::lookup\\_remember\\_table\(\)](#), [GiNaC::function::register\\_new\(\)](#), and [GiNaC::function::store\\_remember\\_t](#)

### 6.147.3.6 `init_table()`

```
void GiNaC::remember_table::init_table () [protected]
```

References [max\\_assoc\\_size](#), [remember\\_strategy](#), and [table\\_size](#).

Referenced by [clear\\_all\\_entries\(\)](#), and [remember\\_table\(\)](#).

## 6.147.4 Member Data Documentation

### 6.147.4.1 `table_size`

```
unsigned GiNaC::remember_table::table_size [protected]
```

Referenced by [add\\_entry\(\)](#), [init\\_table\(\)](#), [lookup\\_entry\(\)](#), [remember\\_table\(\)](#), and [remember\\_table\(\)](#).

### 6.147.4.2 `max_assoc_size`

```
unsigned GiNaC::remember_table::max_assoc_size [protected]
```

Referenced by [init\\_table\(\)](#), and [remember\\_table\(\)](#).

### 6.147.4.3 `remember_strategy`

```
unsigned GiNaC::remember_table::remember_strategy [protected]
```

Referenced by [init\\_table\(\)](#), and [remember\\_table\(\)](#).

The documentation for this class was generated from the following files:

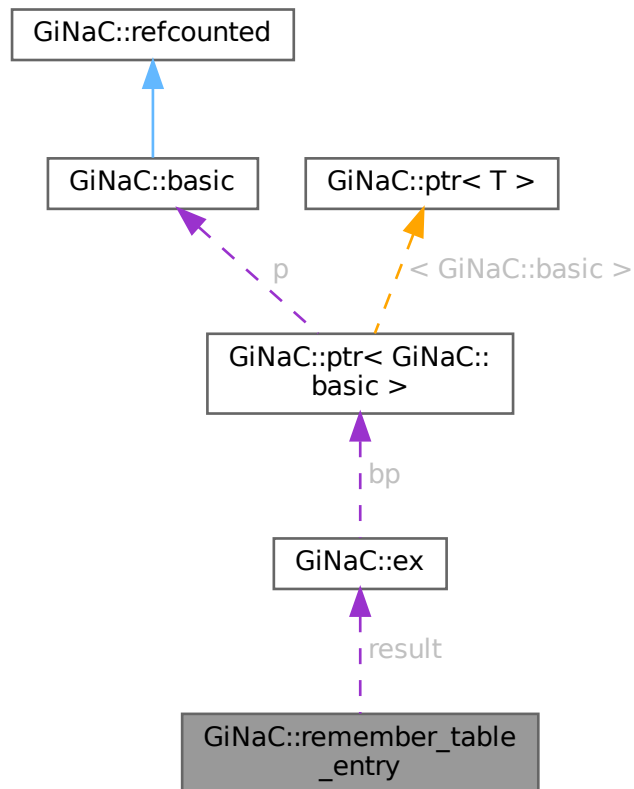
- [remember.h](#)
- [remember.cpp](#)

## 6.148 GiNaC::remember\_table\_entry Class Reference

A single entry in the remember table of a function.

```
#include <remember.h>
```

Collaboration diagram for GiNaC::remember\_table\_entry:



### Public Member Functions

- `remember_table_entry` (function const &f, ex const &r)
- bool `is_equal` (function const &f) const
- ex `get_result` () const
- unsigned long `get_last_access` () const
- unsigned long `get_successful_hits` () const

### Protected Attributes

- unsigned `hashvalue`
- `exvector` seq
- ex `result`
- unsigned long `last_access`
- unsigned `successful_hits`

## Static Protected Attributes

- static unsigned long [access\\_counter](#) = 0

## 6.148.1 Detailed Description

A single entry in the remember table of a function.

Needs to be a friend of class function to access 'seq'. 'last\_access' and 'successful\_hits' are updated at each successful 'is\_equal'.

## 6.148.2 Constructor & Destructor Documentation

### 6.148.2.1 remember\_table\_entry()

```
GiNaC::remember_table_entry::remember_table_entry (
 function const & f,
 ex const & r)
```

References [access\\_counter](#), [last\\_access](#), and [successful\\_hits](#).

## 6.148.3 Member Function Documentation

### 6.148.3.1 is\_equal()

```
bool GiNaC::remember_table_entry::is_equal (
 function const & f) const
```

References [access\\_counter](#), [GiNaC::basic::gethash\(\)](#), [GINAC\\_ASSERT](#), [hashvalue](#), [is\\_equal\(\)](#), [last\\_access](#), [GiNaC::container\\_storage< C >::seq](#), [seq](#), and [successful\\_hits](#).

Referenced by [is\\_equal\(\)](#).

### 6.148.3.2 get\_result()

```
ex GiNaC::remember_table_entry::get_result () const [inline]
```

References [result](#).

### 6.148.3.3 get\_last\_access()

```
unsigned long GiNaC::remember_table_entry::get_last_access () const [inline]
```

References [last\\_access](#).

### 6.148.3.4 get\_successful\_hits()

```
unsigned long GiNaC::remember_table_entry::get_successful_hits () const [inline]
```

References [successful\\_hits](#).

## 6.148.4 Member Data Documentation

### 6.148.4.1 hashvalue

```
unsigned GiNaC::remember_table_entry::hashvalue [protected]
```

Referenced by [is\\_equal\(\)](#).

### 6.148.4.2 seq

```
exvector GiNaC::remember_table_entry::seq [protected]
```

Referenced by [is\\_equal\(\)](#).

### 6.148.4.3 result

```
ex GiNaC::remember_table_entry::result [protected]
```

Referenced by [get\\_result\(\)](#).

### 6.148.4.4 last\_access

```
unsigned long GiNaC::remember_table_entry::last_access [mutable], [protected]
```

Referenced by [get\\_last\\_access\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

### 6.148.4.5 successful\_hits

```
unsigned GiNaC::remember_table_entry::successful_hits [mutable], [protected]
```

Referenced by [get\\_successful\\_hits\(\)](#), [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

### 6.148.4.6 access\_counter

```
unsigned long GiNaC::remember_table_entry::access_counter = 0 [static], [protected]
```

Referenced by [is\\_equal\(\)](#), and [remember\\_table\\_entry\(\)](#).

The documentation for this class was generated from the following files:

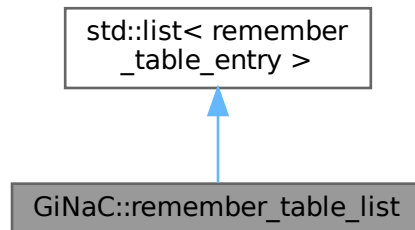
- [remember.h](#)
- [remember.cpp](#)

## 6.149 GiNaC::remember\_table\_list Class Reference

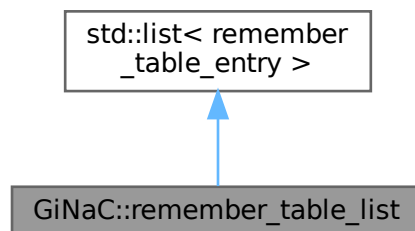
A list of entries in the remember table having some least significant bits of the hashvalue in common.

```
#include <remember.h>
```

Inheritance diagram for GiNaC::remember\_table\_list:



Collaboration diagram for GiNaC::remember\_table\_list:



### Public Member Functions

- [remember\\_table\\_list](#) (unsigned as, unsigned strat)
- void [add\\_entry](#) (function const &f, ex const &result)
- bool [lookup\\_entry](#) (function const &f, ex &result) const

### Protected Attributes

- unsigned [max\\_assoc\\_size](#)
- unsigned [remember\\_strategy](#)

## 6.149.1 Detailed Description

A list of entries in the remember table having some least significant bits of the hashvalue in common.

## 6.149.2 Constructor & Destructor Documentation

### 6.149.2.1 remember\_table\_list()

```
GiNaC::remember_table_list::remember_table_list (
 unsigned as,
 unsigned strat)
```

References [max\\_assoc\\_size](#), and [remember\\_strategy](#).

## 6.149.3 Member Function Documentation

### 6.149.3.1 add\_entry()

```
void GiNaC::remember_table_list::add_entry (
 function const & f,
 ex const & result)
```

References [GiNaC::remember\\_strategies::delete\\_cyclic](#), [GiNaC::remember\\_strategies::delete\\_ifu](#), [GiNaC::remember\\_strategies::delete\\_never](#), [GINAC\\_ASSERT](#), [max\\_assoc\\_size](#), and [remember\\_strategy](#).

### 6.149.3.2 lookup\_entry()

```
bool GiNaC::remember_table_list::lookup_entry (
 function const & f,
 ex & result) const
```

## 6.149.4 Member Data Documentation

### 6.149.4.1 max\_assoc\_size

```
unsigned GiNaC::remember_table_list::max_assoc_size [protected]
```

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

### 6.149.4.2 remember\_strategy

```
unsigned GiNaC::remember_table_list::remember_strategy [protected]
```

Referenced by [add\\_entry\(\)](#), and [remember\\_table\\_list\(\)](#).

The documentation for this class was generated from the following files:

- [remember.h](#)
- [remember.cpp](#)

## 6.150 GiNaC::return\_type\_t Struct Reference

To distinguish between different kinds of non-commutative objects.

```
#include <registrar.h>
```

### Public Member Functions

- bool `operator<` (const `return_type_t` &other) const  
*Strict weak ordering (so one can put `return_type_t`'s into a STL container).*
- bool `operator==` (const `return_type_t` &other) const
- bool `operator!=` (const `return_type_t` &other) const

### Public Attributes

- `std::type_info` const \* `tinfo`  
*to distinguish between non-commutative objects of different type.*
- unsigned `rl`  
*to distinguish between non-commutative objects of the same type.*

### 6.150.1 Detailed Description

To distinguish between different kinds of non-commutative objects.

### 6.150.2 Member Function Documentation

#### 6.150.2.1 `operator<()`

```
bool GiNaC::return_type_t::operator< (
 const return_type_t & other) const [inline]
```

Strict weak ordering (so one can put `return_type_t`'s into a STL container).

References `rl`, and `tinfo`.

#### 6.150.2.2 `operator==()`

```
bool GiNaC::return_type_t::operator== (
 const return_type_t & other) const [inline]
```

References `rl`, and `tinfo`.

Referenced by `operator!=()`.

### 6.150.2.3 operator"!=()

```
bool GiNaC::return_type_t::operator!= (
 const return_type_t & other) const [inline]
```

References [operator==\(.\)](#).

## 6.150.3 Member Data Documentation

### 6.150.3.1 tinfo

```
std::type_info const* GiNaC::return_type_t::tinfo
```

to distinguish between non-commutative objects of different type.

Referenced by [GiNaC::is\\_clifford\\_tinfo\(\)](#), [GiNaC::is\\_color\\_tinfo\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(.\)](#), and [GiNaC::basic::return\\_type\\_tinfo\(\)](#).

### 6.150.3.2 rl

```
unsigned GiNaC::return_type_t::rl
```

to distinguish between non-commutative objects of the same type.

Think of gamma matrices with different representation labels.

Referenced by [GiNaC::get\\_representation\\_label\(\)](#), [GiNaC::make\\_return\\_type\\_t\(\)](#), [operator<\(\)](#), [operator==\(.\)](#), [GiNaC::basic::return\\_type\\_tinfo\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::return\\_type\\_tinfo\(\)](#).

The documentation for this struct was generated from the following file:

- [registrar.h](#)

## 6.151 GiNaC::return\_types Class Reference

```
#include <flags.h>
```

### Public Types

- enum { [commutative](#) , [noncommutative](#) , [noncommutative\\_composite](#) }

## 6.151.1 Member Enumeration Documentation

### 6.151.1.1 anonymous enum

```
anonymous enum
```

## Enumerator

|                          |  |
|--------------------------|--|
| commutative              |  |
| noncommutative           |  |
| noncommutative_composite |  |

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.152 GiNaC::relational::safe\_bool\_helper Struct Reference

### Public Member Functions

- void [nonnull](#) ()

### 6.152.1 Member Function Documentation

#### 6.152.1.1 nonnull()

```
void GiNaC::relational::safe_bool_helper::nonnull () [inline]
```

Referenced by [GiNaC::relational::make\\_safe\\_bool\(\)](#).

The documentation for this struct was generated from the following file:

- [relational.h](#)

## 6.153 GiNaC::scalar\_products Class Reference

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

```
#include <indexed.h>
```

### Public Member Functions

- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &sp)  
*Register scalar product pair and its value.*
- void [add](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim, const [ex](#) &sp)  
*Register scalar product pair and its value for a specific space dimension.*
- void [add\\_vectors](#) (const [lst](#) &l, const [ex](#) &dim=[wild](#)())  
*Register list of vectors.*
- void [clear](#) ()  
*Clear all registered scalar products.*
- bool [is\\_defined](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Check whether scalar product pair is defined.*
- [ex](#) [evaluate](#) (const [ex](#) &v1, const [ex](#) &v2, const [ex](#) &dim) const  
*Return value of defined scalar product pair.*
- void [debugprint](#) () const

## Protected Attributes

- [smap](#) `spm`

### 6.153.1 Detailed Description

Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).

See also

[simplify\\_indexed](#)

### 6.153.2 Member Function Documentation

#### 6.153.2.1 `add()` [1/2]

```
void GiNaC::scalar_products::add (
 const ex & v1,
 const ex & v2,
 const ex & sp)
```

Register scalar product pair and its value.

References [spm](#).

Referenced by [add\\_vectors\(\)](#).

#### 6.153.2.2 `add()` [2/2]

```
void GiNaC::scalar_products::add (
 const ex & v1,
 const ex & v2,
 const ex & dim,
 const ex & sp)
```

Register scalar product pair and its value for a specific space dimension.

References [spm](#).

#### 6.153.2.3 `add_vectors()`

```
void GiNaC::scalar_products::add_vectors (
 const lst & l,
 const ex & dim = wild\(\))
```

Register list of vectors.

This adds all possible pairs of products  $a.i * b.i$  with the value  $a*b$  (note that this is not a scalar vector product but an ordinary product of scalars).

References [add\(\)](#).

#### 6.153.2.4 clear()

```
void GiNaC::scalar_products::clear ()
```

Clear all registered scalar products.

References [spm](#).

#### 6.153.2.5 is\_defined()

```
bool GiNaC::scalar_products::is_defined (
 const ex & v1,
 const ex & v2,
 const ex & dim) const
```

Check whether scalar product pair is defined.

References [GiNaC::ex::find\(\)](#), and [spm](#).

#### 6.153.2.6 evaluate()

```
ex GiNaC::scalar_products::evaluate (
 const ex & v1,
 const ex & v2,
 const ex & dim) const
```

Return value of defined scalar product pair.

References [GiNaC::ex::find\(\)](#), and [spm](#).

#### 6.153.2.7 debugprint()

```
void GiNaC::scalar_products::debugprint () const
```

References [GiNaC::spmapkey::debugprint\(\)](#), [k](#), and [spm](#).

### 6.153.3 Member Data Documentation

#### 6.153.3.1 spm

```
spmap GiNaC::scalar_products::spm [protected]
```

Referenced by [add\(\)](#), [add\(\)](#), [clear\(\)](#), [debugprint\(\)](#), [evaluate\(\)](#), and [is\\_defined\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.154 GiNaC::series\_options Class Reference

Flags to control series expansion.

```
#include <flags.h>
```

### Public Types

- enum { [suppress\\_branchcut](#) = 0x0001 }

### 6.154.1 Detailed Description

Flags to control series expansion.

### 6.154.2 Member Enumeration Documentation

#### 6.154.2.1 anonymous enum

anonymous enum

#### Enumerator

|                                    |                                                                                                                                                                                                                                                        |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">suppress_branchcut</a> | Suppress branch cuts in series expansion. Branch cuts manifest themselves as step functions, if this option is not passed. If it is passed and expansion at a point on a cut is performed, then the analytic continuation of the function is expanded. |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.155 GiNaC::solve\_algo Class Reference

Switch to control algorithm for linear system solving.

```
#include <flags.h>
```

### Public Types

- enum {  
[automatic](#) , [gauss](#) , [divfree](#) , [bareiss](#) ,  
[markowitz](#) }

### 6.155.1 Detailed Description

Switch to control algorithm for linear system solving.

## 6.155.2 Member Enumeration Documentation

### 6.155.2.1 anonymous enum

anonymous enum

#### Enumerator

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| automatic | Let the system choose. A heuristics is applied for automatic determination of a suitable algorithm.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| gauss     | <p>Gauss elimination. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$ <p>This algorithm is well-suited for numerical matrices but generally suffers from the expensive division (and computation of GCDs) at each step.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| divfree   | <p>Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$ <p>This algorithm is only there for the purpose of cross-checks. It suffers from exponential intermediate expression swell. Use it only for small systems.</p>                                                                                                                                                                                                                                                                                                                                                                        |
| bareiss   | <p>Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is <i>delayed</i> until it can be carried out without computing GCDs. If <math>m_{i,j}^{(0)}</math> are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules</p> $m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$ <p>(We have set <math>m_{-1,-1}^{(-1)} = 1</math> in order to avoid a case distinction in above formula.) It can be shown that nothing more than polynomial long division is needed for carrying out the division. This is generally the fastest algorithm for solving linear systems. In contrast to division-free elimination it only has a linear expression swell. For two-dimensional systems, the two algorithms are equivalent, however.</p> |
| markowitz | Markowitz-ordered Gaussian elimination. Same as the usual Gaussian elimination, but with additional effort spent on selecting pivots that minimize fill-in. Faster than the methods above for large sparse matrices (particularly with symbolic coefficients), otherwise slightly slower than Gaussian elimination.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

The documentation for this class was generated from the following file:

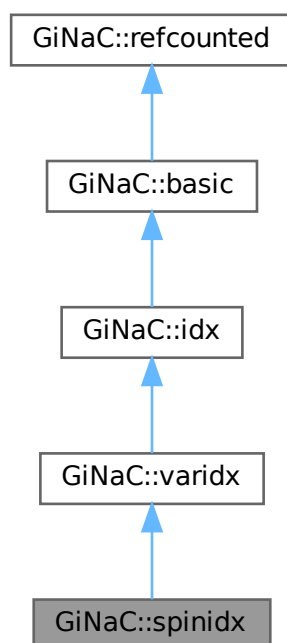
- [flags.h](#)

## 6.156 GiNaC::spinidx Class Reference

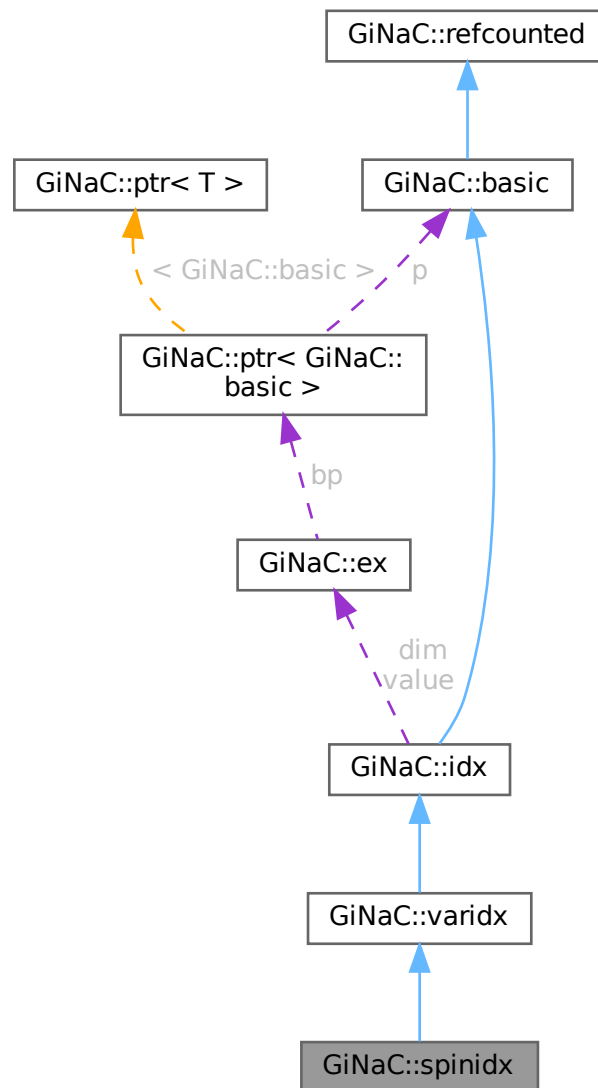
This class holds a spinor index that can be dotted or undotted and that also has a variance.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::spinidx:



Collaboration diagram for `GiNaC::spinidx`:



## Public Member Functions

- `spinidx` (const `ex` &`v`, const `ex` &`dim`=2, bool `covariant`=false, bool `dotted`=false)  
Construct index with given value, dimension, variance and dot.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- `ex conjugate` () const override
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
Load (deserialize) the object from an archive node.

- `bool is_dotted () const`  
*Check whether the index is dotted.*
- `bool is_undotted () const`  
*Check whether the index is not dotted.*
- `ex toggle_dot () const`  
*Make a new index with the same value and variance but the opposite dottedness.*
- `ex toggle_variance_dot () const`  
*Make a new index with the same value but opposite variance and dottedness.*

## Public Member Functions inherited from `GiNaC::varidx`

- `varidx (const ex &v, const ex &dim, bool covariant=false)`  
*Construct index with given value, dimension and variance.*
- `bool is_dummy_pair_same_type (const basic &other) const override`  
*Check whether the index forms a dummy index pair with another index of the same type.*
- `void archive (archive_node &n) const override`  
*Save (serialize) the object into archive node.*
- `void read_archive (const archive_node &n, lst &symbols) override`  
*Load (deserialize) the object from an archive node.*
- `bool is_covariant () const`  
*Check whether the index is covariant.*
- `bool is_contravariant () const`  
*Check whether the index is contravariant (not covariant).*
- `ex toggle_variance () const`  
*Make a new index with the same value but the opposite variance.*

## Public Member Functions inherited from `GiNaC::idx`

- `idx (const ex &v, const ex &dim)`  
*Construct index with given value and dimension.*
- `bool info (unsigned inf) const override`  
*Information about the object.*
- `size_t nops () const override`  
*Number of operands/members.*
- `ex op (size_t i) const override`  
*Return operand/member at position i.*
- `ex map (map_function &f) const override`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf () const override`  
*By default, `basic::evalf` would evaluate the index value but we don't want a.1 to become a.*
- `ex subs (const exmap &m, unsigned options=0) const override`  
*Substitute a set of objects by arbitrary expressions.*
- `ex get_value () const`  
*Get value of index.*
- `bool is_numeric () const`  
*Check whether the index is numeric.*
- `bool is_symbolic () const`  
*Check whether the index is symbolic.*
- `ex get_dim () const`

- *Get dimension of index space.*
- `bool is_dim_numeric () const`  
*Check whether the dimension is numeric.*
- `bool is_dim_symbolic () const`  
*Check whether the dimension is symbolic.*
- `ex replace_dim (const ex &new_dim) const`  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim (const idx &other) const`  
*Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate () const`  
*Create a clone of this object on the heap.*
- `virtual ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- `virtual ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- `virtual void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- `virtual void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- `virtual ex operator[] (const ex &index) const`
- `virtual ex operator[] (size_t i) const`
- `virtual ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- `virtual ex & operator[] (const ex &index)`
- `virtual ex & operator[] (size_t i)`
- `virtual bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- `virtual bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- `virtual void accept (GiNaC::visitor &v) const`
- `virtual bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- `virtual int degree (const ex &s) const`  
*Return degree of highest power in object s.*

- virtual int `ldegree` (const `ex` &`s`) const  
*Return degree of lowest power in object `s`.*
- virtual `ex coeff` (const `ex` &`s`, int `n=1`) const  
*Return coefficient of degree `n` in object `s`.*
- virtual `ex expand` (unsigned `options=0`) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &`s`, bool `distributed=false`) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &`r`, int `order`, unsigned `options=0`) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &`repl`, `exmap` &`rev_lookup`, `lst` &`modifier`) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &`repl`) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &`repl`) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &`xi`) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &`self`, const `ex` &`other`) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &`self`, const `numeric` &`other`) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` `self`, `exvector::iterator` `other`, `exvector` &`v`) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class `T` >  
void `print_dispatch` (const `print_context` &`c`, unsigned `level`) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &`ri`, const `print_context` &`c`, unsigned `level`) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::varidx](#)

- bool [match\\_same\\_type](#) (const [basic](#) &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::idx](#)

- [ex\\_derivative](#) (const [symbol](#) &s) const override  
*Implementation of [ex::diff\(\)](#) for an index always returns 0.*
- unsigned [calchash](#) () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [print\\_index](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_csrc](#) (const [print\\_csrc](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- bool `dotted`

## Protected Attributes inherited from [GiNaC::varidx](#)

- bool `covariant`  
 *$x.mu$ , default is contravariant:  $x \sim mu$*

## Protected Attributes inherited from [GiNaC::idx](#)

- `ex` value  
*Expression that constitutes the index (numeric or symbolic name)*
- `ex` dim  
*Dimension of space (can be symbolic or numeric)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

### 6.156.1 Detailed Description

This class holds a spinor index that can be dotted or undotted and that also has a variance.

This is used in the Weyl-van-der-Waerden formalism where the dot indicates complex conjugation. There is an associated (asymmetric) metric tensor that can be used to raise/lower spinor indices.

### 6.156.2 Constructor & Destructor Documentation

#### 6.156.2.1 `spinidx()`

```
GiNaC::spinidx::spinidx (
 const ex & v,
 const ex & dim = 2,
 bool covariant = false,
 bool dotted = false)
```

Construct index with given value, dimension, variance and dot.

#### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>v</i>         | Value of index (numeric or symbolic)            |
| <i>dim</i>       | Dimension of index space (numeric or symbolic)  |
| <i>covariant</i> | Make covariant index (default is contravariant) |
| <i>dotted</i>    | Make covariant dotted (default is undotted)     |

## 6.156.3 Member Function Documentation

### 6.156.3.1 `is_dummy_pair_same_type()`

```
bool GiNaC::spinidx::is_dummy_pair_same_type (
 const basic & other) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

References [dotted](#).

### 6.156.3.2 `conjugate()`

```
ex GiNaC::spinidx::conjugate () const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [toggle\\_dot\(\)](#).

### 6.156.3.3 `archive()`

```
void GiNaC::spinidx::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

References [dotted](#), and [n](#).

### 6.156.3.4 `read_archive()`

```
void GiNaC::spinidx::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

References [dotted](#), and [n](#).

### 6.156.3.5 match\_same\_type()

```
bool GiNaC::spinidx::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

References [dotted](#), and [GINAC\\_ASSERT](#).

### 6.156.3.6 is\_dotted()

```
bool GiNaC::spinidx::is_dotted () const [inline]
```

Check whether the index is dotted.

References [dotted](#).

### 6.156.3.7 is\_undotted()

```
bool GiNaC::spinidx::is_undotted () const [inline]
```

Check whether the index is not dotted.

References [dotted](#).

### 6.156.3.8 toggle\_dot()

```
ex GiNaC::spinidx::toggle_dot () const
```

Make a new index with the same value and variance but the opposite dottedness.

References [GiNaC::basic::clearflag\(\)](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

Referenced by [conjugate\(\)](#).

### 6.156.3.9 toggle\_variance\_dot()

```
ex GiNaC::spinidx::toggle_variance_dot () const
```

Make a new index with the same value but opposite variance and dottedness.

References [GiNaC::basic::clearflag\(\)](#), [GiNaC::varidx::covariant](#), [dotted](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_cal](#).

### 6.156.3.10 do\_print()

```
void GiNaC::spinidx::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::varidx::covariant](#), [dotted](#), and [GiNaC::idx::print\\_index\(\)](#).

### 6.156.3.11 do\_print\_latex()

```
void GiNaC::spinidx::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), [dotted](#), and [GiNaC::idx::print\\_index\(\)](#).

### 6.156.3.12 do\_print\_tree()

```
void GiNaC::spinidx::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::varidx::covariant](#), [GiNaC::idx::dim](#), [dotted](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [GiNaC::idx::value](#).

## 6.156.4 Member Data Documentation

### 6.156.4.1 dotted

```
bool GiNaC::spinidx::dotted [protected]
```

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_tree\(\)](#), [is\\_dotted\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [is\\_undotted\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_dot\(\)](#), and [toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

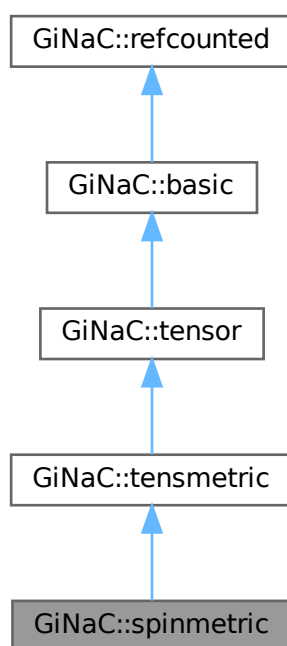
- [idx.h](#)
- [idx.cpp](#)

## 6.157 GiNaC::spinmetric Class Reference

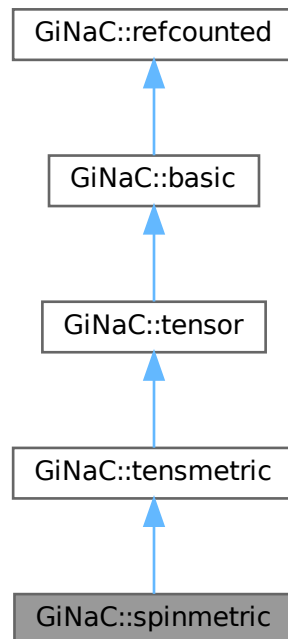
This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::spinmetric:



Collaboration diagram for `GiNaC::spinmetric`:



### Public Member Functions

- `bool info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed spinor metric with something else.*

### Public Member Functions inherited from `GiNaC::tensmetric`

- `bool info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

Public Member Functions inherited from **GiNaC::basic**

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex eval () const`  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf () const`  
*Evaluate object numerically.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprnttree () const`  
*Little wrapper around prnttree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `bool is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`

- Return coefficient of degree  $n$  in object  $s$ .*

    - virtual `ex expand` (unsigned `options=0`) const

*Expand expression, i.e.*
  - virtual `ex collect` (const `ex` & $s$ , bool `distributed=false`) const

*Sort expanded expression in terms of powers of some object(s).*
  - virtual `ex series` (const `relational` & $r$ , int `order`, unsigned `options=0`) const

*Default implementation of `ex::series()`.*
  - virtual `ex normal` (`exmap` & $repl$ , `exmap` & $rev\_lookup$ , `lst` & $modifier$ ) const

*Default implementation of `ex::normal()`.*
  - virtual `ex to_rational` (`exmap` & $repl$ ) const

*Default implementation of `ex::to_rational()`.*
  - virtual `ex to_polynomial` (`exmap` & $repl$ ) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` & $xi$ ) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` & $self$ , const `ex` & $other$ ) const
- Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` & $self$ , const `numeric` & $other$ ) const
- Multiply an indexed expression with a scalar.*
- virtual `return_type_t return_type_info` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class  $T$  >
  - void `print_dispatch` (const `print_context` & $c$ , unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` & $ri$ , const `print_context` & $c$ , unsigned `level`) const
- Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` & $n$ ) const
- Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` & $n$ , `lst` & $syms$ )
- Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` & $m$ , unsigned `options`) const
- Helper function for `subs()`.*
- `ex diff` (const `symbol` & $s$ , unsigned `nth=1`) const
- Default interface of  $n$ th derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` & $other$ ) const
- Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` & $other$ ) const
- Test for syntactic equality.*
- const `basic` & `hold` () const
- Stop further evaluation.*
- unsigned `gethash` () const
  - const `basic` & `setflag` (unsigned `f`) const
- Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned `f`) const
- Clear some `status_flags`.*

## Public Member Functions inherited from GiNaC::refcounted

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

## Protected Member Functions

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

## Protected Member Functions inherited from GiNaC::tensmetric

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

## Protected Member Functions inherited from GiNaC::tensor

- unsigned [return\\_type](#) () const override

## Protected Member Functions inherited from GiNaC::basic

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.157.1 Detailed Description

This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.

If indexed, it must have exactly two indices of the same type which must be of class `spinidx` or a subclass and have dimension 2.

## 6.157.2 Member Function Documentation

### 6.157.2.1 `info()`

```
bool GiNaC::spinmetric::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.157.2.2 `eval_indexed()`

```
ex GiNaC::spinmetric::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::\\_ex\\_1](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), and [GiNaC::basic::op\(\)](#).

### 6.157.2.3 contract\_with()

```
bool GiNaC::spinmetric::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed spinor metric with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::\\_ex2](#), [GiNaC::\\_ex\\_2](#), [GiNaC::delta\\_tensor\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::is\\_dummy\\_pair\(\)](#), and [GiNaC::idx::is\\_symbolic\(\)](#).

### 6.157.2.4 do\_print()

```
void GiNaC::spinmetric::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.157.2.5 do\_print\_latex()

```
void GiNaC::spinmetric::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

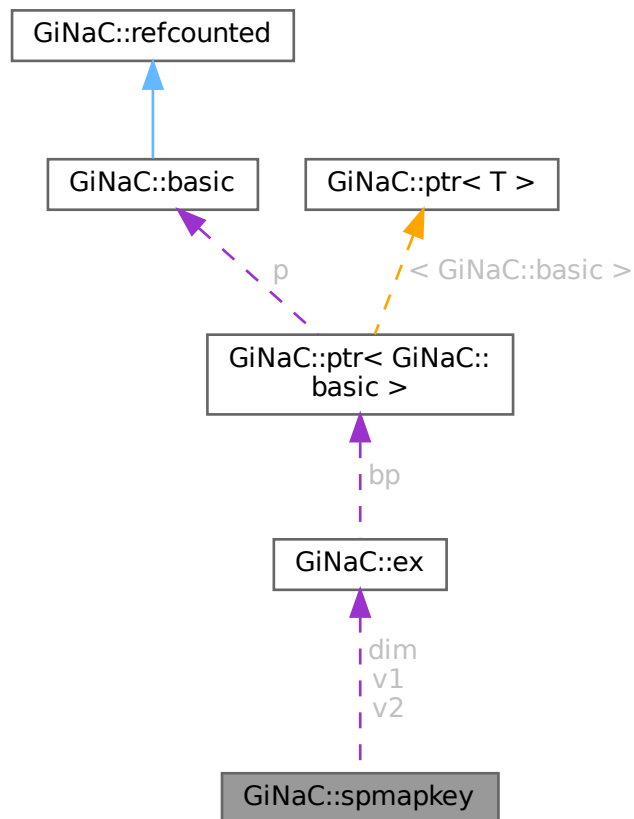
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.158 GiNaC::spmapkey Class Reference

```
#include <indexed.h>
```

Collaboration diagram for `GiNaC::spmapkey`:



## Public Member Functions

- `spmapkey()`
- `spmapkey(const ex &v1, const ex &v2, const ex &dim=wild())`
- `bool operator== (const spmapkey &other) const`
- `bool operator< (const spmapkey &other) const`
- `void debugprint() const`

## Protected Attributes

- `ex v1`
- `ex v2`
- `ex dim`

## 6.158.1 Constructor & Destructor Documentation

### 6.158.1.1 spmapkey() [1/2]

```
GiNaC::spmapkey::spmapkey () [inline]
```

### 6.158.1.2 spmapkey() [2/2]

```
GiNaC::spmapkey::spmapkey (
 const ex & v1,
 const ex & v2,
 const ex & dim = wild())
```

References [GiNaC::ex::compare\(\)](#), [GiNaC::ex::op\(\)](#), [v1](#), and [v2](#).

## 6.158.2 Member Function Documentation

### 6.158.2.1 operator==( )

```
bool GiNaC::spmapkey::operator==(
 const spmapkey & other) const
```

References [dim](#), [GiNaC::ex::is\\_equal\(\)](#), [v1](#), and [v2](#).

### 6.158.2.2 operator<( )

```
bool GiNaC::spmapkey::operator< (
 const spmapkey & other) const
```

References [GiNaC::ex::compare\(\)](#), [dim](#), [v1](#), and [v2](#).

### 6.158.2.3 debugprint( )

```
void GiNaC::spmapkey::debugprint () const
```

References [dim](#), [v1](#), and [v2](#).

Referenced by [GiNaC::scalar\\_products::debugprint\(\)](#).

## 6.158.3 Member Data Documentation

### 6.158.3.1 v1

```
ex GiNaC::spmapkey::v1 [protected]
```

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\( \)](#), and [spmapkey\(\)](#).

### 6.158.3.2 v2

```
ex GiNaC::spmapkey::v2 [protected]
```

Referenced by [debugprint\(\)](#), [operator<\(\)](#), [operator==\( \)](#), and [spmapkey\(\)](#).

### 6.158.3.3 dim

`ex GiNaC::spmapkey::dim [protected]`

Referenced by [debugprint\(\)](#), [operator<\(\)](#), and [operator==\(\)](#).

The documentation for this class was generated from the following files:

- [indexed.h](#)
- [indexed.cpp](#)

## 6.159 GiNaC::status\_flags Class Reference

Flags to store information about the state of an object.

```
#include <flags.h>
```

### Public Types

- enum {  
[dynallocated](#) = 0x0001 , [evaluated](#) = 0x0002 , [expanded](#) = 0x0004 , [hash\\_calculated](#) = 0x0008 ,  
[not\\_shareable](#) = 0x0010 , [has\\_indices](#) = 0x0020 , [has\\_no\\_indices](#) = 0x0040 , [is\\_positive](#) = 0x0080 ,  
[is\\_negative](#) = 0x0100 , [purely\\_indefinite](#) = 0x0200 }

### 6.159.1 Detailed Description

Flags to store information about the state of an object.

See also

[basic::flags](#)

### 6.159.2 Member Enumeration Documentation

#### 6.159.2.1 anonymous enum

anonymous enum

#### Enumerator

|                                 |                                                                                                                                               |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">dynallocated</a>    | heap-allocated (i.e. created by new if we want to be clever and bypass the stack,<br>See also<br><a href="#">ex::construct_from_basic()</a> ) |
| <a href="#">evaluated</a>       | <a href="#">.eval()</a> has already done its job                                                                                              |
| <a href="#">expanded</a>        | <a href="#">.expand(0)</a> has already done its job (other <a href="#">expand()</a> options ignore this flag)                                 |
| <a href="#">hash_calculated</a> | <a href="#">.calchash()</a> has already done its job                                                                                          |
| <a href="#">not_shareable</a>   | don't share instances of this object between different expressions unless explicitly asked to<br>(used by <a href="#">ex::compare()</a> )     |
| <a href="#">has_indices</a>     |                                                                                                                                               |
| <a href="#">has_no_indices</a>  |                                                                                                                                               |
| <a href="#">is_positive</a>     |                                                                                                                                               |

The documentation for this class was generated from the following file:

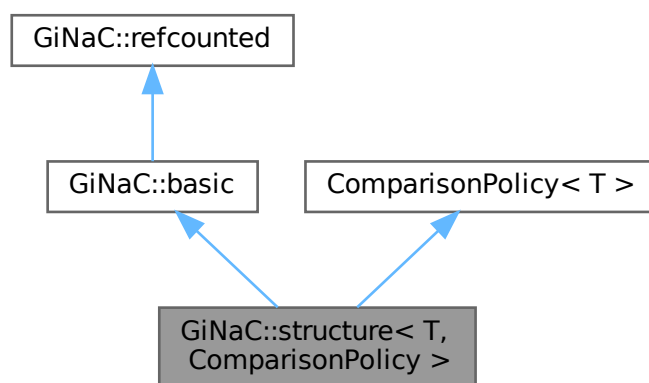
- [flags.h](#)

## 6.160 GiNaC::structure< T, ComparisonPolicy > Class Template Reference

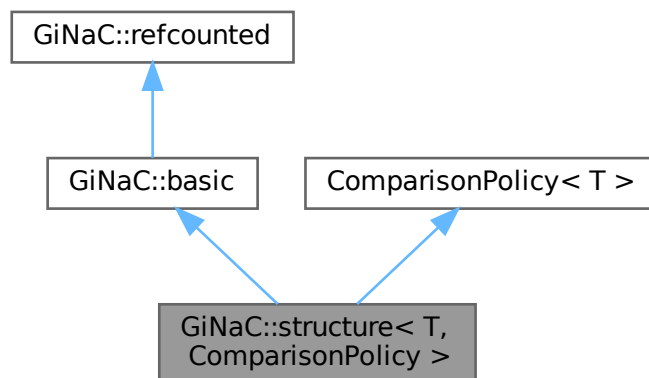
Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include <structure.h>
```

Inheritance diagram for GiNaC::structure< T, ComparisonPolicy >:



Collaboration diagram for GiNaC::structure< T, ComparisonPolicy >:



## Public Member Functions

- [structure](#) (const T &t)  
*Construct structure as a copy of a given C++ structure.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalm](#) () const override  
*Evaluate sums, products and integer powers of matrices.*
- [ex eval\\_indexed](#) (const [basic](#) &i) const override  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- void [print](#) (const [print\\_context](#) &c, unsigned level=0) const override  
*Output to stream.*
- unsigned [precedence](#) () const override  
*Return relative operator precedence (for parenthezing output).*
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- size\_t [nops](#) () const override  
*Number of operands/members.*
- [ex op](#) (size\_t i) const override  
*Return operand/member at position i.*
- [ex operator\[\]](#) (const [ex](#) &index) const override
- [ex operator\[\]](#) (size\_t i) const override
- [ex & let\\_op](#) (size\_t i) override  
*Return modifiable operand/member at position i.*
- [ex & operator\[\]](#) (const [ex](#) &index) override
- [ex & operator\[\]](#) (size\_t i) override
- bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const override  
*Test for occurrence of a pattern.*
- bool [match](#) (const [ex](#) &pattern, [exmap](#) &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex map](#) ([map\\_function](#) &f) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- int [degree](#) (const [ex](#) &s) const override  
*Return degree of highest power in object s.*
- int [ldegree](#) (const [ex](#) &s) const override  
*Return degree of lowest power in object s.*
- [ex coeff](#) (const [ex](#) &s, int n=1) const override  
*Return coefficient of degree n in object s.*
- [ex expand](#) (unsigned [options](#)=0) const override  
*Expand expression, i.e.*
- [ex collect](#) (const [ex](#) &s, bool distributed=false) const override  
*Sort expanded expression in terms of powers of some object(s).*
- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Default implementation of [ex::normal\(\)](#).*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Default implementation of [ex::to\\_rational\(\)](#).*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override

- [numeric integer\\_content](#) () const override
- [ex smod](#) (const [numeric](#) &xi) const override  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- [numeric max\\_coefficient](#) () const override  
*Implementation [ex::max\\_coefficient\(\)](#).*
- [exvector get\\_free\\_indices](#) () const override  
*Return a vector containing the free indices of an expression.*
- [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const override  
*Add two indexed expressions.*
- [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const override  
*Multiply an indexed expression with a scalar.*
- [bool contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Try to contract two indexed expressions that appear in the same product.*
- [unsigned return\\_type](#) () const override
- [return\\_type\\_t return\\_type\\_tinfo](#) () const override
- [const T \\* operator->](#) () const
- [T & get\\_struct](#) ()
- [const T & get\\_struct](#) () const

## Public Member Functions inherited from [GiNaC::basic](#)

- [virtual ~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- [const basic & operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- [virtual basic \\* duplicate](#) () const  
*Create a clone of this object on the heap.*
- [virtual ex evalf](#) () const  
*Evaluate object numerically.*
- [virtual ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- [virtual void dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- [virtual void dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- [virtual void accept](#) ([GiNaC::visitor](#) &v) const
- [virtual bool is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- [virtual ex conjugate](#) () const
- [virtual ex real\\_part](#) () const
- [virtual ex imag\\_part](#) () const
- [template<class T >](#)  
[void print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [void print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- [virtual void archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- [virtual void read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*

- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `ex eval_ncmul` (const `exvector` &`v`) const override
- `bool match_same_type` (const `basic` &`other`) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- `ex derivative` (const `symbol` &`s`) const override  
*Default implementation of `ex::diff()`.*
- `bool is_equal_same_type` (const `basic` &`other`) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual int `compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &`c`, unsigned `level`) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &`c`, unsigned `level`) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &`c`, unsigned `level`) const  
*Python parsable output to stream.*

### Static Private Member Functions

- static const char \* [get\\_class\\_name](#) ()

### Private Attributes

- T [obj](#)

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
of type [status\\_flags](#)
- unsigned [hashvalue](#)  
hash value

## 6.160.1 Detailed Description

**template<class T, template< class > class ComparisonPolicy>**  
**class GiNaC::structure< T, ComparisonPolicy >**

Wrapper template for making [GiNaC](#) classes out of C++ structures.

## 6.160.2 Constructor & Destructor Documentation

### 6.160.2.1 structure()

```
template<class T , template< class > class ComparisonPolicy>
GiNaC::structure< T, CP >::structure (
 const T & t) [inline]
```

Construct structure as a copy of a given C++ structure.

Default constructor.

## 6.160.3 Member Function Documentation

### 6.160.3.1 get\_class\_name()

```
template<class T , template< class > class ComparisonPolicy>
static const char * GiNaC::structure< T, ComparisonPolicy >::get_class_name () [inline],
[static], [private]
```

**6.160.3.2 eval()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval () const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.3 evalm()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::evalm () const [inline], [override], [virtual]
```

Evaluate sums, products and integer powers of matrices.

Reimplemented from [GiNaC::basic](#).

**6.160.3.4 eval\_ncmul()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_ncmul (
 const exvector & v) const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::hold\\_ncmul\(\)](#).

**6.160.3.5 eval\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::eval_indexed (
 const basic & i) const [inline], [override], [virtual]
```

Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::hold\(\)](#).

**6.160.3.6 print()**

```
template<class T , template< class > class ComparisonPolicy>
void GiNaC::structure< T, ComparisonPolicy >::print (
 const print_context & c,
 unsigned level = 0) const [inline], [override], [virtual]
```

Output to stream.

This performs double dispatch on the dynamic type of \*this and the dynamic type of the supplied print context.

## Parameters

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| <i>c</i>     | print context object that describes the output formatting                                                 |
| <i>level</i> | value that is used to identify the precedence or indentation level for placing parentheses and formatting |

Reimplemented from [GiNaC::basic](#).

References [c](#).

**6.160.3.7 precedence()**

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::precedence () const [inline], [override], [virtual]
```

Return relative operator precedence (for parenthezing output).

Reimplemented from [GiNaC::basic](#).

**6.160.3.8 info()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::info (
 unsigned inf) const [inline], [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.9 nops()**

```
template<class T , template< class > class ComparisonPolicy>
size_t GiNaC::structure< T, ComparisonPolicy >::nops () const [inline], [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

**6.160.3.10 op()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::op (
 size_t i) const [inline], [override], [virtual]
```

Return operand/member at position *i*.

Reimplemented from [GiNaC::basic](#).

**6.160.3.11 operator[]()** [1/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
 const ex & index) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.12 operator[]()** [2/4]

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::operator[] (
 size_t i) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.13 let\_op()**

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::let_op (
 size_t i) [inline], [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

**6.160.3.14 operator[]()** [3/4]

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
 const ex & index) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.15 operator[]()** [4/4]

```
template<class T , template< class > class ComparisonPolicy>
ex & GiNaC::structure< T, ComparisonPolicy >::operator[] (
 size_t i) [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.16 has()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::has (
 const ex & pattern,
 unsigned options = 0) const [inline], [override], [virtual]
```

Test for occurrence of a pattern.

An object 'has' a pattern if it matches the pattern itself or one of the children 'has' it. As a consequence (according to the definition of children) given  $e=x+y+z$ ,  $e.has(x)$  is true but  $e.has(x+y)$  is false.

Reimplemented from [GiNaC::basic](#).

References [options](#).

**6.160.3.17 match()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match (
 const ex & pattern,
 exmap & repl_lst) const [inline], [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

**6.160.3.18 match\_same\_type()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::match_same_type (
 const basic & other) const [inline], [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.19 subs()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::subs (
 const exmap & m,
 unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The *ex* returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), and [options](#).

**6.160.3.20 map()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::map (
 map_function & f) const [inline], [override], [virtual]
```

Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).

Reimplemented from [GiNaC::basic](#).

**6.160.3.21 degree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::degree (
 const ex & s) const [inline], [override], [virtual]
```

Return degree of highest power in object s.

Reimplemented from [GiNaC::basic](#).

**6.160.3.22 ldegree()**

```
template<class T , template< class > class ComparisonPolicy>
int GiNaC::structure< T, ComparisonPolicy >::ldegree (
 const ex & s) const [inline], [override], [virtual]
```

Return degree of lowest power in object s.

Reimplemented from [GiNaC::basic](#).

**6.160.3.23 coeff()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::coeff (
 const ex & s,
 int n = 1) const [inline], [override], [virtual]
```

Return coefficient of degree n in object s.

Reimplemented from [GiNaC::basic](#).

References [n](#).

**6.160.3.24 expand()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::expand (
 unsigned options = 0) const [inline], [override], [virtual]
```

Expand expression, i.e.

multiply it out and return the result as a new expression.

Reimplemented from [GiNaC::basic](#).

References [options](#).

### 6.160.3.25 collect()

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::collect (
 const ex & s,
 bool distributed = false) const [inline], [override], [virtual]
```

Sort expanded expression in terms of powers of some object(s).

## Parameters

|                    |                                                            |
|--------------------|------------------------------------------------------------|
| <i>s</i>           | object(s) to sort in                                       |
| <i>distributed</i> | recursive or distributed form (only used when s is a list) |

Reimplemented from [GiNaC::basic](#).

**6.160.3.26 derivative()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::derivative (
 const symbol & s) const [inline], [override], [protected], [virtual]
```

Default implementation of [ex::diff\(\)](#).

It maps the operation on the operands (or returns 0 when the object has no operands).

## See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.27 series()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::series (
 const relational & r,
 int order,
 unsigned options = 0) const [inline], [override], [virtual]
```

Default implementation of [ex::series\(\)](#).

This performs Taylor expansion.

## See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [options](#), [order](#), and [r](#).

**6.160.3.28 normal()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [inline], [override], [virtual]
```

Default implementation of [ex::normal\(\)](#).

It normalizes the children and replaces the object with a temporary symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.29 to\_rational()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_rational (
 exmap & repl) const [inline], [override], [virtual]
```

Default implementation of [ex::to\\_rational\(\)](#).

This replaces the object with a temporary symbol.

Reimplemented from [GiNaC::basic](#).

**6.160.3.30 to\_polynomial()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::to_polynomial (
 exmap & repl) const [inline], [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.31 integer\_content()**

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::integer_content () const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

**6.160.3.32 smod()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::smod (
 const numeric & xi) const [inline], [override], [virtual]
```

Apply symmetric modular homomorphism to an expanded multivariate polynomial.

This function is usually used internally by [heur\\_gcd\(\)](#).

## Parameters

|      |         |
|------|---------|
| $xi$ | modulus |
|------|---------|

## Returns

mapped polynomial

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.33 max\_coefficient()**

```
template<class T , template< class > class ComparisonPolicy>
numeric GiNaC::structure< T, ComparisonPolicy >::max_coefficient () const [inline], [override],
[virtual]
```

Implementation [ex::max\\_coefficient\(\)](#).

## See also

[heur\\_gcd](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.34 get\_free\_indices()**

```
template<class T , template< class > class ComparisonPolicy>
exvector GiNaC::structure< T, ComparisonPolicy >::get_free_indices () const [inline], [override],
[virtual]
```

Return a vector containing the free indices of an expression.

Reimplemented from [GiNaC::basic](#).

**6.160.3.35 add\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::add_indexed (
 const ex & self,
 const ex & other) const [inline], [override], [virtual]
```

Add two indexed expressions.

They are guaranteed to be of class indexed (or a subclass) and their indices are compatible. This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

|              |                                                    |
|--------------|----------------------------------------------------|
| <i>self</i>  | First indexed expression; its base object is *this |
| <i>other</i> | Second indexed expression                          |

## Returns

sum of self and other

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

**6.160.3.36 scalar\_mul\_indexed()**

```
template<class T , template< class > class ComparisonPolicy>
ex GiNaC::structure< T, ComparisonPolicy >::scalar_mul_indexed (
 const ex & self,
 const numeric & other) const [inline], [override], [virtual]
```

Multiply an indexed expression with a scalar.

This function is used internally by [simplify\\_indexed\(\)](#).

## Parameters

|              |                                              |
|--------------|----------------------------------------------|
| <i>self</i>  | Indexed expression; its base object is *this |
| <i>other</i> | Numeric value                                |

## Returns

product of self and other

## See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ex](#).

**6.160.3.37 contract\_with()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::contract_with (
 exvector::iterator self,
```

```
exvector::iterator other,
exvector & v) const [inline], [override], [virtual]
```

Try to contract two indexed expressions that appear in the same product.

If a contraction exists, the function overwrites one or both of the expressions and returns true. Otherwise it returns false. It is guaranteed that both expressions are of class indexed (or a subclass) and that at least one dummy index has been found. This functions is used internally by [simplify\\_indexed\(\)](#).

#### Parameters

|              |                                                               |
|--------------|---------------------------------------------------------------|
| <i>self</i>  | Pointer to first indexed expression; its base object is *this |
| <i>other</i> | Pointer to second indexed expression                          |
| <i>v</i>     | The complete vector of factors                                |

#### Returns

true if the contraction was successful, false otherwise

#### See also

[ex::simplify\\_indexed\(\)](#)

Reimplemented from [GiNaC::basic](#).

#### 6.160.3.38 return\_type()

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::return_type () const [inline], [override],
[virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

#### 6.160.3.39 return\_type\_tinfo()

```
template<class T , template< class > class ComparisonPolicy>
return_type_t GiNaC::structure< T, ComparisonPolicy >::return_type_tinfo () const [inline],
[override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [r](#), and [GiNaC::return\\_type\\_t::rl](#).

**6.160.3.40 is\_equal\_same\_type()**

```
template<class T , template< class > class ComparisonPolicy>
bool GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type (
 const basic & other) const [inline], [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.41 calchash()**

```
template<class T , template< class > class ComparisonPolicy>
unsigned GiNaC::structure< T, ComparisonPolicy >::calchash () const [inline], [override],
[protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

**6.160.3.42 operator->()**

```
template<class T , template< class > class ComparisonPolicy>
const T * GiNaC::structure< T, ComparisonPolicy >::operator-> () const [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.43 get\_struct() [1/2]**

```
template<class T , template< class > class ComparisonPolicy>
T & GiNaC::structure< T, ComparisonPolicy >::get_struct () [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

**6.160.3.44 get\_struct() [2/2]**

```
template<class T , template< class > class ComparisonPolicy>
const T & GiNaC::structure< T, ComparisonPolicy >::get_struct () const [inline]
```

References [GiNaC::structure< T, ComparisonPolicy >::obj](#).

## 6.160.4 Member Data Documentation

### 6.160.4.1 obj

```
template<class T , template< class > class ComparisonPolicy>
T GiNaC::structure< T, ComparisonPolicy >::obj [private]
```

Referenced by [GiNaC::structure< T, ComparisonPolicy >::get\\_struct\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::get\\_struct\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::is\\_equal\\_same\\_type\(\)](#), and [GiNaC::structure< T, ComparisonPolicy >::operator->\(\)](#).

The documentation for this class was generated from the following file:

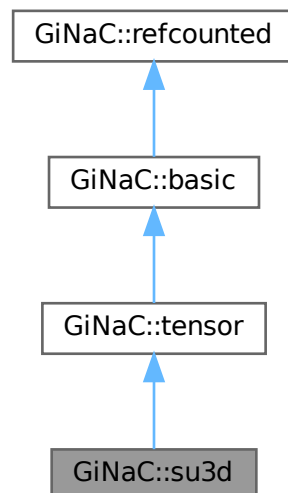
- [structure.h](#)

## 6.161 GiNaC::su3d Class Reference

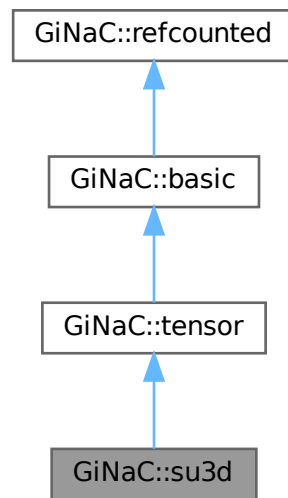
This class represents the tensor of symmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3d:



Collaboration diagram for GiNaC::su3d:



### Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed symmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed symmetric structure constant with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*

- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual void `print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual void `dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence () const`  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf) const`  
*Information about the object.*
- virtual size\_t `nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0) const`  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v) const`
- virtual bool `is_polynomial (const ex &var) const`  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0) const`  
*Default implementation of ex::series().*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier) const`  
*Default implementation of ex::normal().*
- virtual `ex to_rational (exmap &repl) const`  
*Default implementation of ex::to\_rational().*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const
- void [do\\_print\\_latex](#) (const [print\\_latex](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.161.1 Detailed Description

This class represents the tensor of symmetric su(3) structure constants.

## 6.161.2 Member Function Documentation

### 6.161.2.1 eval\_indexed()

```
ex GiNaC::su3d::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of indexed symmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex1\\_3](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [GiNaC::\\_ex\\_1\\_3](#), [GiNaC::\\_ex\\_6](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 6.161.2.2 contract\_with()

```
bool GiNaC::su3d::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed symmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 6.161.2.3 return\_type()

```
unsigned GiNaC::su3d::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.161.2.4 do\_print()

```
void GiNaC::su3d::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.161.2.5 do\_print\_latex()

```
void GiNaC::su3d::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

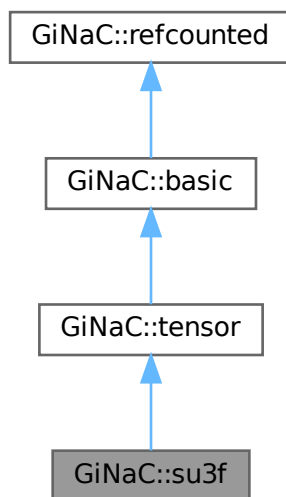
- [color.h](#)
- [color.cpp](#)

## 6.162 GiNaC::su3f Class Reference

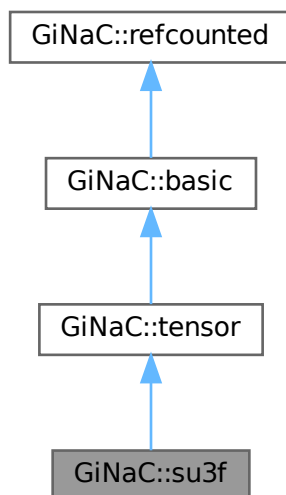
This class represents the tensor of antisymmetric su(3) structure constants.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3f:



Collaboration diagram for GiNaC::su3f:



## Public Member Functions

- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of indexed antisymmetric structure constant.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed antisymmetric structure constant with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- `virtual ~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- `const basic & operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- `virtual basic * duplicate` () const  
*Create a clone of this object on the heap.*
- `virtual ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- `virtual ex evalf` () const  
*Evaluate object numerically.*
- `virtual ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- `virtual ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- `virtual void print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- `virtual void dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- `virtual void dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- `virtual unsigned precedence` () const  
*Return relative operator precedence (for parenthesizing output).*
- `virtual bool info` (unsigned inf) const  
*Information about the object.*
- `virtual size_t nops` () const  
*Number of operands/members.*
- `virtual ex op` (size\_t i) const  
*Return operand/member at position i.*
- `virtual ex operator[]` (const `ex` &index) const
- `virtual ex operator[]` (size\_t i) const
- `virtual ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- `virtual ex & operator[]` (const `ex` &index)
- `virtual ex & operator[]` (size\_t i)

- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int [n](#)=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const
- *Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const
- *Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const
- *Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const
- *Test for syntactic equality.*
- `const basic & hold` () const
- *Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const
- *Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const
- *Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `unsigned return_type` () const override
- `void do_print` (const `print_context` &`c`, unsigned `level`) const
- `void do_print_latex` (const `print_latex` &`c`, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::tensor`

- `unsigned return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &`v`) const
- `virtual bool match_same_type` (const `basic` &`other`) const
- *Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &`s`) const
- *Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &`other`) const
- *Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &`other`) const
- *Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const
- *Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.162.1 Detailed Description

This class represents the tensor of antisymmetric su(3) structure constants.

## 6.162.2 Member Function Documentation

### 6.162.2.1 [eval\\_indexed\(\)](#)

```
ex GiNaC::su3f::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of indexed antisymmetric structure constant.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1\\_2](#), [GiNaC::\\_ex3](#), [GiNaC::\\_ex\\_1\\_2](#), [CMPINDICES](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::sqrt\(\)](#), and [std::swap\(\)](#).

### 6.162.2.2 [contract\\_with\(\)](#)

```
bool GiNaC::su3f::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed antisymmetric structure constant with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_T\(\)](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::get\\_representation\\_label\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::l](#), [GiNaC::ex::op\(\)](#), and [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#).

### 6.162.2.3 return\_type()

```
unsigned GiNaC::su3f::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.162.2.4 do\_print()

```
void GiNaC::su3f::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.162.2.5 do\_print\_latex()

```
void GiNaC::su3f::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

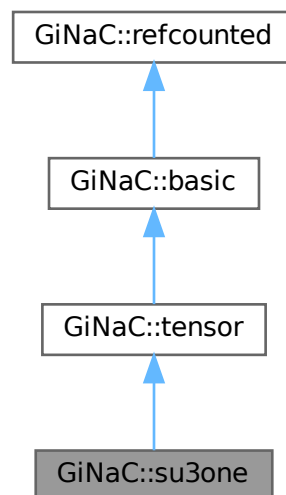
- [color.h](#)
- [color.cpp](#)

## 6.163 GiNaC::su3one Class Reference

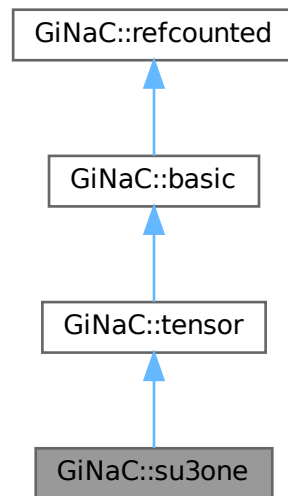
This class represents the su(3) unity element.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3one:



Collaboration diagram for `GiNaC::su3one`:



#### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

#### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

#### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

### Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around print to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const

- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int `n`=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool `distributed`=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const

*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const

*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const

*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const

*Try to contract two indexed expressions that appear in the same product.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >
  - void `print_dispatch` (const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const

*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const

*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

*Load (deserialize) the object from an archive node.*

- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned f) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int r) noexcept

## Protected Attributes inherited from `GiNaC::basic`

- `unsigned flags`  
*of type `status_flags`*
- `unsigned hashvalue`  
*hash value*

### 6.163.1 Detailed Description

This class represents the  $su(3)$  unity element.

### 6.163.2 Member Function Documentation

#### 6.163.2.1 `do_print()`

```
void GiNaC::su3one::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.163.2.2 do\_print\_latex()

```
void GiNaC::su3one::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following file:

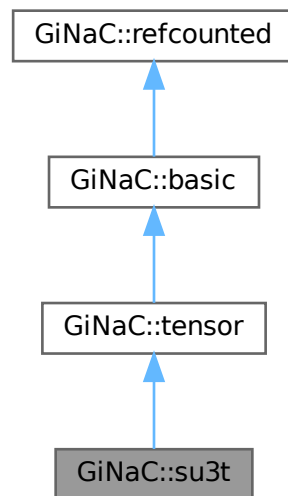
- [color.h](#)

## 6.164 GiNaC::su3t Class Reference

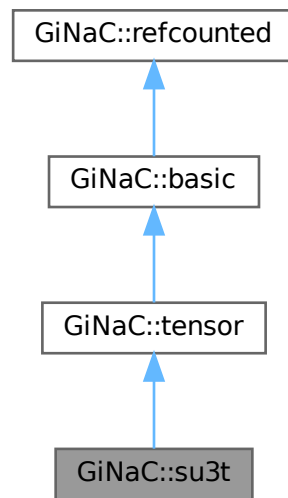
This class represents an su(3) generator.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3t:



Collaboration diagram for GiNaC::su3t:



### Public Member Functions

- bool [contract\\_with](#) (exvector::iterator self, exvector::iterator other, [exvector](#) &v) const override  
*Contraction of generator with something else.*

### Public Member Functions inherited from [GiNaC::tensor](#)

- bool [replace\\_contr\\_index](#) (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*

- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i)` const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree ()` const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info (unsigned inf)` const  
*Information about the object.*
- virtual size\_t `nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex &let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex &operator[] (const ex &index)`
- virtual `ex &operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

### Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

### Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `ensure_if_modifiable` () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void `do_print` (const `print_context` &c, unsigned level) const  
*Default output to stream.*
- void `do_print_tree` (const `print_tree` &c, unsigned level) const  
*Tree output to stream.*
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from `GiNaC::basic`

- unsigned `flags`  
*of type `status_flags`*
- unsigned `hashvalue`  
*hash value*

## 6.164.1 Detailed Description

This class represents an  $su(3)$  generator.

## 6.164.2 Member Function Documentation

### 6.164.2.1 contract\_with()

```
bool GiNaC::su3t::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of generator with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::color\\_ONE\(\)](#), [GiNaC::color\\_trace\(\)](#), and [GINAC\\_ASSERT](#).

### 6.164.2.2 do\_print()

```
void GiNaC::su3t::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.164.2.3 do\_print\_latex()

```
void GiNaC::su3t::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

- [color.h](#)
- [color.cpp](#)

## 6.165 GiNaC::subs\_options Class Reference

Flags to control the behavior of [subs\(\)](#).

```
#include <flags.h>
```

### Public Types

- enum {  
[no\\_pattern](#) = 0x0001 , [subs\\_no\\_pattern](#) = 0x0001 , [algebraic](#) = 0x0002 , [subs\\_algebraic](#) = 0x0002 ,  
[pattern\\_is\\_product](#) = 0x0004 , [pattern\\_is\\_not\\_product](#) = 0x0008 , [no\\_index\\_renaming](#) = 0x0010 ,  
[really\\_subs\\_idx](#) = 0x0020 }

### 6.165.1 Detailed Description

Flags to control the behavior of [subs\(\)](#).

## 6.165.2 Member Enumeration Documentation

### 6.165.2.1 anonymous enum

```
anonymous enum
```

## Enumerator

|                        |                                                              |
|------------------------|--------------------------------------------------------------|
| no_pattern             | disable pattern matching                                     |
| subs_no_pattern        |                                                              |
| algebraic              | enable algebraic substitutions                               |
| subs_algebraic         |                                                              |
| pattern_is_product     | used internally by <a href="#">expairseq::subschildren()</a> |
| pattern_is_not_product | used internally by <a href="#">expairseq::subschildren()</a> |
| no_index_renaming      |                                                              |
| really_subs_idx        |                                                              |

The documentation for this class was generated from the following file:

- [flags.h](#)

## 6.166 GiNaC::sy\_is\_less Class Reference

### Public Member Functions

- [sy\\_is\\_less](#) (exvector::iterator v\_)
- bool [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh) const

### Private Attributes

- exvector::iterator [v](#)

### 6.166.1 Constructor & Destructor Documentation

#### 6.166.1.1 sy\_is\_less()

```
GiNaC::sy_is_less::sy_is_less (
 exvector::iterator v_) [inline]
```

### 6.166.2 Member Function Documentation

#### 6.166.2.1 operator>()()

```
bool GiNaC::sy_is_less::operator() (
 const ex & lh,
 const ex & rh) const [inline]
```

References [GINAC\\_ASSERT](#), and [v](#).

## 6.166.3 Member Data Documentation

### 6.166.3.1 v

`exvector::iterator GiNaC::sy_is_less::v` [private]

Referenced by [operator\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.167 GiNaC::sy\_swap Class Reference

### Public Member Functions

- [sy\\_swap](#) (exvector::iterator v\_, bool &s)
- void [operator\(\)](#) (const [ex](#) &lh, const [ex](#) &rh)

### Public Attributes

- bool & [swapped](#)

### Private Attributes

- exvector::iterator [v](#)

## 6.167.1 Constructor & Destructor Documentation

### 6.167.1.1 sy\_swap()

```
GiNaC::sy_swap::sy_swap (
 exvector::iterator v_,
 bool & s) [inline]
```

## 6.167.2 Member Function Documentation

### 6.167.2.1 operator()()

```
void GiNaC::sy_swap::operator() (
 const ex & lh,
 const ex & rh) [inline]
```

References [GINAC\\_ASSERT](#), [swapped](#), and [v](#).

### 6.167.3 Member Data Documentation

#### 6.167.3.1 v

```
exvector::iterator GiNaC::sy_swap::v [private]
```

Referenced by [operator\(\)\(\)](#).

#### 6.167.3.2 swapped

```
bool& GiNaC::sy_swap::swapped
```

Referenced by [operator\(\)\(\)](#).

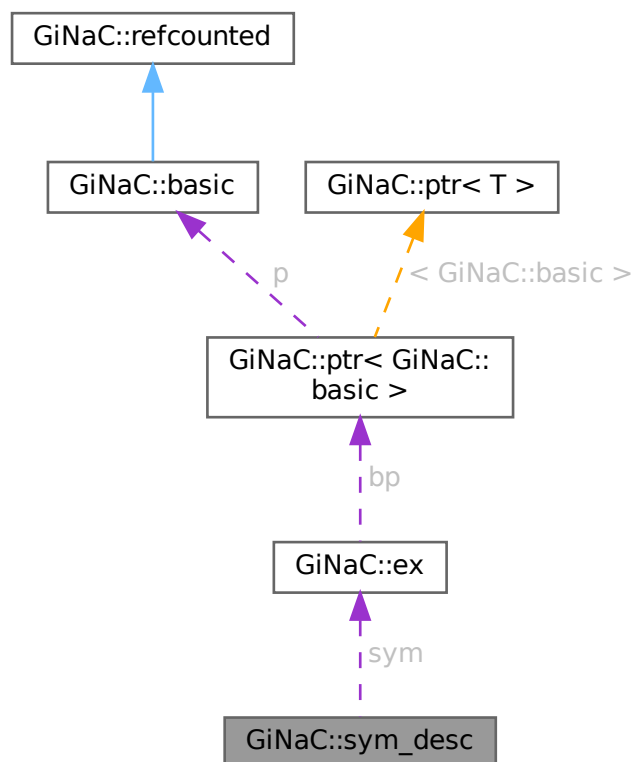
The documentation for this class was generated from the following file:

- [symmetry.cpp](#)

## 6.168 GiNaC::sym\_desc Struct Reference

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

Collaboration diagram for GiNaC::sym\_desc:



## Public Member Functions

- [sym\\_desc](#) (const [ex](#) &s)  
*Initialize symbol, leave other variables uninitialized.*
- bool [operator<](#) (const [sym\\_desc](#) &x) const  
*Comparison operator for sorting.*

## Public Attributes

- [ex](#) [sym](#)  
*Reference to symbol.*
- int [deg\\_a](#)  
*Highest degree of symbol in polynomial "a".*
- int [deg\\_b](#)  
*Highest degree of symbol in polynomial "b".*
- int [ldeg\\_a](#)  
*Lowest degree of symbol in polynomial "a".*
- int [ldeg\\_b](#)  
*Lowest degree of symbol in polynomial "b".*
- int [max\\_deg](#)  
*Maximum of deg\_a and deg\_b (Used for sorting)*
- size\_t [max\\_lcnops](#)  
*Maximum number of terms of leading coefficient of symbol in both polynomials.*

### 6.168.1 Detailed Description

This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".

A vector of these structures with information about all symbols in two polynomials can be created with the function [get\\_symbol\\_stats\(\)](#).

See also

[get\\_symbol\\_stats](#)

### 6.168.2 Constructor & Destructor Documentation

#### 6.168.2.1 sym\_desc()

```
GiNaC::sym_desc::sym_desc (
 const ex & s) [inline]
```

Initialize symbol, leave other variables uninitialized.

### 6.168.3 Member Function Documentation

#### 6.168.3.1 `operator<()`

```
bool GiNaC::sym_desc::operator< (
 const sym_desc & x) const [inline]
```

Comparison operator for sorting.

References [max\\_deg](#), [max\\_lcnops](#), and [x](#).

### 6.168.4 Member Data Documentation

#### 6.168.4.1 `sym`

```
ex GiNaC::sym_desc::sym
```

Reference to symbol.

#### 6.168.4.2 `deg_a`

```
int GiNaC::sym_desc::deg_a
```

Highest degree of symbol in polynomial "a".

#### 6.168.4.3 `deg_b`

```
int GiNaC::sym_desc::deg_b
```

Highest degree of symbol in polynomial "b".

#### 6.168.4.4 `ldeg_a`

```
int GiNaC::sym_desc::ldeg_a
```

Lowest degree of symbol in polynomial "a".

#### 6.168.4.5 `ldeg_b`

```
int GiNaC::sym_desc::ldeg_b
```

Lowest degree of symbol in polynomial "b".

#### 6.168.4.6 max\_deg

```
int GiNaC::sym_desc::max_deg
```

Maximum of deg\_a and deg\_b (Used for sorting)

Referenced by [operator<\(\)](#).

#### 6.168.4.7 max\_lcnops

```
size_t GiNaC::sym_desc::max_lcnops
```

Maximum number of terms of leading coefficient of symbol in both polynomials.

Referenced by [operator<\(\)](#).

The documentation for this struct was generated from the following file:

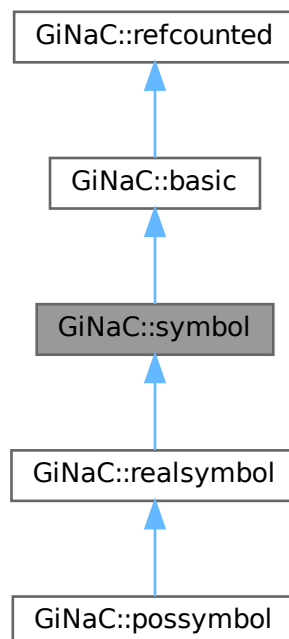
- [normal.cpp](#)

## 6.169 GiNaC::symbol Class Reference

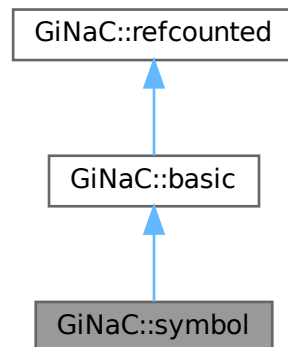
Basic CAS symbol.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::symbol:



Collaboration diagram for GiNaC::symbol:



### Public Member Functions

- [symbol](#) (const std::string &initname)
- [symbol](#) (const std::string &initname, const std::string &texname)
- bool [info](#) (unsigned inf) const override  
*Information about the object.*
- [ex eval](#) () const override  
*Perform automatic non-interruptive term rewriting rules.*
- [ex evalf](#) () const override  
*Evaluate object numerically.*
- [ex series](#) (const [relational](#) &s, int [order](#), unsigned [options](#)=0) const override  
*Implementation of [ex::series\(\)](#) for symbols.*
- [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const override  
*Implementation of [ex::normal\(\)](#) for symbols.*
- [ex to\\_rational](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_rational\(\)](#) for symbols.*
- [ex to\\_polynomial](#) ([exmap](#) &repl) const override  
*Implementation of [ex::to\\_polynomial\(\)](#) for symbols.*
- [ex conjugate](#) () const override
- [ex real\\_part](#) () const override
- [ex imag\\_part](#) () const override
- bool [is\\_polynomial](#) (const [ex](#) &var) const override  
*Check whether this is a polynomial in the given variables.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms) override  
*Read (a.k.a.*
- virtual unsigned [get\\_domain](#) () const
- void [set\\_name](#) (const std::string &n)
- void [set\\_TeX\\_name](#) (const std::string &n)
- std::string [get\\_name](#) () const
- std::string [get\\_TeX\\_name](#) () const

Public Member Functions inherited from **GiNaC::basic**

- virtual `~basic ()`  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic (const basic &other)`
- `const basic & operator= (const basic &other)`  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate () const`  
*Create a clone of this object on the heap.*
- virtual `ex evalm () const`  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ () const`  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed (const basic &i) const`  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual `void print (const print_context &c, unsigned level=0) const`  
*Output to stream.*
- virtual `void dbgprint () const`  
*Little wrapper around print to be called within a debugger.*
- virtual `void dbgprinttree () const`  
*Little wrapper around printtree to be called within a debugger.*
- virtual `unsigned precedence () const`  
*Return relative operator precedence (for parenthesizing output).*
- virtual `size_t nops () const`  
*Number of operands/members.*
- virtual `ex op (size_t i) const`  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index) const`
- virtual `ex operator[] (size_t i) const`
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual `bool has (const ex &other, unsigned options=0) const`  
*Test for occurrence of a pattern.*
- virtual `bool match (const ex &pattern, exmap &repls) const`  
*Check whether the expression matches a given pattern.*
- virtual `ex map (map_function &f) const`  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual `void accept (GiNaC::visitor &v) const`
- virtual `int degree (const ex &s) const`  
*Return degree of highest power in object s.*
- virtual `int ldegree (const ex &s) const`  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1) const`  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0) const`  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false) const`  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `numeric integer_content () const`

- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual unsigned `return_type` () const
- virtual `return_type_t return_type_tinfo` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for single differentiation of a symbol.*
- bool `is_equal_same_type` (const `basic` &other) const override  
*Returns true if two objects of same type are equal.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const
- void `do_print_python_repr` (const `print_python_repr` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- unsigned [serial](#)  
*unique serial number for comparison*
- std::string [name](#)  
*printname of this symbol*
- std::string [TeX\\_name](#)  
*LaTeX name of this symbol.*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## Static Private Attributes

- static unsigned [next\\_serial](#) = 0

### 6.169.1 Detailed Description

Basic CAS symbol.

It has a name because it must know how to output itself.

### 6.169.2 Constructor & Destructor Documentation

#### 6.169.2.1 [symbol\(\)](#) [1/2]

```
GiNaC::symbol::symbol (
 const std::string & initname) [explicit]
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

### 6.169.2.2 `symbol()` [2/2]

```
GiNaC::symbol::symbol (
 const std::string & initname,
 const std::string & texname)
```

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.169.3 Member Function Documentation

### 6.169.3.1 `info()`

```
bool GiNaC::symbol::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::cinteger\\_polynomial](#), [GiNaC::info\\_flags::crational\\_polynomial](#), [GiNaC::info\\_flags::expanded](#), [get\\_domain\(\)](#), [GiNaC::info\\_flags::has\\_indices](#), [GiNaC::info\\_flags::integer\\_polynomial](#), [GiNaC::info\\_flags::nonnegative](#), [GiNaC::info\\_flags::polynomial](#), [GiNaC::domain::positive](#), [GiNaC::info\\_flags::positive](#), [GiNaC::info\\_flags::rational\\_function](#), [GiNaC::info\\_flags::rational\\_polynomial](#), [GiNaC::domain::real](#), [GiNaC::info\\_flags::real](#), and [GiNaC::info\\_flags::symbol](#).

Referenced by [GiNaC::conjugate\\_expl\\_derivative\(\)](#), [GiNaC::imag\\_part\\_expl\\_derivative\(\)](#), and [GiNaC::real\\_part\\_expl\\_derivative\(\)](#).

### 6.169.3.2 `eval()`

```
ex GiNaC::symbol::eval () const [inline], [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.3 `evalf()`

```
ex GiNaC::symbol::evalf () const [inline], [override], [virtual]
```

Evaluate object numerically.

Reimplemented from [GiNaC::basic](#).

#### 6.169.3.4 series()

```
ex GiNaC::symbol::series (
 const relational & r,
 int order,
 unsigned options = 0) const [override], [virtual]
```

Implementation of [ex::series\(\)](#) for symbols.

See also

[ex::series](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GINAC\\_ASSERT](#), [is\\_equal\\_same\\_type\(\)](#), [GiNaC::ex::is\\_zero\(\)](#), [order](#), [r](#), and [GiNaC::ex::rhs\(\)](#).

Referenced by [GiNaC::EllipticE\\_series\(\)](#), and [GiNaC::EllipticK\\_series\(\)](#).

#### 6.169.3.5 subs()

```
ex GiNaC::symbol::subs (
 const exmap & m,
 unsigned options = 0) const [inline], [override], [virtual]
```

Substitute a set of objects by arbitrary expressions.

The ex returned will already be evaluated.

Reimplemented from [GiNaC::basic](#).

References [m](#), [options](#), and [GiNaC::basic::subs\\_one\\_level\(\)](#).

#### 6.169.3.6 normal()

```
ex GiNaC::symbol::normal (
 exmap & repl,
 exmap & rev_lookup,
 lst & modifier) const [override], [virtual]
```

Implementation of [ex::normal\(\)](#) for symbols.

This returns the unmodified symbol.

See also

[ex::normal](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#).

### 6.169.3.7 to\_rational()

```
ex GiNaC::symbol::to_rational (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_rational\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.8 to\_polynomial()

```
ex GiNaC::symbol::to_polynomial (
 exmap & repl) const [override], [virtual]
```

Implementation of [ex::to\\_polynomial\(\)](#) for symbols.

This returns the unmodified symbol.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.9 conjugate()

```
ex GiNaC::symbol::conjugate () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.169.3.10 real\_part()

```
ex GiNaC::symbol::real_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.169.3.11 imag\_part()

```
ex GiNaC::symbol::imag_part () const [override], [virtual]
```

Reimplemented from [GiNaC::basic](#).

### 6.169.3.12 is\_polynomial()

```
bool GiNaC::symbol::is_polynomial (
 const ex & var) const [override], [virtual]
```

Check whether this is a polynomial in the given variables.

Reimplemented from [GiNaC::basic](#).

### 6.169.3.13 archive()

```
void GiNaC::symbol::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [n](#), [name](#), and [TeX\\_name](#).

### 6.169.3.14 read\_archive()

```
void GiNaC::symbol::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

Read object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::container< C >::append\(\)](#), [GiNaC::status\\_flags::dynallocated](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [n](#), [name](#), [next\\_serial](#), [serial](#), [GiNaC::basic::setflag\(\)](#), and [TeX\\_name](#).

### 6.169.3.15 derivative()

```
ex GiNaC::symbol::derivative (
 const symbol & s) const [override], [protected], [virtual]
```

Implementation of [ex::diff\(\)](#) for single differentiation of a symbol.

It returns 1 or 0.

See also

[ex::diff](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), and [GiNaC::basic::compare\\_same\\_type\(\)](#).

**6.169.3.16 is\_equal\_same\_type()**

```
bool GiNaC::symbol::is_equal_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if two objects of same type are equal.

Normally needs not be reimplemented as long as it wasn't overwritten by some parent class, since it just calls [compare\\_same\\_type\(\)](#). The reason why this function exists is that sometimes it is easier to determine equality than an order relation and then it can be overridden.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [serial](#).

Referenced by [series\(\)](#).

**6.169.3.17 calchash()**

```
unsigned GiNaC::symbol::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [GiNaC::make\\_hash\\_seed\(\)](#), [serial](#), and [GiNaC::basic::setflag\(\)](#).

**6.169.3.18 get\_domain()**

```
virtual unsigned GiNaC::symbol::get_domain () const [inline], [virtual]
```

Reimplemented in [GiNaC::realsymbol](#), and [GiNaC::possymbol](#).

References [GiNaC::domain::complex](#).

Referenced by [do\\_print\\_tree\(\)](#), and [info\(\)](#).

**6.169.3.19 set\_name()**

```
void GiNaC::symbol::set_name (
 const std::string & n) [inline]
```

References [n](#), and [name](#).

### 6.169.3.20 set\_TeX\_name()

```
void GiNaC::symbol::set_TeX_name (
 const std::string & n) [inline]
```

References [n](#), and [TeX\\_name](#).

### 6.169.3.21 get\_name()

```
std::string GiNaC::symbol::get_name () const
```

References [name](#), and [serial](#).

Referenced by [do\\_print\(\)](#), and [get\\_TeX\\_name\(\)](#).

### 6.169.3.22 get\_TeX\_name()

```
std::string GiNaC::symbol::get_TeX_name () const
```

References [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [get\\_name\(\)](#), and [TeX\\_name](#).

### 6.169.3.23 do\_print()

```
void GiNaC::symbol::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [get\\_name\(\)](#).

### 6.169.3.24 do\_print\_latex()

```
void GiNaC::symbol::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::get\\_default\\_TeX\\_name\(\)](#), [name](#), [serial](#), and [TeX\\_name](#).

### 6.169.3.25 do\_print\_tree()

```
void GiNaC::symbol::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [get\\_domain\(\)](#), [GiNaC::basic::hashvalue](#), [name](#), and [serial](#).

### 6.169.3.26 do\_print\_python\_repr()

```
void GiNaC::symbol::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), [name](#), [serial](#), and [TeX\\_name](#).

## 6.169.4 Member Data Documentation

### 6.169.4.1 serial

```
unsigned GiNaC::symbol::serial [protected]
```

unique serial number for comparison

Referenced by [calchash\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [is\\_equal\\_same\\_type\(\)](#), and [read\\_archive\(\)](#).

### 6.169.4.2 name

```
std::string GiNaC::symbol::name [mutable], [protected]
```

printname of this symbol

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_name\(\)](#), [read\\_archive\(\)](#), and [set\\_name\(\)](#).

### 6.169.4.3 TeX\_name

```
std::string GiNaC::symbol::TeX_name [protected]
```

LaTeX name of this symbol.

Referenced by [archive\(\)](#), [do\\_print\\_latex\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [get\\_TeX\\_name\(\)](#), [read\\_archive\(\)](#), and [set\\_TeX\\_name\(\)](#).

### 6.169.4.4 next\_serial

```
unsigned GiNaC::symbol::next_serial = 0 [static], [private]
```

Referenced by [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

- [symbol.h](#)
- [normal.cpp](#)
- [pseries.cpp](#)
- [symbol.cpp](#)

## 6.170 GiNaC::symbolset Class Reference

### Public Member Functions

- [symbolset](#) (const [ex](#) &e)
- bool [has](#) (const [ex](#) &e) const

### Private Member Functions

- void [insert\\_symbols](#) (const [ex](#) &e)

### Private Attributes

- [exset](#) [s](#)

## 6.170.1 Constructor & Destructor Documentation

### 6.170.1.1 symbolset()

```
GiNaC::symbolset::symbolset (
 const ex & e) [inline], [explicit]
```

References [insert\\_symbols\(\)](#).

## 6.170.2 Member Function Documentation

### 6.170.2.1 insert\_symbols()

```
void GiNaC::symbolset::insert_symbols (
 const ex & e) [inline], [private]
```

References [insert\\_symbols\(\)](#), and [s](#).

Referenced by [insert\\_symbols\(\)](#), and [symbolset\(\)](#).

### 6.170.2.2 has()

```
bool GiNaC::symbolset::has (
 const ex & e) const [inline]
```

References [s](#).

Referenced by [GiNaC::lsolve\(\)](#).

### 6.170.3 Member Data Documentation

#### 6.170.3.1 s

```
exset GiNaC::symbolset::s [private]
```

Referenced by [has\(\)](#), and [insert\\_symbols\(\)](#).

The documentation for this class was generated from the following file:

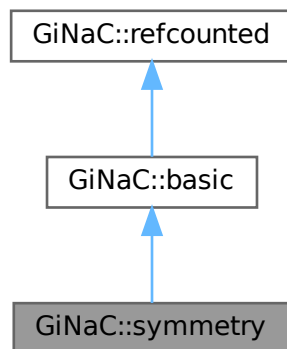
- [inifcns.cpp](#)

## 6.171 GiNaC::symmetry Class Reference

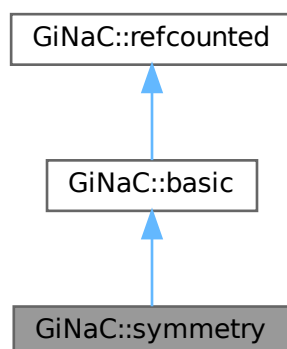
This class describes the symmetry of a group of indices.

```
#include <symmetry.h>
```

Inheritance diagram for GiNaC::symmetry:



Collaboration diagram for GiNaC::symmetry:



## Public Types

- enum [symmetry\\_type](#) { [none](#) , [symmetric](#) , [antisymmetric](#) , [cyclic](#) }
- Type of symmetry.*

## Public Member Functions

- [symmetry](#) (unsigned i)  
*Create leaf node that represents one index.*
- [symmetry](#) ([symmetry\\_type](#) t, const [symmetry](#) &c1, const [symmetry](#) &c2)  
*Create node with two children.*
- void [archive](#) ([archive\\_node](#) &n) const override  
*Save (a.k.a.*
- void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &[syms](#)) override  
*Read (a.k.a.*
- [symmetry\\_type](#) [get\\_type](#) () const  
*Get symmetry type.*
- void [set\\_type](#) ([symmetry\\_type](#) t)  
*Set symmetry type.*
- [symmetry](#) & [add](#) (const [symmetry](#) &c)  
*Add child node, check index sets for consistency.*
- void [validate](#) (unsigned n)  
*Verify that all indices of this node are in the range [0..n-1].*
- bool [has\\_symmetry](#) () const  
*Check whether this node actually represents any kind of symmetry.*
- bool [has\\_nonsymmetric](#) () const  
*Check whether this node involves anything non symmetric.*
- bool [has\\_cyclic](#) () const  
*Check whether this node involves a cyclic symmetry.*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex](#) [eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex](#) [evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex](#) [evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex](#) [eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex](#) [eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const

- Output to stream.*

  - virtual void `dbgprint` () const

*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const

*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const

*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const

*Information about the object.*
- virtual size\_t `nops` () const

*Number of operands/members.*
- virtual `ex op` (size\_t i) const

*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)

*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const

*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const

*Check whether the expression matches a given pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const

*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const

*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const

*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const

*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const

*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const

*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const

*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const

*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const

*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const

*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const

*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const

*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const

- Implementation `ex::max_coefficient()`.
- virtual `exvector get_free_indices ()` const  
Return a vector containing the free indices of an expression.
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
Add two indexed expressions.
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
Multiply an indexed expression with a scalar.
- virtual bool `contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
Try to contract two indexed expressions that appear in the same product.
- virtual unsigned `return_type ()` const
- virtual `return_type_t return_type_info ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch (const print_context &c, unsigned level)` const  
Like `print()`, but dispatch to the specified class.
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
Like `print()`, but dispatch to the specified class.
- `ex subs_one_level (const exmap &m, unsigned options)` const  
Helper function for `subs()`.
- `ex diff (const symbol &s, unsigned nth=1)` const  
Default interface of `nth` derivative `ex::diff(s, n)`.
- int `compare (const basic &other)` const  
Compare objects syntactically to establish canonical ordering.
- bool `is_equal (const basic &other)` const  
Test for syntactic equality.
- const `basic & hold ()` const  
Stop further evaluation.
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
Set some `status_flags`.
- const `basic & clearflag (unsigned f)` const  
Clear some `status_flags`.

## Public Member Functions inherited from GiNaC::refcounted

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

## Protected Member Functions

- unsigned `calchash ()` const override  
Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.
- void `do_print (const print_context &c, unsigned level)` const
- void `do_print_tree (const print_tree &c, unsigned level)` const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- [symmetry\\_type](#) type  
*Type of symmetry described by this node.*
- std::set< unsigned > [indices](#)  
*Sorted union set of all indices handled by this node.*
- [exvector children](#)  
*Vector of child nodes.*

## Friends

- class [sy\\_is\\_less](#)
- class [sy\\_swap](#)
- int [canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.171.1 Detailed Description

This class describes the symmetry of a group of indices.

These objects can be grouped into a tree to form complex mixed symmetries.

## 6.171.2 Member Enumeration Documentation

### 6.171.2.1 symmetry\_type

enum [GiNaC::symmetry::symmetry\\_type](#)

Type of symmetry.

Enumerator

|               |                        |
|---------------|------------------------|
| none          | no symmetry properties |
| symmetric     | totally symmetric      |
| antisymmetric | totally antisymmetric  |
| cyclic        | cyclic symmetry        |

## 6.171.3 Constructor & Destructor Documentation

### 6.171.3.1 symmetry() [1/2]

```
GiNaC::symmetry::symmetry (
 unsigned i)
```

Create leaf node that represents one index.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [indices](#), and [GiNaC::basic::setflag\(\)](#).

### 6.171.3.2 symmetry() [2/2]

```
GiNaC::symmetry::symmetry (
 symmetry_type t,
 const symmetry & c1,
 const symmetry & c2)
```

Create node with two children.

References [add\(\)](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.171.4 Member Function Documentation

### 6.171.4.1 archive()

```
void GiNaC::symmetry::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

Archive the object.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [children](#), [indices](#), [n](#), and [type](#).

### 6.171.4.2 read\_archive()

```
void GiNaC::symmetry::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

Construct object from [archive\\_node](#).

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [add\(\)](#), [indices](#), [n](#), and [type](#).

### 6.171.4.3 calchash()

```
unsigned GiNaC::symmetry::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [children](#), [GiNaC::status\\_flags::evaluated](#), [GiNaC::basic::flags](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [indices](#), [GiNaC::make\\_hash\\_seed\(\)](#), [none](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::basic::setflag\(\)](#), and [type](#).

#### 6.171.4.4 get\_type()

```
symmetry_type GiNaC::symmetry::get_type () const [inline]
```

Get symmetry type.

References [type](#).

#### 6.171.4.5 set\_type()

```
void GiNaC::symmetry::set_type (
 symmetry_type t) [inline]
```

Set symmetry type.

References [type](#).

Referenced by [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), and [GiNaC::sy\\_symm\(\)](#).

#### 6.171.4.6 add()

```
symmetry & GiNaC::symmetry::add (
 const symmetry & c)
```

Add child node, check index sets for consistency.

References [c](#), [children](#), [GINAC\\_ASSERT](#), [indices](#), [none](#), and [type](#).

Referenced by [read\\_archive\(\)](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_anti\(\)](#), [GiNaC::sy\\_cycl\(\)](#), [GiNaC::sy\\_cycl\(\)](#), [GiNaC::sy\\_none\(\)](#), [GiNaC::sy\\_none\(\)](#), [GiNaC::sy\\_symm\(\)](#), [GiNaC::sy\\_symm\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).

#### 6.171.4.7 validate()

```
void GiNaC::symmetry::validate (
 unsigned n)
```

Verify that all indices of this node are in the range [0..n-1].

This function throws an exception if the verification fails. If the top node has a type != none and no children, add all indices in the range [0..n-1] as children.

References [add\(\)](#), [indices](#), [n](#), [none](#), and [type](#).

#### 6.171.4.8 has\_symmetry()

```
bool GiNaC::symmetry::has_symmetry () const [inline]
```

Check whether this node actually represents any kind of symmetry.

References [children](#), [none](#), and [type](#).

#### 6.171.4.9 has\_nonsymmetric()

```
bool GiNaC::symmetry::has_nonsymmetric () const
```

Check whether this node involves anything non symmetric.

References [antisymmetric](#), [children](#), [cyclic](#), and [type](#).

#### 6.171.4.10 has\_cyclic()

```
bool GiNaC::symmetry::has_cyclic () const
```

Check whether this node involves a cyclic symmetry.

References [children](#), [cyclic](#), and [type](#).

#### 6.171.4.11 do\_print()

```
void GiNaC::symmetry::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [antisymmetric](#), [c](#), [children](#), [cyclic](#), [indices](#), [none](#), [symmetric](#), and [type](#).

#### 6.171.4.12 do\_print\_tree()

```
void GiNaC::symmetry::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [antisymmetric](#), [c](#), [children](#), [cyclic](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [indices](#), [none](#), [symmetric](#), and [type](#).

### 6.171.5 Friends And Related Symbol Documentation

#### 6.171.5.1 sy\_is\_less

```
friend class sy_is_less [friend]
```

#### 6.171.5.2 sy\_swap

```
friend class sy_swap [friend]
```

#### 6.171.5.3 canonicalize

```
int canonicalize (
 exvector::iterator v,
 const symmetry & symm) [friend]
```

Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>v</i>    | Start of expression vector |
| <i>symm</i> | Root node of symmetry tree |

## Returns

the overall sign introduced by the reordering (+1, -1 or 0) or `numeric_limits<int>::max()` if nothing changed

## 6.171.6 Member Data Documentation

### 6.171.6.1 type

`symmetry_type` GiNaC::symmetry::type [private]

Type of symmetry described by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_type\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), [has\\_symmetry\(\)](#), [read\\_archive\(\)](#), [set\\_type\(\)](#), and [validate\(\)](#).

### 6.171.6.2 indices

`std::set<unsigned>` GiNaC::symmetry::indices [private]

Sorted union set of all indices handled by this node.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [read\\_archive\(\)](#), [symmetry\(\)](#), and [validate\(\)](#).

### 6.171.6.3 children

`exvector` GiNaC::symmetry::children [private]

Vector of child nodes.

Referenced by [add\(\)](#), [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [has\\_cyclic\(\)](#), [has\\_nonsymmetric\(\)](#), and [has\\_symmetry\(\)](#).

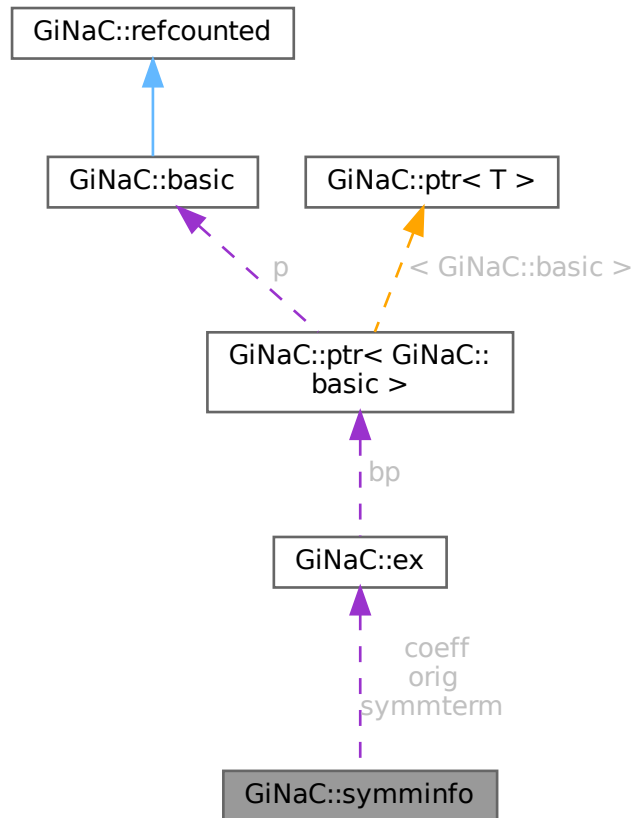
The documentation for this class was generated from the following files:

- [symmetry.h](#)
- [symmetry.cpp](#)

## 6.172 GiNaC::symminfo Class Reference

This structure stores the individual symmetrized terms obtained during the simplification of sums.

Collaboration diagram for GiNaC::symminfo:



### Public Member Functions

- [symminfo](#) ()
- [symminfo](#) (const [ex](#) &symmterm\_, const [ex](#) &orig\_, size\_t num\_)

### Public Attributes

- [ex symmterm](#)  
*symmetrized term*
- [ex coeff](#)  
*coefficient of symmetrized term*
- [ex orig](#)  
*original term*
- [size\\_t num](#)  
*how many symmetrized terms resulted from the original term*

## 6.172.1 Detailed Description

This structure stores the individual symmetrized terms obtained during the simplification of sums.

## 6.172.2 Constructor & Destructor Documentation

### 6.172.2.1 symminfo() [1/2]

```
GiNaC::symminfo::symminfo () [inline]
```

### 6.172.2.2 symminfo() [2/2]

```
GiNaC::symminfo::symminfo (
 const ex & symmterm_,
 const ex & orig_,
 size_t num_) [inline]
```

References [coeff](#), [GiNaC::ex::nops\(\)](#), [GiNaC::ex::op\(\)](#), and [symmterm](#).

## 6.172.3 Member Data Documentation

### 6.172.3.1 symmterm

```
ex GiNaC::symminfo::symmterm
```

symmetrized term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm::operator\(\)\(\)](#), and [symminfo\(\)](#).

### 6.172.3.2 coeff

```
ex GiNaC::symminfo::coeff
```

coefficient of symmetrized term

Referenced by [symminfo\(\)](#).

### 6.172.3.3 orig

```
ex GiNaC::symminfo::orig
```

original term

Referenced by [GiNaC::symminfo\\_is\\_less\\_by\\_orig::operator\(\)\(\)](#).

### 6.172.3.4 num

```
size_t GiNaC::symminfo::num
```

how many symmetrized terms resulted from the original term

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.173 GiNaC::symminfo\_is\_less\_by\_orig Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

### 6.173.1 Member Function Documentation

#### 6.173.1.1 operator>()

```
bool GiNaC::symminfo_is_less_by_orig::operator() (
 const symminfo & si1,
 const symminfo & si2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::orig](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.174 GiNaC::symminfo\_is\_less\_by\_symmterm Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [symminfo](#) &si1, const [symminfo](#) &si2) const

### 6.174.1 Member Function Documentation

#### 6.174.1.1 operator>()

```
bool GiNaC::symminfo_is_less_by_symmterm::operator() (
 const symminfo & si1,
 const symminfo & si2) const [inline]
```

References [GiNaC::ex::compare\(\)](#), and [GiNaC::symminfo::symmterm](#).

The documentation for this class was generated from the following file:

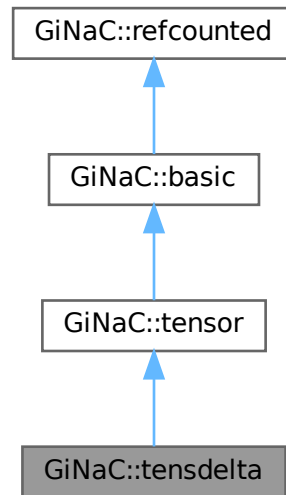
- [indexed.cpp](#)

## 6.175 GiNaC::tensdelta Class Reference

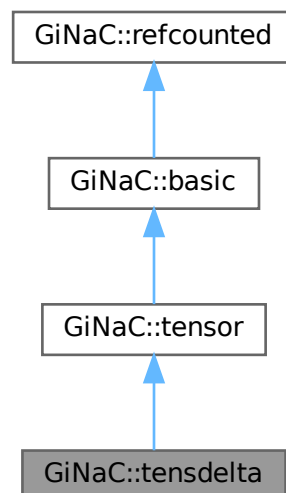
This class represents the delta tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensdelta:



Collaboration diagram for GiNaC::tensdelta:



## Public Member Functions

- bool `info` (unsigned inf) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed delta tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed delta tensor with something else.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)

- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int [n](#)=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &`m`, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &`s`, unsigned `nth=1`) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &`other`) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &`other`) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `unsigned return_type` () const override
- `void do_print` (const `print_context` &`c`, unsigned `level`) const
- `void do_print_latex` (const `print_latex` &`c`, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::tensor`

- `unsigned return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- `virtual ex eval_ncmul` (const `exvector` &`v`) const
- `virtual bool match_same_type` (const `basic` &`other`) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- `virtual ex derivative` (const `symbol` &`s`) const  
*Default implementation of `ex::diff()`.*
- `virtual int compare_same_type` (const `basic` &`other`) const  
*Returns order relation between two objects of same type.*
- `virtual bool is_equal_same_type` (const `basic` &`other`) const  
*Returns true if two objects of same type are equal.*
- `virtual unsigned calchash` () const  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*

- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.175.1 Detailed Description

This class represents the delta tensor.

If indexed, it must have exactly two indices of the same type.

## 6.175.2 Member Function Documentation

### 6.175.2.1 [info\(\)](#)

```
bool GiNaC::tensdelta::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.175.2.2 [eval\\_indexed\(\)](#)

```
ex GiNaC::tensdelta::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed delta tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GiNaC::\\_ex1](#), [GiNaC::idx::get\\_dim\(\)](#), [GiNaC::idx::get\\_value\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::info\\_flags::integer](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 6.175.2.3 contract\_with()

```
bool GiNaC::tensdelta::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed delta tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.175.2.4 return\_type()

```
unsigned GiNaC::tensdelta::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.175.2.5 do\_print()

```
void GiNaC::tensdelta::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.175.2.6 do\_print\_latex()

```
void GiNaC::tensdelta::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

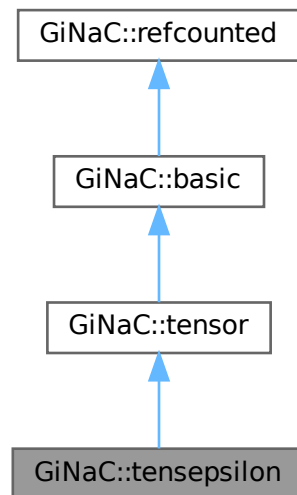
- [tensor.h](#)
- [tensor.cpp](#)

## 6.176 GiNaC::tensepsilon Class Reference

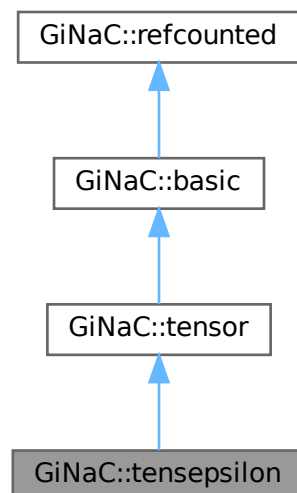
This class represents the totally antisymmetric epsilon tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensepsilon:



Collaboration diagram for GiNaC::tensepsilon:



## Public Member Functions

- `tensepsilon` (bool `minkowski`, bool `pos_sig`)
- bool `info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed epsilon tensor.*
- bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of epsilon tensor with something else.*
- void `archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- void `read_archive` (const `archive_node` &n, `lst` &syms) override  
*Read (a.k.a.*

## Public Member Functions inherited from `GiNaC::tensor`

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const

- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex & let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex & operator\[\]](#) (const [ex](#) &index)
- virtual [ex & operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_info](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*

- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- unsigned `return_type` () const override
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::tensor`

- unsigned `return_type` () const override

## Protected Member Functions inherited from `GiNaC::basic`

- `basic` ()
- virtual `ex eval_ncmul` (const `exvector` &v) const
- virtual bool `match_same_type` (const `basic` &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual `ex derivative` (const `symbol` &s) const  
*Default implementation of `ex::diff()`.*
- virtual int `compare_same_type` (const `basic` &other) const  
*Returns order relation between two objects of same type.*
- virtual bool `is_equal_same_type` (const `basic` &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned `calchash` () const

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

- void [ensure\\_if\\_modifiable](#) () const

Ensure the object may be modified without hurting others, throws if this is not the case.

- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

Default output to stream.

- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const

Tree output to stream.

- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const

Python parsable output to stream.

## Private Attributes

- bool [minkowski](#)

If true, tensor is in Minkowski-type space.

- bool [pos\\_sig](#)

If true, the metric is assumed to be  $\text{diag}(-1, 1, 1 \dots)$ .

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)

of type [status\\_flags](#)

- unsigned [hashvalue](#)

hash value

## 6.176.1 Detailed Description

This class represents the totally antisymmetric epsilon tensor.

If indexed, all indices must be of the same type and their number must be equal to the dimension of the index space.

## 6.176.2 Constructor & Destructor Documentation

### 6.176.2.1 [tensepsilon\(\)](#)

```
GiNaC::tensepsilon::tensepsilon (
 bool minkowski,
 bool pos_sig)
```

## 6.176.3 Member Function Documentation

### 6.176.3.1 info()

```
bool GiNaC::tensepsilon::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.176.3.2 eval\_indexed()

```
ex GiNaC::tensepsilon::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed epsilon tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex0](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::is\\_zero\(\)](#), [minkowski](#), [GiNaC::info\\_flags::nonnegint](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [pos\\_sig](#), and [x](#).

### 6.176.3.3 contract\_with()

```
bool GiNaC::tensepsilon::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of epsilon tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::\\_ex1](#), [GiNaC::delta\\_tensor\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::lorentz\\_g\(\)](#), [GiNaC::metric\\_tensor\(\)](#), [minkowski](#), [pos\\_sig](#), and [GiNaC::ex::simplify\\_indexed\(\)](#).

### 6.176.3.4 archive()

```
void GiNaC::tensepsilon::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.5 read\_archive()

```
void GiNaC::tensepsilon::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [minkowski](#), [n](#), and [pos\\_sig](#).

### 6.176.3.6 return\_type()

```
unsigned GiNaC::tensepsilon::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.176.3.7 do\_print()

```
void GiNaC::tensepsilon::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

### 6.176.3.8 do\_print\_latex()

```
void GiNaC::tensepsilon::do_print_latex (
 const print_latex & c,
 unsigned level) const [protected]
```

## 6.176.4 Member Data Documentation

### 6.176.4.1 minkowski

```
bool GiNaC::tensepsilon::minkowski [private]
```

If true, tensor is in Minkowski-type space.

Otherwise it is in a Euclidean space.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

### 6.176.4.2 pos\_sig

```
bool GiNaC::tensepsilon::pos_sig [private]
```

If true, the metric is assumed to be  $\text{diag}(-1,1,1,\dots)$ .

Otherwise it is  $\text{diag}(1,-1,-1,\dots)$ . This is only relevant if `minkowski = true`.

Referenced by [archive\(\)](#), [contract\\_with\(\)](#), [eval\\_indexed\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

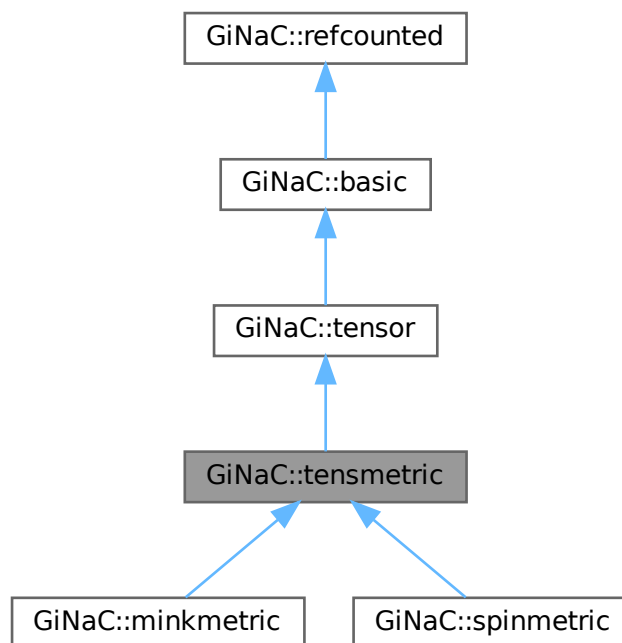
- [tensor.h](#)
- [tensor.cpp](#)

## 6.177 GiNaC::tensmetric Class Reference

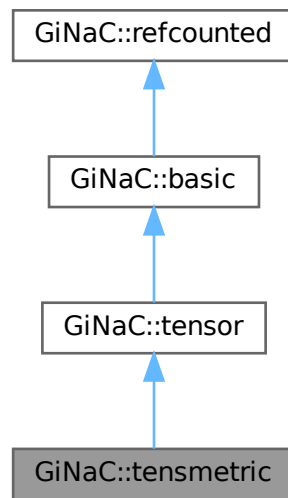
This class represents a general metric tensor which can be used to raise/lower indices.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensmetric:



Collaboration diagram for GiNaC::tensmetric:



### Public Member Functions

- `bool info` (unsigned int) const override  
*Information about the object.*
- `ex eval_indexed` (const `basic` &i) const override  
*Automatic symbolic evaluation of an indexed metric tensor.*
- `bool contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const override  
*Contraction of an indexed metric tensor with something else.*

### Public Member Functions inherited from `GiNaC::tensor`

- `bool replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

### Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*

- virtual `ex evalf ()` const  
*Evaluate object numerically.*
- virtual `ex evalm ()` const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ ()` const  
*Evaluate integrals, if result is known.*
- virtual void `print (const print_context &c, unsigned level=0)` const  
*Output to stream.*
- virtual void `dbgprint ()` const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprintree ()` const  
*Little wrapper around printree to be called within a debugger.*
- virtual unsigned `precedence ()` const  
*Return relative operator precedence (for parenthezing output).*
- virtual `size_t nops ()` const  
*Number of operands/members.*
- virtual `ex op (size_t i)` const  
*Return operand/member at position i.*
- virtual `ex operator[] (const ex &index)` const
- virtual `ex operator[] (size_t i)` const
- virtual `ex & let_op (size_t i)`  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[] (const ex &index)`
- virtual `ex & operator[] (size_t i)`
- virtual bool `has (const ex &other, unsigned options=0)` const  
*Test for occurrence of a pattern.*
- virtual bool `match (const ex &pattern, exmap &repls)` const  
*Check whether the expression matches a given pattern.*
- virtual `ex subs (const exmap &m, unsigned options=0)` const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map (map_function &f)` const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept (GiNaC::visitor &v)` const
- virtual bool `is_polynomial (const ex &var)` const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree (const ex &s)` const  
*Return degree of highest power in object s.*
- virtual int `ldegree (const ex &s)` const  
*Return degree of lowest power in object s.*
- virtual `ex coeff (const ex &s, int n=1)` const  
*Return coefficient of degree n in object s.*
- virtual `ex expand (unsigned options=0)` const  
*Expand expression, i.e.*
- virtual `ex collect (const ex &s, bool distributed=false)` const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series (const relational &r, int order, unsigned options=0)` const  
*Default implementation of `ex::series()`.*
- virtual `ex normal (exmap &repl, exmap &rev_lookup, lst &modifier)` const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational (exmap &repl)` const  
*Default implementation of `ex::to_rational()`.*

- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*
- virtual [ex scalar\\_mul\\_indexed](#) (const [ex](#) &self, const [numeric](#) &other) const  
*Multiply an indexed expression with a scalar.*
- virtual [return\\_type\\_t return\\_type\\_tinfo](#) () const
- virtual [ex conjugate](#) () const
- virtual [ex real\\_part](#) () const
- virtual [ex imag\\_part](#) () const
- template<class T >  
void [print\\_dispatch](#) (const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- void [print\\_dispatch](#) (const [registered\\_class\\_info](#) &ri, const [print\\_context](#) &c, unsigned level) const  
*Like [print\(\)](#), but dispatch to the specified class.*
- virtual void [archive](#) ([archive\\_node](#) &n) const  
*Save (serialize) the object into archive node.*
- virtual void [read\\_archive](#) (const [archive\\_node](#) &n, [lst](#) &syms)  
*Load (deserialize) the object from an archive node.*
- [ex subs\\_one\\_level](#) (const [exmap](#) &m, unsigned [options](#)) const  
*Helper function for [subs\(\)](#).*
- [ex diff](#) (const [symbol](#) &s, unsigned nth=1) const  
*Default interface of nth derivative [ex::diff\(s, n\)](#).*
- int [compare](#) (const [basic](#) &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool [is\\_equal](#) (const [basic](#) &other) const  
*Test for syntactic equality.*
- const [basic](#) & [hold](#) () const  
*Stop further evaluation.*
- unsigned [gethash](#) () const
- const [basic](#) & [setflag](#) (unsigned f) const  
*Set some [status\\_flags](#).*
- const [basic](#) & [clearflag](#) (unsigned f) const  
*Clear some [status\\_flags](#).*

## Public Member Functions inherited from [GiNaC::refcounted](#)

- [refcounted](#) () noexcept
- unsigned int [add\\_reference](#) () noexcept
- unsigned int [remove\\_reference](#) () noexcept
- unsigned int [get\\_refcount](#) () const noexcept
- void [set\\_refcount](#) (unsigned int r) noexcept

### Protected Member Functions

- unsigned [return\\_type](#) () const override
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const

### Protected Member Functions inherited from [GiNaC::tensor](#)

- unsigned [return\\_type](#) () const override

### Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

### Additional Inherited Members

### Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

## 6.177.1 Detailed Description

This class represents a general metric tensor which can be used to raise/lower indices.

If indexed, it must have exactly two indices of the same type which must be of class `varidx` or a subclass.

## 6.177.2 Member Function Documentation

### 6.177.2.1 info()

```
bool GiNaC::tensmetric::info (
 unsigned inf) const [override], [virtual]
```

Information about the object.

See also

class [info\\_flags](#)

Reimplemented from [GiNaC::basic](#).

References [GiNaC::info\\_flags::real](#).

### 6.177.2.2 eval\_indexed()

```
ex GiNaC::tensmetric::eval_indexed (
 const basic & i) const [override], [virtual]
```

Automatic symbolic evaluation of an indexed metric tensor.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::delta\\_tensor\(\)](#), [GiNaC::idx::get\\_dim\(\)](#), [GINAC\\_ASSERT](#), [GiNaC::basic::hold\(\)](#), [GiNaC::varidx::is\\_covariant\(\)](#), [GiNaC::ex::is\\_equal\(\)](#), [m](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::subs\\_options::no\\_pattern](#), [GiNaC::basic::nops\(\)](#), [GiNaC::basic::op\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::basic::subs\(\)](#).

### 6.177.2.3 contract\_with()

```
bool GiNaC::tensmetric::contract_with (
 exvector::iterator self,
 exvector::iterator other,
 exvector & v) const [override], [virtual]
```

Contraction of an indexed metric tensor with something else.

Reimplemented from [GiNaC::basic](#).

References [GINAC\\_ASSERT](#), and [GiNaC::tensor::replace\\_contr\\_index\(\)](#).

### 6.177.2.4 return\_type()

```
unsigned GiNaC::tensmetric::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::commutative](#).

### 6.177.2.5 do\_print()

```
void GiNaC::tensmetric::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

The documentation for this class was generated from the following files:

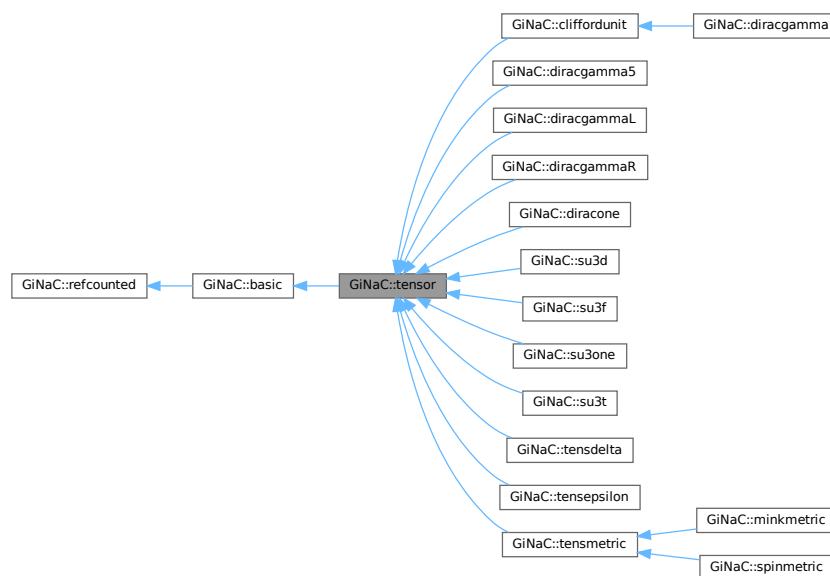
- [tensor.h](#)
- [tensor.cpp](#)

## 6.178 GiNaC::tensor Class Reference

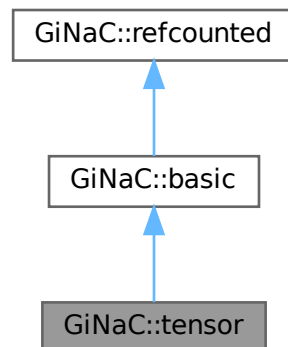
This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensor:



Collaboration diagram for GiNaC::tensor:



## Public Member Functions

- bool `replace_contr_index` (exvector::iterator self, exvector::iterator other) const  
*Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).*

## Public Member Functions inherited from GiNaC::basic

- virtual `~basic` ()  
*basic destructor, virtual because class ex will delete objects of derived classes via a basic\*.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprinttree` () const  
*Little wrapper around printtree to be called within a debugger.*

- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual size\_t [nops](#) () const  
*Number of operands/members.*
- virtual [ex op](#) (size\_t i) const  
*Return operand/member at position i.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [let\\_op](#) (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const  
*Test for occurrence of a pattern.*
- virtual bool [match](#) (const [ex](#) &pattern, [exmap](#) &repls) const  
*Check whether the expression matches a given pattern.*
- virtual [ex subs](#) (const [exmap](#) &m, unsigned [options](#)=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual [ex map](#) ([map\\_function](#) &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void [accept](#) ([GiNaC::visitor](#) &v) const
- virtual bool [is\\_polynomial](#) (const [ex](#) &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int [degree](#) (const [ex](#) &s) const  
*Return degree of highest power in object s.*
- virtual int [ldegree](#) (const [ex](#) &s) const  
*Return degree of lowest power in object s.*
- virtual [ex coeff](#) (const [ex](#) &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual [ex expand](#) (unsigned [options](#)=0) const  
*Expand expression, i.e.*
- virtual [ex collect](#) (const [ex](#) &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const  
*Default implementation of [ex::series\(\)](#).*
- virtual [ex normal](#) ([exmap](#) &repl, [exmap](#) &rev\_lookup, [lst](#) &modifier) const  
*Default implementation of [ex::normal\(\)](#).*
- virtual [ex to\\_rational](#) ([exmap](#) &repl) const  
*Default implementation of [ex::to\\_rational\(\)](#).*
- virtual [ex to\\_polynomial](#) ([exmap](#) &repl) const
- virtual [numeric integer\\_content](#) () const
- virtual [ex smod](#) (const [numeric](#) &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual [numeric max\\_coefficient](#) () const  
*Implementation [ex::max\\_coefficient\(\)](#).*
- virtual [exvector get\\_free\\_indices](#) () const  
*Return a vector containing the free indices of an expression.*
- virtual [ex add\\_indexed](#) (const [ex](#) &self, const [ex](#) &other) const  
*Add two indexed expressions.*

- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `return_type_t return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- virtual void `archive` (`archive_node` &n) const  
*Save (serialize) the object into archive node.*
- virtual void `read_archive` (const `archive_node` &n, `lst` &syms)  
*Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- unsigned `return_type` () const override

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.178.1 Detailed Description

This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.

They are represented without indices. To attach indices to them, wrap them in an object of class [indexed](#).

### 6.178.2 Member Function Documentation

#### 6.178.2.1 [return\\_type\(\)](#)

```
unsigned GiNaC::tensor::return_type () const [inline], [override], [protected], [virtual]
```

Reimplemented from [GiNaC::basic](#).

References [GiNaC::return\\_types::noncommutative\\_composite](#).

## 6.178.2.2 replace\_contr\_index()

```
bool GiNaC::tensor::replace_contr_index (
 exvector::iterator self,
 exvector::iterator other) const
```

Replace dummy index in contracted-with object by the contracting object's second index (used internally for delta and metric tensor contractions).

References [GiNaC::\\_ex1](#), [GiNaC::is\\_dummy\\_pair\(\)](#), [GiNaC::idx::is\\_symbolic\(\)](#), [GiNaC::idx::minimal\\_dim\(\)](#), [GiNaC::idx::replace\\_dim\(\)](#), and [GiNaC::ex::subs\(\)](#).

Referenced by [GiNaC::tensdelta::contract\\_with\(\)](#), and [GiNaC::tensmetric::contract\\_with\(\)](#).

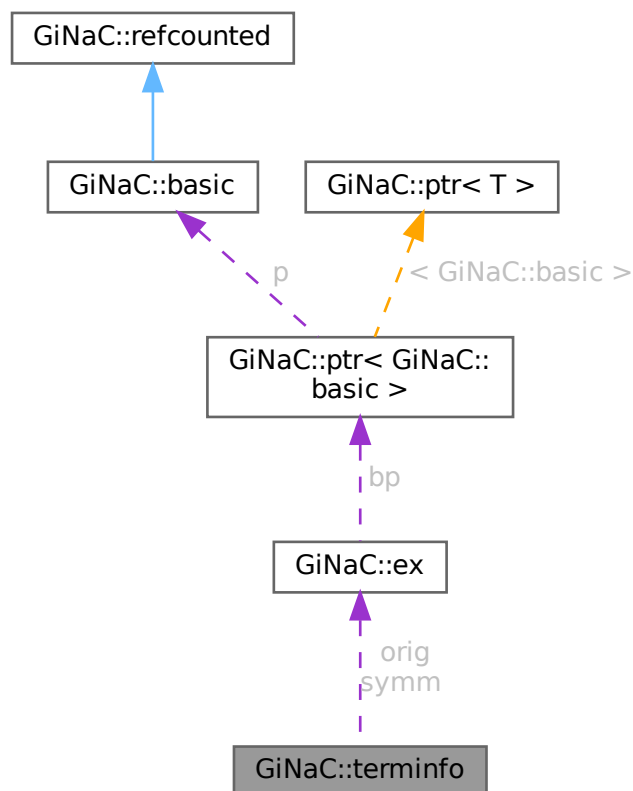
The documentation for this class was generated from the following files:

- [tensor.h](#)
- [tensor.cpp](#)

## 6.179 GiNaC::terminfo Class Reference

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

Collaboration diagram for GiNaC::terminfo:



## Public Member Functions

- [terminfo](#) (const [ex](#) &orig\_, const [ex](#) &symm\_)

## Public Attributes

- [ex orig](#)  
*original term*
- [ex symm](#)  
*symmetrized term*

## 6.179.1 Detailed Description

This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.

## 6.179.2 Constructor & Destructor Documentation

### 6.179.2.1 [terminfo\(\)](#)

```
GiNaC::terminfo::terminfo (
 const ex & orig_,
 const ex & symm_) [inline]
```

## 6.179.3 Member Data Documentation

### 6.179.3.1 [orig](#)

[ex](#) [GiNaC::terminfo::orig](#)

original term

### 6.179.3.2 [symm](#)

[ex](#) [GiNaC::terminfo::symm](#)

symmetrized term

Referenced by [GiNaC::terminfo\\_is\\_less::operator>\(\)\(\)](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.180 GiNaC::terminfo\_is\_less Class Reference

### Public Member Functions

- `bool operator()` (const `terminfo` &ti1, const `terminfo` &ti2) const

### 6.180.1 Member Function Documentation

#### 6.180.1.1 operator>()

```
bool GiNaC::terminfo_is_less::operator() (
 const terminfo & ti1,
 const terminfo & ti2) const [inline]
```

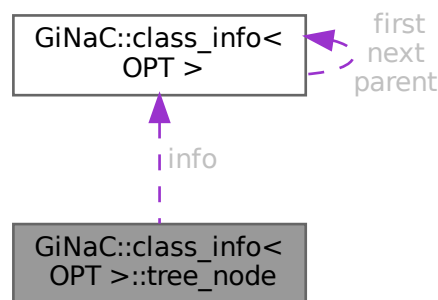
References [GiNaC::ex::compare\(\)](#), and [GiNaC::terminfo::symm](#).

The documentation for this class was generated from the following file:

- [indexed.cpp](#)

## 6.181 GiNaC::class\_info< OPT >::tree\_node Struct Reference

Collaboration diagram for GiNaC::class\_info< OPT >::tree\_node:



### Public Member Functions

- `tree_node` (`class_info` \*i)
- void `add_child` (`tree_node` \*n)

### Public Attributes

- `std::vector< tree_node * >` children
- `class_info *` info

## 6.181.1 Constructor & Destructor Documentation

### 6.181.1.1 `tree_node()`

```
template<class OPT >
GiNaC::class_info< OPT >::tree_node::tree_node (
 class_info * i) [inline]
```

## 6.181.2 Member Function Documentation

### 6.181.2.1 `add_child()`

```
template<class OPT >
void GiNaC::class_info< OPT >::tree_node::add_child (
 tree_node * n) [inline]
```

References [GiNaC::class\\_info< OPT >::tree\\_node::children](#), and [n](#).

## 6.181.3 Member Data Documentation

### 6.181.3.1 `children`

```
template<class OPT >
std::vector<tree_node *> GiNaC::class_info< OPT >::tree_node::children
```

Referenced by [GiNaC::class\\_info< OPT >::tree\\_node::add\\_child\(\)](#).

### 6.181.3.2 `info`

```
template<class OPT >
class_info* GiNaC::class_info< OPT >::tree_node::info
```

The documentation for this struct was generated from the following file:

- [class\\_info.h](#)

## 6.182 GiNaC::unarchive\_table\_t Class Reference

```
#include <archive.h>
```

### Public Member Functions

- [unarchive\\_table\\_t\(\)](#)
- [~unarchive\\_table\\_t\(\)](#)
- [synthesize\\_func find](#) (const std::string &classname) const
- void [insert](#) (const std::string &classname, [synthesize\\_func](#) f)

### Static Private Attributes

- static int [usecount](#) = 0
- static [unarchive\\_map\\_t](#) \* [unarch\\_map](#) = nullptr

## 6.182.1 Constructor & Destructor Documentation

### 6.182.1.1 unarchive\_table\_t()

```
GiNaC::unarchive_table_t::unarchive_table_t ()
```

References [unarch\\_map](#), and [usecount](#).

### 6.182.1.2 ~unarchive\_table\_t()

```
GiNaC::unarchive_table_t::~~unarchive_table_t ()
```

References [unarch\\_map](#), and [usecount](#).

## 6.182.2 Member Function Documentation

### 6.182.2.1 find()

```
synthesize_func GiNaC::unarchive_table_t::find (
 const std::string & classname) const
```

References [unarch\\_map](#).

Referenced by [GiNaC::find\\_factory\\_fcn\(\)](#).

### 6.182.2.2 insert()

```
void GiNaC::unarchive_table_t::insert (
 const std::string & classname,
 synthesize_func f)
```

References [unarch\\_map](#).

## 6.182.3 Member Data Documentation

### 6.182.3.1 usecount

```
int GiNaC::unarchive_table_t::usecount = 0 [static], [private]
```

Referenced by [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

### 6.182.3.2 unarch\_map

```
unarchive_map_t * GiNaC::unarchive_table_t::unarch_map = nullptr [static], [private]
```

Referenced by [find\(\)](#), [insert\(\)](#), [unarchive\\_table\\_t\(\)](#), and [~unarchive\\_table\\_t\(\)](#).

The documentation for this class was generated from the following files:

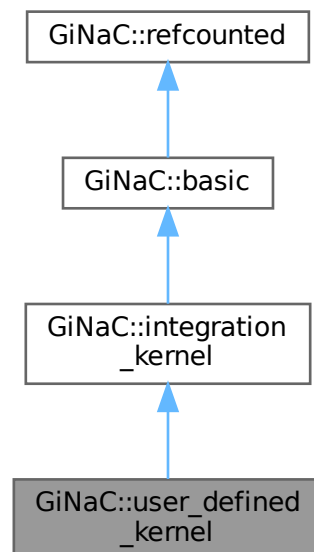
- [archive.h](#)
- [archive.cpp](#)

## 6.183 GiNaC::user\_defined\_kernel Class Reference

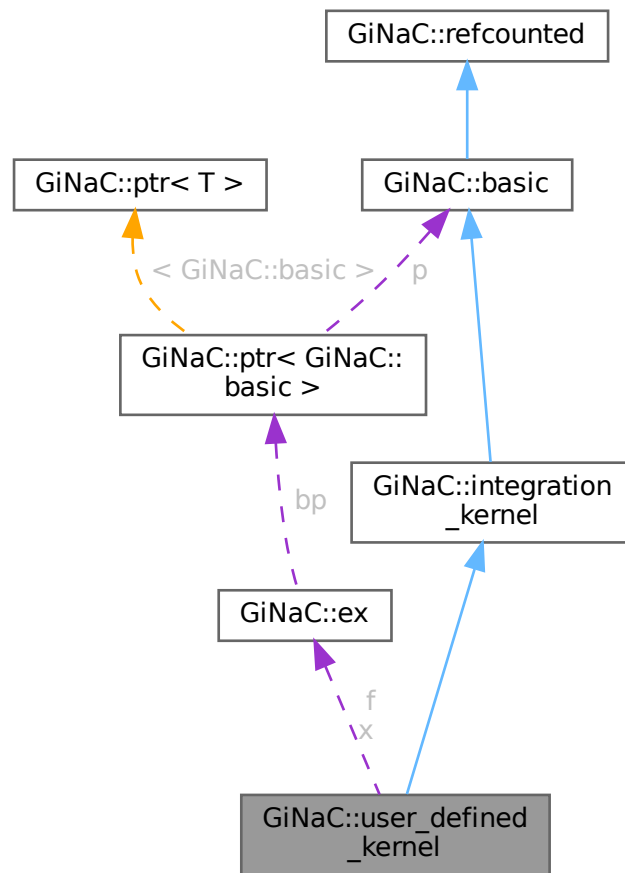
A user-defined integration kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::user\_defined\_kernel:



Collaboration diagram for GiNaC::user\_defined\_kernel:



## Public Member Functions

- **user\_defined\_kernel** (const **ex** &f, const **ex** &x)
- **size\_t nops** () const override  
*Number of operands/members.*
- **ex op** (size\_t i) const override  
*Return operand/member at position i.*
- **ex & let\_op** (size\_t i) override  
*Return modifiable operand/member at position i.*
- **bool is\_numeric** (void) const override  
*This routine returns true, if the integration kernel can be evaluated numerically.*
- **ex Laurent\_series** (const **ex** &x, int order) const override  
*Returns the Laurent series, starting possibly with the pole term.*

## Public Member Functions inherited from [GiNaC::integration\\_kernel](#)

- [ex series](#) (const [relational](#) &r, int [order](#), unsigned [options](#)=0) const override  
*Default implementation of [ex::series\(\)](#).*
- virtual bool [has\\_trailing\\_zero](#) (void) const  
*This routine returns true, if the integration kernel has a trailing zero.*
- virtual [ex get\\_numerical\\_value](#) (const [ex](#) &lambda, int N\_trunc=0) const  
*Evaluates the integrand at lambda.*
- size\_t [get\\_cache\\_size](#) (void) const  
*Returns the current size of the cache.*
- void [set\\_cache\\_step](#) (int cache\_steps) const  
*Sets the step size by which the cache is increased.*
- [ex get\\_series\\_coeff](#) (int i) const  
*Wrapper around [series\\_coeff\(i\)](#), converts cl\_N to numeric.*
- cln::cl\_N [series\\_coeff](#) (int i) const  
*Subclasses have either to implement [series\\_coeff\\_impl](#) or the two methods [Laurent\\_series](#) and [uses\\_Laurent\\_series](#).*

## Public Member Functions inherited from [GiNaC::basic](#)

- virtual [~basic](#) ()  
*basic destructor, virtual because class [ex](#) will delete objects of derived classes via a [basic\\*](#).*
- [basic](#) (const [basic](#) &other)
- const [basic](#) & [operator=](#) (const [basic](#) &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual [basic](#) \* [duplicate](#) () const  
*Create a clone of this object on the heap.*
- virtual [ex eval](#) () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual [ex evalf](#) () const  
*Evaluate object numerically.*
- virtual [ex evalm](#) () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual [ex eval\\_integ](#) () const  
*Evaluate integrals, if result is known.*
- virtual [ex eval\\_indexed](#) (const [basic](#) &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*
- virtual void [print](#) (const [print\\_context](#) &c, unsigned level=0) const  
*Output to stream.*
- virtual void [dbgprint](#) () const  
*Little wrapper around [print](#) to be called within a debugger.*
- virtual void [dbgprinttree](#) () const  
*Little wrapper around [printtree](#) to be called within a debugger.*
- virtual unsigned [precedence](#) () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool [info](#) (unsigned inf) const  
*Information about the object.*
- virtual [ex operator\[\]](#) (const [ex](#) &index) const
- virtual [ex operator\[\]](#) (size\_t i) const
- virtual [ex](#) & [operator\[\]](#) (const [ex](#) &index)
- virtual [ex](#) & [operator\[\]](#) (size\_t i)
- virtual bool [has](#) (const [ex](#) &other, unsigned [options](#)=0) const

- Test for occurrence of a pattern.*

  - virtual bool `match` (const `ex` &pattern, `exmap` &repls) const
- Check whether the expression matches a given pattern.*

  - virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const
- Substitute a set of objects by arbitrary expressions.*

  - virtual `ex map` (`map_function` &f) const
- Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*

  - virtual void `accept` (`GiNaC::visitor` &v) const
  - virtual bool `is_polynomial` (const `ex` &var) const
- Check whether this is a polynomial in the given variables.*

  - virtual int `degree` (const `ex` &s) const
- Return degree of highest power in object s.*

  - virtual int `ldegree` (const `ex` &s) const
- Return degree of lowest power in object s.*

  - virtual `ex coeff` (const `ex` &s, int `n`=1) const
- Return coefficient of degree n in object s.*

  - virtual `ex expand` (unsigned `options`=0) const
- Expand expression, i.e.*

  - virtual `ex collect` (const `ex` &s, bool `distributed`=false) const
- Sort expanded expression in terms of powers of some object(s).*

  - virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const
- Default implementation of `ex::normal()`.*

  - virtual `ex to_rational` (`exmap` &repl) const
- Default implementation of `ex::to_rational()`.*

  - virtual `ex to_polynomial` (`exmap` &repl) const
  - virtual `numeric integer_content` () const
  - virtual `ex smod` (const `numeric` &xi) const
- Apply symmetric modular homomorphism to an expanded multivariate polynomial.*

  - virtual `numeric max_coefficient` () const
- Implementation `ex::max_coefficient()`.*

  - virtual `exvector get_free_indices` () const
- Return a vector containing the free indices of an expression.*

  - virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const
- Add two indexed expressions.*

  - virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const
- Multiply an indexed expression with a scalar.*

  - virtual bool `contract_with` (`exvector::iterator` self, `exvector::iterator` other, `exvector` &v) const
- Try to contract two indexed expressions that appear in the same product.*

  - virtual unsigned `return_type` () const
  - virtual `return_type_t return_type_tinfo` () const
  - virtual `ex conjugate` () const
  - virtual `ex real_part` () const
  - virtual `ex imag_part` () const
- template<class T >

  - void `print_dispatch` (const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const
- Like `print()`, but dispatch to the specified class.*

  - virtual void `archive` (`archive_node` &n) const
- Save (serialize) the object into archive node.*

  - virtual void `read_archive` (const `archive_node` &n, `lst` &syms)

- *Load (deserialize) the object from an archive node.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned `nth`=1) const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- `int compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- `bool is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- `const basic & hold` () const  
*Stop further evaluation.*
- `unsigned gethash` () const
- `const basic & setflag` (unsigned `f`) const  
*Set some `status_flags`.*
- `const basic & clearflag` (unsigned `f`) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- `unsigned int add_reference` () noexcept
- `unsigned int remove_reference` () noexcept
- `unsigned int get_refcount` () const noexcept
- `void set_refcount` (unsigned int `r`) noexcept

## Protected Member Functions

- `bool uses_Laurent_series` () const override  
*Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from `GiNaC::integration_kernel`

- `virtual cln::cl_N series_coeff_impl` (int `i`) const  
*For  $\omega = d\lambda$  only the coefficient of  $\lambda^0$  is non-zero.*
- `ex get_numerical_value_impl` (const `ex` &lambda, const `ex` &pre, int `shift`, int `N_trunc`) const  
*The actual implementation for computing a numerical value for the integrand.*
- `void do_print` (const `print_context` &c, unsigned `level`) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- virtual unsigned [calchash](#) () const  
*Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- [ex f](#)
- [ex x](#)

## Protected Attributes inherited from [GiNaC::integration\\_kernel](#)

- int [cache\\_step\\_size](#)
- std::vector< [cln::cl\\_N](#) > [series\\_vec](#)

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.183.1 Detailed Description

A user-defined integration kernel.

The input is an expression  $f$ , depending on a variable  $x$ . It is assumed that  $f$  has a Laurent expansion around  $x = 0$  and maximally a simple pole at  $x = 0$ .

## 6.183.2 Constructor & Destructor Documentation

### 6.183.2.1 `user_defined_kernel()`

```
GiNaC::user_defined_kernel::user_defined_kernel (
 const ex & f,
 const ex & x)
```

## 6.183.3 Member Function Documentation

### 6.183.3.1 `nops()`

```
size_t GiNaC::user_defined_kernel::nops () const [override], [virtual]
```

Number of operands/members.

Reimplemented from [GiNaC::basic](#).

### 6.183.3.2 `op()`

```
ex GiNaC::user_defined_kernel::op (
 size_t i) const [override], [virtual]
```

Return operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [f](#), and [x](#).

### 6.183.3.3 `let_op()`

```
ex & GiNaC::user_defined_kernel::let_op (
 size_t i) [override], [virtual]
```

Return modifiable operand/member at position i.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::ensure\\_if\\_modifiable\(\)](#), [f](#), and [x](#).

### 6.183.3.4 `is_numeric()`

```
bool GiNaC::user_defined_kernel::is_numeric (
 void) const [override], [virtual]
```

This routine returns true, if the integration kernel can be evaluated numerically.

Reimplemented from [GiNaC::integration\\_kernel](#).

References [GiNaC::ex::evalf\(\)](#), [f](#), [GiNaC::ex::info\(\)](#), [GiNaC::info\\_flags::numeric](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.183.3.5 Laurent\_series()

```
ex GiNaC::user_defined_kernel::Laurent_series (
 const ex & x,
 int order) const [override], [virtual]
```

Returns the Laurent series, starting possibly with the pole term.

Neglected terms are of order  $O(x^{order})$ .

Reimplemented from [GiNaC::integration\\_kernel](#).

References [f](#), [order](#), [GiNaC::ex::series\(\)](#), [GiNaC::ex::subs\(\)](#), and [x](#).

### 6.183.3.6 uses\_Laurent\_series()

```
bool GiNaC::user_defined_kernel::uses_Laurent_series () const [override], [protected], [virtual]
```

Returns true, if the coefficients are computed from the Laurent series (in which case the method `Laurent_series` needs to be implemented).

Returns false if this is not the case (and the class has an implementation of `series_coeff_impl`).

Reimplemented from [GiNaC::integration\\_kernel](#).

### 6.183.3.7 do\_print()

```
void GiNaC::user_defined_kernel::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [f](#), [GiNaC::ex::print\(\)](#), and [x](#).

## 6.183.4 Member Data Documentation

### 6.183.4.1 f

```
ex GiNaC::user_defined_kernel::f [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

### 6.183.4.2 x

```
ex GiNaC::user_defined_kernel::x [protected]
```

Referenced by [do\\_print\(\)](#), [is\\_numeric\(\)](#), [Laurent\\_series\(\)](#), [let\\_op\(\)](#), and [op\(\)](#).

The documentation for this class was generated from the following files:

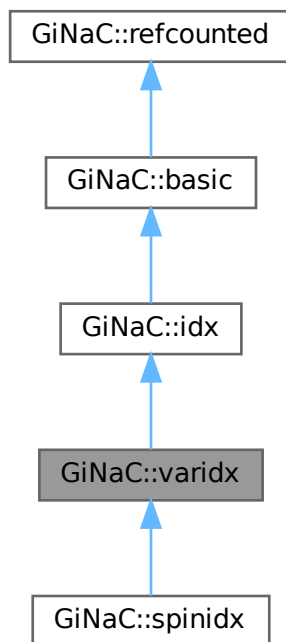
- [integration\\_kernel.h](#)
- [integration\\_kernel.cpp](#)

## 6.184 GiNaC::varidx Class Reference

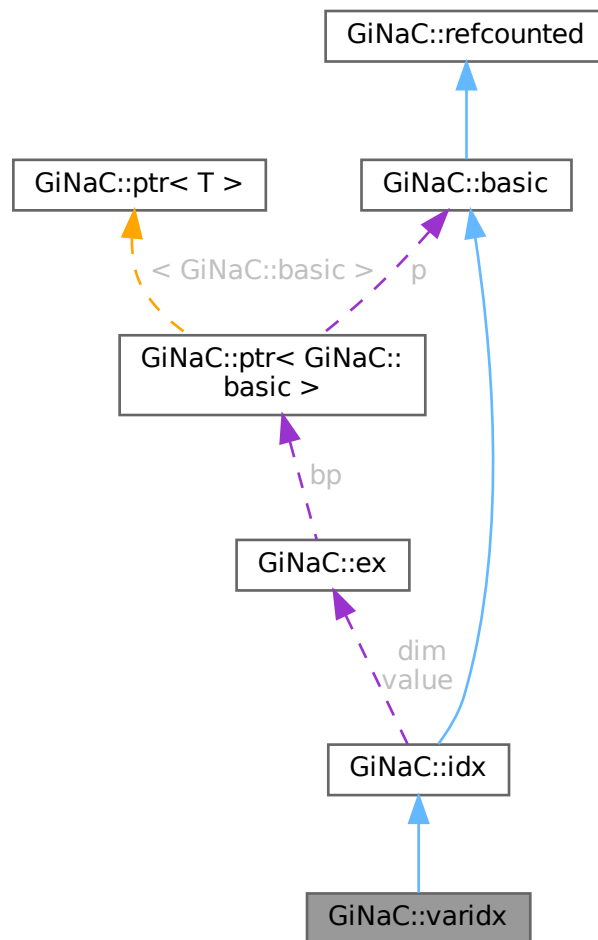
This class holds an index with a variance (co- or contravariant).

```
#include <idx.h>
```

Inheritance diagram for GiNaC::varidx:



Collaboration diagram for GiNaC::varidx:



## Public Member Functions

- `varidx` (const `ex` &`v`, const `ex` &`dim`, bool `covariant`=false)  
Construct index with given value, dimension and variance.
- bool `is_dummy_pair_same_type` (const `basic` &`other`) const override  
Check whether the index forms a dummy index pair with another index of the same type.
- void `archive` (`archive_node` &`n`) const override  
Save (serialize) the object into archive node.
- void `read_archive` (const `archive_node` &`n`, `lst` &`syms`) override  
Load (deserialize) the object from an archive node.
- bool `is_covariant` () const  
Check whether the index is covariant.
- bool `is_contravariant` () const  
Check whether the index is contravariant (not covariant).
- `ex toggle_variance` () const  
Make a new index with the same value but the opposite variance.

## Public Member Functions inherited from `GiNaC::idx`

- `idx` (const `ex` &`v`, const `ex` &`dim`)  
*Construct index with given value and dimension.*
- bool `info` (unsigned int) const override  
*Information about the object.*
- `size_t nops` () const override  
*Number of operands/members.*
- `ex op` (size\_t `i`) const override  
*Return operand/member at position `i`.*
- `ex map` (map\_function &`f`) const override  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- `ex evalf` () const override  
*By default, `basic::evalf` would evaluate the index value but we don't want `a.1` to become `a`.*
- `ex subs` (const `exmap` &`m`, unsigned `options`=0) const override  
*Substitute a set of objects by arbitrary expressions.*
- `ex get_value` () const  
*Get value of index.*
- bool `is_numeric` () const  
*Check whether the index is numeric.*
- bool `is_symbolic` () const  
*Check whether the index is symbolic.*
- `ex get_dim` () const  
*Get dimension of index space.*
- bool `is_dim_numeric` () const  
*Check whether the dimension is numeric.*
- bool `is_dim_symbolic` () const  
*Check whether the dimension is symbolic.*
- `ex replace_dim` (const `ex` &`new_dim`) const  
*Make a new index with the same value but a different dimension.*
- `ex minimal_dim` (const `idx` &`other`) const  
*Return the minimum of the dimensions of this and another index.*

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &`other`)
- const `basic` & `operator=` (const `basic` &`other`)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic * duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &`i`) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprnttree` () const  
*Little wrapper around prnttree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual bool `match` (const `ex` &pattern, `exmap` &repls) const  
*Check whether the expression matches a given pattern.*
- virtual void `accept` (GiNaC::visitor &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*
- virtual `exvector get_free_indices` () const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed` (const `ex` &self, const `ex` &other) const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed` (const `ex` &self, const `numeric` &other) const  
*Multiply an indexed expression with a scalar.*
- virtual bool `contract_with` (exvector::iterator self, exvector::iterator other, `exvector` &v) const  
*Try to contract two indexed expressions that appear in the same product.*

- virtual unsigned `return_type` () const
- virtual `return_type_t` `return_type_tinfo` () const
- virtual `ex conjugate` () const
- virtual `ex real_part` () const
- virtual `ex imag_part` () const
- template<class T >  
void `print_dispatch` (const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch` (const `registered_class_info` &ri, const `print_context` &c, unsigned level) const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level` (const `exmap` &m, unsigned `options`) const  
*Helper function for `subs()`.*
- `ex diff` (const `symbol` &s, unsigned nth=1) const  
*Default interface of nth derivative `ex::diff(s, n)`.*
- int `compare` (const `basic` &other) const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal` (const `basic` &other) const  
*Test for syntactic equality.*
- const `basic` & `hold` () const  
*Stop further evaluation.*
- unsigned `gethash` () const
- const `basic` & `setflag` (unsigned f) const  
*Set some `status_flags`.*
- const `basic` & `clearflag` (unsigned f) const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted` () noexcept
- unsigned int `add_reference` () noexcept
- unsigned int `remove_reference` () noexcept
- unsigned int `get_refcount` () const noexcept
- void `set_refcount` (unsigned int r) noexcept

## Protected Member Functions

- bool `match_same_type` (const `basic` &other) const override  
*Returns true if the attributes of two objects are similar enough for a match.*
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from `GiNaC::idx`

- `ex derivative` (const `symbol` &s) const override  
*Implementation of `ex::diff()` for an index always returns 0.*
- unsigned `calchash` () const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `print_index` (const `print_context` &c, unsigned level) const
- void `do_print` (const `print_context` &c, unsigned level) const
- void `do_print_csrc` (const `print_csrc` &c, unsigned level) const
- void `do_print_latex` (const `print_latex` &c, unsigned level) const
- void `do_print_tree` (const `print_tree` &c, unsigned level) const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Protected Attributes

- bool [covariant](#)  
 *$x.\mu$ , default is contravariant:  $x \sim \mu$*

## Protected Attributes inherited from [GiNaC::idx](#)

- [ex value](#)  
*Expression that constitutes the index (numeric or symbolic name)*
- [ex dim](#)  
*Dimension of space (can be symbolic or numeric)*

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.184.1 Detailed Description

This class holds an index with a variance (co- or contravariant).

There is an associated metric tensor that can be used to raise/lower indices.

### 6.184.2 Constructor & Destructor Documentation

#### 6.184.2.1 [varidx](#)()

```
GiNaC::varidx::varidx (
 const ex & v,
 const ex & dim,
 bool covariant = false)
```

Construct index with given value, dimension and variance.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>v</i>         | Value of index (numeric or symbolic)            |
| <i>dim</i>       | Dimension of index space (numeric or symbolic)  |
| <i>covariant</i> | Make covariant index (default is contravariant) |

## 6.184.3 Member Function Documentation

### 6.184.3.1 is\_dummy\_pair\_same\_type()

```
bool GiNaC::varidx::is_dummy_pair_same_type (
 const basic & other) const [override], [virtual]
```

Check whether the index forms a dummy index pair with another index of the same type.

Reimplemented from [GiNaC::idx](#).

References [covariant](#).

### 6.184.3.2 archive()

```
void GiNaC::varidx::archive (
 archive_node & n) const [override], [virtual]
```

Save (serialize) the object into archive node.

Archive the object.

Losely speaking, this method turns an expression into a byte stream (which can be saved and restored later on, or sent via network, etc.)

Reimplemented from [GiNaC::idx](#).

References [covariant](#), and [n](#).

### 6.184.3.3 read\_archive()

```
void GiNaC::varidx::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Load (deserialize) the object from an archive node.

Construct object from [archive\\_node](#).

#### Note

This method is essentially a constructor. However, constructors can't be virtual. So, if unarchiving routines are implemented as constructors one would need to define such a constructor in every class, even if all it does is simply calling constructor of a superclass.

Reimplemented from [GiNaC::idx](#).

References [covariant](#), and [n](#).

#### 6.184.3.4 match\_same\_type()

```
bool GiNaC::varidx::match_same_type (
 const basic & other) const [override], [protected], [virtual]
```

Returns true if the attributes of two objects are similar enough for a match.

This function must not match subexpressions (this is already done by [basic::match\(\)](#)). Only attributes not accessible by [op\(\)](#) should be compared. This is also the reason why this function doesn't take the wildcard replacement list from [match\(\)](#) as an argument: only subexpressions are subject to wildcard matches. Also, this function only needs to be implemented for container classes because [is\\_equal\\_same\\_type\(\)](#) is automatically used instead of [match\\_same\\_type\(\)](#) if [nops\(\)](#) == 0.

See also

[basic::match](#)

Reimplemented from [GiNaC::idx](#).

References [covariant](#), and [GINAC\\_ASSERT](#).

#### 6.184.3.5 is\_covariant()

```
bool GiNaC::varidx::is_covariant () const [inline]
```

Check whether the index is covariant.

References [covariant](#).

Referenced by [GiNaC::tensmetric::eval\\_indexed\(\)](#).

#### 6.184.3.6 is\_contravariant()

```
bool GiNaC::varidx::is_contravariant () const [inline]
```

Check whether the index is contravariant (not covariant).

References [covariant](#).

#### 6.184.3.7 toggle\_variance()

```
ex GiNaC::varidx::toggle_variance () const
```

Make a new index with the same value but the opposite variance.

References [GiNaC::basic::clearflag\(\)](#), [covariant](#), [GiNaC::basic::duplicate\(\)](#), and [GiNaC::status\\_flags::hash\\_calculated](#).

### 6.184.3.8 do\_print()

```
void GiNaC::varidx::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), [covariant](#), and [GiNaC::idx::print\\_index\(\)](#).

### 6.184.3.9 do\_print\_tree()

```
void GiNaC::varidx::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [covariant](#), [GiNaC::idx::dim](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), [GiNaC::ex::print\(\)](#), and [GiNaC::idx::value](#).

## 6.184.4 Member Data Documentation

### 6.184.4.1 covariant

```
bool GiNaC::varidx::covariant [protected]
```

x.mu, default is contravariant:  $x \sim \mu$

Referenced by [archive\(\)](#), [do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [is\\_contravariant\(\)](#), [is\\_covariant\(\)](#), [is\\_dummy\\_pair\\_same\\_type\(\)](#), [match\\_same\\_type\(\)](#), [read\\_archive\(\)](#), [toggle\\_variance\(\)](#), and [GiNaC::spinidx::toggle\\_variance\\_dot\(\)](#).

The documentation for this class was generated from the following files:

- [idx.h](#)
- [idx.cpp](#)

## 6.185 GiNaC::visitor Class Reference

Degenerate base class for visitors.

```
#include <basic.h>
```

### Protected Member Functions

- virtual [~visitor\(\)](#)

### 6.185.1 Detailed Description

Degenerate base class for visitors.

basic and derivative classes support Robert C. Martin's Acyclic Visitor pattern (cf. <http://condor.depaul.edu/dmumaugh/OOT/Design-Principles/acv.pdf> or chapter 10 of Andrei Alexandrescu's "Modern C++ Design").

### 6.185.2 Constructor & Destructor Documentation

#### 6.185.2.1 ~visitor()

```
virtual GiNaC::visitor::~~visitor () [inline], [protected], [virtual]
```

The documentation for this class was generated from the following file:

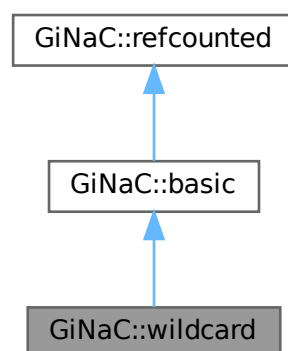
- [basic.h](#)

## 6.186 GiNaC::wildcard Class Reference

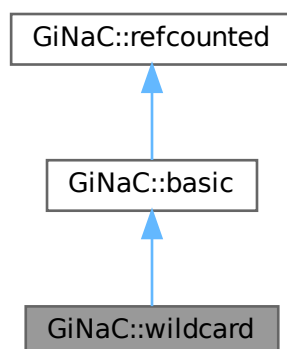
This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

```
#include <wildcard.h>
```

Inheritance diagram for GiNaC::wildcard:



Collaboration diagram for GiNaC::wildcard:



## Public Member Functions

- `wildcard` (unsigned `label`)  
*Construct wildcard with specified label.*
- `bool match` (const `ex` &pattern, `exmap` &repl\_lst) const override  
*Check whether the expression matches a given pattern.*
- `void archive` (`archive_node` &n) const override  
*Save (a.k.a.*
- `void read_archive` (const `archive_node` &n, `lst` &symbols) override  
*Read (a.k.a.*
- unsigned `get_label` () const

## Public Member Functions inherited from `GiNaC::basic`

- virtual `~basic` ()  
*basic destructor, virtual because class `ex` will delete objects of derived classes via a `basic*`.*
- `basic` (const `basic` &other)
- const `basic` & `operator=` (const `basic` &other)  
*basic assignment operator: the other object might be of a derived class.*
- virtual `basic` \* `duplicate` () const  
*Create a clone of this object on the heap.*
- virtual `ex eval` () const  
*Perform automatic non-interruptive term rewriting rules.*
- virtual `ex evalf` () const  
*Evaluate object numerically.*
- virtual `ex evalm` () const  
*Evaluate sums, products and integer powers of matrices.*
- virtual `ex eval_integ` () const  
*Evaluate integrals, if result is known.*
- virtual `ex eval_indexed` (const `basic` &i) const  
*Perform automatic symbolic evaluations on indexed expression that contains this object as the base expression.*

- virtual void `print` (const `print_context` &c, unsigned level=0) const  
*Output to stream.*
- virtual void `dbgprint` () const  
*Little wrapper around print to be called within a debugger.*
- virtual void `dbgprntree` () const  
*Little wrapper around prntree to be called within a debugger.*
- virtual unsigned `precedence` () const  
*Return relative operator precedence (for parenthezing output).*
- virtual bool `info` (unsigned inf) const  
*Information about the object.*
- virtual size\_t `nops` () const  
*Number of operands/members.*
- virtual `ex op` (size\_t i) const  
*Return operand/member at position i.*
- virtual `ex operator[]` (const `ex` &index) const
- virtual `ex operator[]` (size\_t i) const
- virtual `ex & let_op` (size\_t i)  
*Return modifiable operand/member at position i.*
- virtual `ex & operator[]` (const `ex` &index)
- virtual `ex & operator[]` (size\_t i)
- virtual bool `has` (const `ex` &other, unsigned `options`=0) const  
*Test for occurrence of a pattern.*
- virtual `ex subs` (const `exmap` &m, unsigned `options`=0) const  
*Substitute a set of objects by arbitrary expressions.*
- virtual `ex map` (`map_function` &f) const  
*Construct new expression by applying the specified function to all sub-expressions (one level only, not recursively).*
- virtual void `accept` (`GiNaC::visitor` &v) const
- virtual bool `is_polynomial` (const `ex` &var) const  
*Check whether this is a polynomial in the given variables.*
- virtual int `degree` (const `ex` &s) const  
*Return degree of highest power in object s.*
- virtual int `ldegree` (const `ex` &s) const  
*Return degree of lowest power in object s.*
- virtual `ex coeff` (const `ex` &s, int n=1) const  
*Return coefficient of degree n in object s.*
- virtual `ex expand` (unsigned `options`=0) const  
*Expand expression, i.e.*
- virtual `ex collect` (const `ex` &s, bool distributed=false) const  
*Sort expanded expression in terms of powers of some object(s).*
- virtual `ex series` (const `relational` &r, int `order`, unsigned `options`=0) const  
*Default implementation of `ex::series()`.*
- virtual `ex normal` (`exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier) const  
*Default implementation of `ex::normal()`.*
- virtual `ex to_rational` (`exmap` &repl) const  
*Default implementation of `ex::to_rational()`.*
- virtual `ex to_polynomial` (`exmap` &repl) const
- virtual `numeric integer_content` () const
- virtual `ex smod` (const `numeric` &xi) const  
*Apply symmetric modular homomorphism to an expanded multivariate polynomial.*
- virtual `numeric max_coefficient` () const  
*Implementation `ex::max_coefficient()`.*

- virtual `exvector get_free_indices ()` const  
*Return a vector containing the free indices of an expression.*
- virtual `ex add_indexed (const ex &self, const ex &other)` const  
*Add two indexed expressions.*
- virtual `ex scalar_mul_indexed (const ex &self, const numeric &other)` const  
*Multiply an indexed expression with a scalar.*
- virtual `bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v)` const  
*Try to contract two indexed expressions that appear in the same product.*
- virtual `unsigned return_type ()` const
- virtual `return_type_t return_type_tinfo ()` const
- virtual `ex conjugate ()` const
- virtual `ex real_part ()` const
- virtual `ex imag_part ()` const
- template<class T >  
void `print_dispatch (const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- void `print_dispatch (const registered_class_info &ri, const print_context &c, unsigned level)` const  
*Like `print()`, but dispatch to the specified class.*
- `ex subs_one_level (const exmap &m, unsigned options)` const  
*Helper function for `subs()`.*
- `ex diff (const symbol &s, unsigned nth=1)` const  
*Default interface of `nth` derivative `ex::diff(s, n)`.*
- int `compare (const basic &other)` const  
*Compare objects syntactically to establish canonical ordering.*
- bool `is_equal (const basic &other)` const  
*Test for syntactic equality.*
- const `basic & hold ()` const  
*Stop further evaluation.*
- unsigned `gethash ()` const
- const `basic & setflag (unsigned f)` const  
*Set some `status_flags`.*
- const `basic & clearflag (unsigned f)` const  
*Clear some `status_flags`.*

## Public Member Functions inherited from `GiNaC::refcounted`

- `refcounted ()` noexcept
- unsigned int `add_reference ()` noexcept
- unsigned int `remove_reference ()` noexcept
- unsigned int `get_refcount ()` const noexcept
- void `set_refcount (unsigned int r)` noexcept

## Protected Member Functions

- unsigned `calchash ()` const override  
*Compute the hash value of an object and if it makes sense to store it in the objects `status_flags`, do so.*
- void `do_print (const print_context &c, unsigned level)` const
- void `do_print_tree (const print_tree &c, unsigned level)` const
- void `do_print_python_repr (const print_python_repr &c, unsigned level)` const

## Protected Member Functions inherited from [GiNaC::basic](#)

- [basic](#) ()
- virtual [ex eval\\_ncmul](#) (const [exvector](#) &v) const
- virtual bool [match\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if the attributes of two objects are similar enough for a match.*
- virtual [ex derivative](#) (const [symbol](#) &s) const  
*Default implementation of [ex::diff\(\)](#).*
- virtual int [compare\\_same\\_type](#) (const [basic](#) &other) const  
*Returns order relation between two objects of same type.*
- virtual bool [is\\_equal\\_same\\_type](#) (const [basic](#) &other) const  
*Returns true if two objects of same type are equal.*
- void [ensure\\_if\\_modifiable](#) () const  
*Ensure the object may be modified without hurting others, throws if this is not the case.*
- void [do\\_print](#) (const [print\\_context](#) &c, unsigned level) const  
*Default output to stream.*
- void [do\\_print\\_tree](#) (const [print\\_tree](#) &c, unsigned level) const  
*Tree output to stream.*
- void [do\\_print\\_python\\_repr](#) (const [print\\_python\\_repr](#) &c, unsigned level) const  
*Python parsable output to stream.*

## Private Attributes

- unsigned [label](#)  
*Label used to distinguish different wildcards.*

## Additional Inherited Members

## Protected Attributes inherited from [GiNaC::basic](#)

- unsigned [flags](#)  
*of type [status\\_flags](#)*
- unsigned [hashvalue](#)  
*hash value*

### 6.186.1 Detailed Description

This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).

An integer label is used to identify different wildcards.

### 6.186.2 Constructor & Destructor Documentation

#### 6.186.2.1 wildcard()

```
GiNaC::wildcard::wildcard (
 unsigned label)
```

Construct wildcard with specified label.

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), and [GiNaC::basic::setflag\(\)](#).

## 6.186.3 Member Function Documentation

### 6.186.3.1 match()

```
bool GiNaC::wildcard::match (
 const ex & pattern,
 exmap & repl_lst) const [override], [virtual]
```

Check whether the expression matches a given pattern.

For every wildcard object in the pattern, a pair with the wildcard as a key and matching expression as a value is added to *repl\_lst*.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::basic::is\\_equal\(\)](#).

### 6.186.3.2 archive()

```
void GiNaC::wildcard::archive (
 archive_node & n) const [override], [virtual]
```

Save (a.k.a.

serialize) object into archive.

Reimplemented from [GiNaC::basic](#).

References [label](#), and [n](#).

### 6.186.3.3 read\_archive()

```
void GiNaC::wildcard::read_archive (
 const archive_node & n,
 lst & syms) [override], [virtual]
```

Read (a.k.a.

deserialize) object from archive.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::status\\_flags::evaluated](#), [GiNaC::status\\_flags::expanded](#), [label](#), [n](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.4 calchash()

```
unsigned GiNaC::wildcard::calchash () const [override], [protected], [virtual]
```

Compute the hash value of an object and if it makes sense to store it in the objects [status\\_flags](#), do so.

The method inherited from class basic computes a hash value based on the type and hash values of possible members. For this reason it is well suited for container classes but atomic classes should override this implementation because otherwise they would all end up with the same hashvalue.

Reimplemented from [GiNaC::basic](#).

References [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::status\\_flags::hash\\_calculated](#), [GiNaC::basic::hashvalue](#), [label](#), [GiNaC::make\\_hash\\_seed\(\)](#), and [GiNaC::basic::setflag\(\)](#).

### 6.186.3.5 get\_label()

```
unsigned GiNaC::wildcard::get_label () const [inline]
```

References [label](#).

### 6.186.3.6 do\_print()

```
void GiNaC::wildcard::do_print (
 const print_context & c,
 unsigned level) const [protected]
```

References [c](#), and [label](#).

### 6.186.3.7 do\_print\_tree()

```
void GiNaC::wildcard::do_print_tree (
 const print_tree & c,
 unsigned level) const [protected]
```

References [c](#), [GiNaC::basic::flags](#), [GiNaC::basic::hashvalue](#), and [label](#).

### 6.186.3.8 do\_print\_python\_repr()

```
void GiNaC::wildcard::do_print_python_repr (
 const print_python_repr & c,
 unsigned level) const [protected]
```

References [c](#), and [label](#).

## 6.186.4 Member Data Documentation

### 6.186.4.1 label

```
unsigned GiNaC::wildcard::label [private]
```

Label used to distinguish different wildcards.

Referenced by [archive\(\)](#), [calchash\(\)](#), [do\\_print\(\)](#), [do\\_print\\_python\\_repr\(\)](#), [do\\_print\\_tree\(\)](#), [get\\_label\(\)](#), and [read\\_archive\(\)](#).

The documentation for this class was generated from the following files:

- [wildcard.h](#)
- [wildcard.cpp](#)

## 6.187 GiNaC::zeta1\_SERIAL Class Reference

Complex conjugate.

```
#include <inifcns.h>
```

### Static Public Attributes

- static unsigned [serial](#)

### 6.187.1 Detailed Description

Complex conjugate.

Real part. Imaginary part. Absolute value. Step function. Complex sign. Eta function:  $\log(a*b) == \log(a) + \log(b) + \eta(a, b)$ . Sine. Cosine. Tangent. Exponential function. Natural logarithm. Inverse sine (arc sine). Inverse cosine (arc cosine). Inverse tangent (arc tangent). Inverse tangent with two arguments. Hyperbolic Sine. Hyperbolic Cosine. Hyperbolic Tangent. Inverse hyperbolic Sine (area hyperbolic sine). Inverse hyperbolic Cosine (area hyperbolic cosine). Inverse hyperbolic Tangent (area hyperbolic tangent). Dilogarithm. Trilogarithm. Derivatives of Riemann's Zeta-function. Multiple zeta value including Riemann's zeta-function.

### 6.187.2 Member Data Documentation

#### 6.187.2.1 serial

```
unsigned GiNaC::zeta1_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("zeta", 1).
 evalf_func(zetal_evalf).
 eval_func(zetal_eval).
 derivative_func(zetal_deriv).
 print_func<print_latex>(zetal_print_latex).
 do_not_evalf_params().
 overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)

## 6.188 GiNaC::zeta2\_SERIAL Class Reference

Alternating Euler sum or colored MZV.

```
#include <inifcns.h>
```

## Static Public Attributes

- static unsigned [serial](#)

## 6.188.1 Detailed Description

Alternating Euler sum or colored MZV.

## 6.188.2 Member Data Documentation

### 6.188.2.1 `serial`

```
unsigned GiNaC::zeta2_SERIAL::serial [static]
```

#### Initial value:

```
= function::register_new(function_options("zeta", 2).
 evalf_func(zeta2_evalf).
 eval_func(zeta2_eval).
 derivative_func(zeta2_deriv).
 print_func<print_latex>(zeta2_print_latex).
 do_not_evalf_params().
 overloaded(2))
```

Referenced by [GiNaC::zeta\(\)](#).

The documentation for this class was generated from the following files:

- [inifcns.h](#)
- [inifcns\\_nstdsums.cpp](#)



# Chapter 7

## File Documentation

### 7.1 add.cpp File Reference

Implementation of [GiNaC](#)'s sums of expressions.

```
#include "add.h"
#include "mul.h"
#include "archive.h"
#include "operators.h"
#include "matrix.h"
#include "utils.h"
#include "clifford.h"
#include "ncmul.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <string>
```

#### Namespaces

- namespace [GiNaC](#)

#### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (add, expairseq, print\_func< [print\\_context](#) >(&add::do\_print). print\_func< [print\\_latex](#) >(&add::do\_print\_latex). print\_func< [print\\_csrc](#) >(&add::do\_print\_csrc). print\_func< [print\\_tree](#) >(&add::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&add::do\_print\_python\_repr))  
add
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (add)

#### 7.1.1 Detailed Description

Implementation of [GiNaC](#)'s sums of expressions.

## 7.2 add.h File Reference

Interface to [GiNaC](#)'s sums of expressions.

```
#include "expairseq.h"
```

### Classes

- class [GiNaC::add](#)  
*Sum of expressions.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([add](#))

### 7.2.1 Detailed Description

Interface to [GiNaC](#)'s sums of expressions.

## 7.3 archive.cpp File Reference

Archiving of [GiNaC](#) expressions.

```
#include "archive.h"
#include "registrar.h"
#include "ex.h"
#include "lst.h"
#include "version.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

## Functions

- static void [GiNaC::write\\_unsigned](#) (std::ostream &os, unsigned val)  
*Write unsigned integer quantity to stream.*
- static unsigned [GiNaC::read\\_unsigned](#) (std::istream &is)  
*Read unsigned integer quantity from stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [archive\\_node](#) &n)  
*Write [archive\\_node](#) to binary data stream.*
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [archive](#) &ar)  
*Write archive to binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, [archive\\_node](#) &n)  
*Read [archive\\_node](#) from binary data stream.*
- std::istream & [GiNaC::operator>>](#) (std::istream &is, [archive](#) &ar)  
*Read archive from binary data stream.*
- static [synthesize\\_func](#) [GiNaC::find\\_factory\\_fcn](#) (const std::string &name)

### 7.3.1 Detailed Description

Archiving of [GiNaC](#) expressions.

## 7.4 archive.h File Reference

Archiving of [GiNaC](#) expressions.

```
#include "ex.h"
#include <iosfwd>
#include <map>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::archive\\_node](#)  
*This class stores all properties needed to record/retrieve the state of one object of class basic (or a derived class).*
- struct [GiNaC::archive\\_node::property\\_info](#)  
*Information about a stored property.*
- struct [GiNaC::archive\\_node::property](#)  
*Archived property (data type, name and associated data)*
- struct [GiNaC::archive\\_node::archive\\_node\\_cit\\_range](#)
- class [GiNaC::unarchive\\_table\\_t](#)
- class [GiNaC::archive](#)  
*This class holds archived versions of [GiNaC](#) expressions (class [ex](#)).*
- struct [GiNaC::archive::archived\\_ex](#)  
*Archived expression descriptor.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_UNARCHIVER(classname)`  
*Helper macros to register a class with (un)archiving (a.k.a.*
- `#define GINAC_BIND_UNARCHIVER(classname)`

## Typedefs

- `typedef unsigned GiNaC::archive_node_id`  
*Numerical ID value to refer to an [archive\\_node](#).*
- `typedef unsigned GiNaC::archive_atom`  
*Numerical ID value to refer to a string.*
- `typedef basic *(* GiNaC::synthesize_func) ()`
- `typedef std::map< std::string, synthesize_func > GiNaC::unarchive_map_t`

## Functions

- `std::ostream & GiNaC::operator<< (std::ostream &os, const archive &ar)`  
*Write archive to binary data stream.*
- `std::istream & GiNaC::operator>> (std::istream &is, archive &ar)`  
*Read archive from binary data stream.*

## Variables

- static `unarchive_table_t GiNaC::unarch_table_instance`

## 7.4.1 Detailed Description

Archiving of [GiNaC](#) expressions.

## 7.4.2 Macro Definition Documentation

### 7.4.2.1 GINAC\_DECLARE\_UNARCHIVER

```
#define GINAC_DECLARE_UNARCHIVER(
 classname)
```

#### Value:

```
class classname ## _unarchiver
{
 static int usecount;
public:
 static GiNaC::basic* create();
 classname ## _unarchiver();
 ~ classname ## _unarchiver();
};
static classname ## _unarchiver classname ## _unarchiver_instance
```

Helper macros to register a class with (un)archiving (a.k.a.

(de)serialization).

Usage: put

`GINAC_DECLARE_UNARCHIVER(myclass);`

into the header file (in the global or namespace scope), and

`GINAC_BIND_UNARCHIVER(myclass);`

into the source file.

Effect: the 'myclass' (being a class derived directly or indirectly from `GiNaC::basic`) can be archived and unarchived.

Note: you need to use `GINAC_{DECLARE,BIND}_UNARCHIVER` incantations in order to make your class (un)archivable *even if your class does not overload 'read\_archive' method*. Sorry for inconvenience.

How it works:

The 'basic' class has a 'read\_archive' virtual method which reads an expression from archive. Derived classes can overload that method. There's a small problem, though. On unarchiving all we have is a set of named byte streams. In C++ the class name (as written in the source code) has nothing to do with its actual type. Thus, we need establish a correspondence ourselves. To do so we maintain a 'class\_name' => 'function\_pointer' table (see the `unarchive_table_t` class above). Every function in this table is supposed to create a new object of the 'class\_name' type. The 'archive\_node' class uses that table to construct an object of correct type. Next it invokes `read_archive` virtual method of newly created object, which does the actual job.

Note: this approach is very simple-minded (it does not handle classes with same names from different namespaces, multiple inheritance, etc), but it happens to work surprisingly well.

#### 7.4.2.2 GINAC\_BIND\_UNARCHIVER

```
#define GINAC_BIND_UNARCHIVER(
 classname)
```

**Value:**

```
classname ## _unarchiver::classname ## _unarchiver() \
{ \
 static GiNaC::unarchive_table_t table; \
 if (usecount++ == 0) { \
 table.insert(std::string(#classname), \
 &(classname ## _unarchiver::create)); \
 } \
} \
GiNaC::basic* classname ## _unarchiver::create() \
{ \
 return new classname(); \
} \
classname ## _unarchiver::~~ classname ## _unarchiver() { } \
int classname ## _unarchiver::usecount = 0
```

## 7.5 assertion.h File Reference

Assertion macro definition.

### Macros

- `#define GINAC_ASSERT(X) ((void)0)`  
Assertion macro for checking invariances.

## 7.5.1 Detailed Description

Assertion macro definition.

## 7.5.2 Macro Definition Documentation

### 7.5.2.1 GINAC\_ASSERT

```
#define GINAC_ASSERT(
 X) ((void)0)
```

Assertion macro for checking invariances.

Referenced by [GiNaC::acos\\_deriv\(\)](#), [GiNaC::acosh\\_deriv\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::symmetry::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::add::add\(\)](#), [GiNaC::remember\\_table\\_list::add\\_entry\(\)](#), [GiNaC::remember\\_table::add\\_entry\(\)](#), [GiNaC::matrix::add\\_indexed\(\)](#), [GiNaC::algebraic\\_match\\_mul\\_with\\_mul\(\)](#), [GiNaC::function::archive\(\)](#), [GiNaC::asin\\_deriv\(\)](#), [GiNaC::asinh\\_deriv\(\)](#), [GiNaC::atan2\\_deriv\(\)](#), [GiNaC::atan\\_deriv\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_deriv\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_deriv\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::mul::can\\_make\\_flat\(\)](#), [GiNaC::clifford::clifford\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::expairseq::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_c](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_v](#), [GiNaC::function::conjugate\(\)](#), [GiNaC::ex::construct\\_from\\_basic\(\)](#), [GiNaC::su3t::contract\\_with\(\)](#), [GiNaC::su3f::contract\\_with\(\)](#), [GiNaC::su3d::contract\\_with\(\)](#), [GiNaC::matrix::contract\\_with\(\)](#), [GiNaC::tensdelta::contract\\_with\(\)](#), [GiNaC::tensmetric::contract\\_with\(\)](#), [GiNaC::spinmetric::contract\\_with\(\)](#), [GiNaC::tensepsilon::contract\\_with\(\)](#), [GiNaC::cos\\_deriv\(\)](#), [GiNaC::cosh\\_deriv\(\)](#), [GiNaC::ex::denom\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::add::eval\(\)](#), [GiNaC::function::eval\(\)](#), [GiNaC::indexed::eval\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::su3f::eval\\_indexed\(\)](#), [GiNaC::su3d::eval\\_indexed\(\)](#), [GiNaC::matrix::eval\\_indexed\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::minkmetric::eval\\_indexed\(\)](#), [GiNaC::spinmetric::eval\\_indexed\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::function::evalf\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::ex::ex\(\)](#), [GiNaC::exp\\_deriv\(\)](#), [GiNaC::expair::expair\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::expairseq::expairseq\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::function::expl\\_derivative\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination](#), [GiNaC::gcd\\_pf\\_mul\(\)](#), [GiNaC::gcd\\_pf\\_pow\(\)](#), [GiNaC::indexed::get\\_indices\(\)](#), [GiNaC::function::get\\_name\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::function::imag\\_part\(\)](#), [GiNaC::function::info\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::mul::integer\\_content\(\)](#), [GiNaC::expair::is\\_canonical\\_numeric\(\)](#), [GiNaC::remember\\_table\\_entry::is\\_equal\(\)](#), [GiNaC::constant::is\\_equal\\_same\\_type\(\)](#), [GiNaC::container< C >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::fderivative::is\\_equal\\_same\\_type\(\)](#), [GiNaC::function::is\\_equal\\_same\\_type\(\)](#), [GiNaC::numeric::is\\_equal\\_same\\_type\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::is\\_equal\\_same\\_type\(\)](#), [GiNaC::symbol::is\\_equal\\_same\\_type\(\)](#), [GiNaC::clifford::let\\_op\(\)](#), [GiNaC::container< C >::let\\_op\(\)](#), [GiNaC::matrix::let\\_op\(\)](#), [GiNaC::lgamma\\_deriv\(\)](#), [GiNaC::Li2\\_deriv\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::log\\_deriv\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::remember\\_table::lookup\\_entry\(\)](#), [GiNaC::Isolve\(\)](#), [GiNaC::ex::makewriteable\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::clifford::match\\_same\\_type\(\)](#), [GiNaC::color::match\\_same\\_type\(\)](#), [GiNaC::fderivative::match\\_same\\_type\(\)](#), [GiNaC::function::match\\_same\\_type\(\)](#), [GiNaC::idx::match\\_same\\_type\(\)](#), [GiNaC::varidx::match\\_same\\_type\(\)](#), [GiNaC::spinidx::match\\_same\\_type\(\)](#), [GiNaC::matrix::match\\_same\\_type\(\)](#), [GiNaC::relational::match\\_same\\_type\(\)](#), [GiNaC::add::max\\_coefficient\(\)](#), [GiNaC::mul::max\\_coefficient\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::ex::normal\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::ex::numer\(\)](#), [GiNaC::ex::numer\\_denom\(\)](#), [GiNaC::clifford::op\(\)](#), [GiNaC::container< C >::op\(\)](#), [GiNaC::expairseq::op\(\)](#), [GiNaC::idx::op\(\)](#), [GiNaC::integral::op\(\)](#), [GiNaC::matrix::op\(\)](#), [GiNaC::power::op\(\)](#), [GiNaC::relational::op\(\)](#), [GiNaC::sy\\_swap::operator\(\)](#), [GiNaC::sy\\_less::operator\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::function::pderivative\(\)](#), [GiNaC::permute\\_free\\_index\\_to\\_front\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::function::power\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::product\\_to\\_exvector\(\)](#), [GiNaC::pseries::pseries\(\)](#), [GiNaC::psi1\\_deriv\(\)](#), [GiNaC::psi2\\_deriv\(\)](#), [GiNaC::ptr< T >::ptr\(\)](#), [GiNaC::matrix::rank\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::function::real\\_part\(\)](#), [GiNaC::rename\\_dummy\\_indices\(\)](#), [GiNaC::function::return\\_type\(\)](#), [GiNaC::ncmul::return\\_type\(\)](#), [GiNaC::relational::return\\_type\(\)](#),

[GiNaC::function::return\\_type\\_tinfo\(\)](#), [GiNaC::relational::return\\_type\\_tinfo\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::matrix::scalar\\_mul\\_indexed\(\)](#),  
[GiNaC::function::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::sin\\_deriv\(\)](#),  
[GiNaC::sinh\\_deriv\(\)](#), [GiNaC::add::smod\(\)](#), [GiNaC::mul::smod\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::ex::subs\(\)](#),  
[GiNaC::ex::subs\(\)](#), [GiNaC::ex::swap\(\)](#), [GiNaC::tan\\_deriv\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_deriv\(\)](#), [GiNaC::tanh\\_series\(\)](#),  
[GiNaC::tgamma\\_deriv\(\)](#), [GiNaC::numeric::to\\_double\(\)](#), [GiNaC::numeric::to\\_int\(\)](#), [GiNaC::numeric::to\\_long\(\)](#),  
[GiNaC::indexed::validate\(\)](#), [GiNaC::zeta1\\_deriv\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::basic::~~basic\(\)](#).

## 7.6 basic.cpp File Reference

Implementation of [GiNaC](#)'s ABC.

```

#include "basic.h"
#include "ex.h"
#include "numeric.h"
#include "power.h"
#include "add.h"
#include "symbol.h"
#include "lst.h"
#include "ncmul.h"
#include "relational.h"
#include "operators.h"
#include "wildcard.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <typeinfo>

```

### Classes

- struct [GiNaC::evalf\\_map\\_function](#)  
*Function object to be applied by [basic::evalf\(\)](#).*
- struct [GiNaC::evalm\\_map\\_function](#)  
*Function object to be applied by [basic::evalm\(\)](#).*
- struct [GiNaC::eval\\_integ\\_map\\_function](#)  
*Function object to be applied by [basic::eval\\_integ\(\)](#).*
- struct [GiNaC::derivative\\_map\\_function](#)  
*Function object to be applied by [basic::derivative\(\)](#).*
- struct [GiNaC::expand\\_map\\_function](#)  
*Function object to be applied by [basic::expand\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([basic](#), void, [print\\_func](#)< [print\\_context](#)>(&[basic::do\\_print](#)), [print\\_func](#)< [print\\_tree](#)>(&[basic::do\\_print\\_tree](#)), [print\\_func](#)< [print\\_python\\_repr](#)>(&[basic::do\\_print\\_python\\_repr](#))) [basic](#)

*basic copy constructor: implicitly assumes that the other class is of the exact same type (as it's used by [duplicate\(\)](#)), so it can copy the [tinfo\\_key](#) and the hash value.*

## Variables

- [GiNaC::evalm\\_map\\_function](#) [GiNaC::map\\_evalm](#)
- [GiNaC::eval\\_integ\\_map\\_function](#) [GiNaC::map\\_eval\\_integ](#)

### 7.6.1 Detailed Description

Implementation of [GiNaC](#)'s ABC.

## 7.7 basic.h File Reference

Interface to [GiNaC](#)'s ABC.

```
#include "flags.h"
#include "ptr.h"
#include "assertion.h"
#include "registrar.h"
#include <cstdint>
#include <map>
#include <set>
#include <typeinfo>
#include <vector>
#include <utility>
```

## Classes

- struct [GiNaC::map\\_function](#)  
*Function object for [map\(\)](#).*
- class [GiNaC::visitor](#)  
*Degenerate base class for visitors.*
- class [GiNaC::basic](#)  
*This class is the ABC (abstract base class) of [GiNaC](#)'s class hierarchy.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< [ex](#) > [GiNaC::exvector](#)
- typedef std::set< [ex](#), [ex\\_is\\_less](#) > [GiNaC::exset](#)
- typedef std::map< [ex](#), [ex](#), [ex\\_is\\_less](#) > [GiNaC::exmap](#)

## Functions

- template<class T >  
bool [GiNaC::is\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, including base classes.*
- template<class T >  
bool [GiNaC::is\\_exactly\\_a](#) (const [basic](#) &obj)  
*Check if obj is a T, not including base classes.*
- template<class B , typename... Args>  
B & [GiNaC::dynallocate](#) (Args &&... args)  
*Constructs a new (class basic or derived) B object on the heap.*
- template<class B >  
B & [GiNaC::dynallocate](#) (std::initializer\_list< [ex](#) > il)  
*Constructs a new (class basic or derived) B object on the heap.*

### 7.7.1 Detailed Description

Interface to [GiNaC](#)'s ABC.

## 7.8 class\_info.h File Reference

Helper templates to provide per-class information for class hierarchies.

```
#include <cstddef>
#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <stdexcept>
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::class\\_info](#)< OPT >
- struct [GiNaC::class\\_info](#)< OPT >::tree\_node

## Namespaces

- namespace [GiNaC](#)

### 7.8.1 Detailed Description

Helper templates to provide per-class information for class hierarchies.

## 7.9 clifford.cpp File Reference

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "clifford.h"
#include "ex.h"
#include "idx.h"
#include "ncmul.h"
#include "symbol.h"
#include "numeric.h"
#include "symmetry.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <stdexcept>
```

### Classes

- struct [GiNaC::is\\_not\\_a\\_clifford](#)  
*Predicate for finding non-clifford objects.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([clifford](#), [indexed](#), [print\\_func< print\\_dflt >\(&clifford::do\\_print\\_dflt\)](#), [print\\_func< print\\_latex >\(&clifford::do\\_print\\_latex\)](#), [print\\_func< print\\_tree >\(&clifford::do\\_print\\_tree\)](#)) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\(diracone](#)
- [GiNaC::print\\_func< print\\_dflt > \(&diracone::do\\_print\)](#), [print\\_func< print\\_latex >\(&diracone](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([clifford](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([cliffordunit](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([diracone](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([diracgamma5](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaL](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([diracgammaR](#))
- static bool [GiNaC::is\\_dirac\\_slash](#) (const [ex](#) &seq0)
- static void [GiNaC::base\\_and\\_index](#) (const [ex](#) &c, [ex](#) &b, [ex](#) &i)

- This function decomposes  $\gamma \sim \mu \rightarrow (1, \mu)$  and  $a \rightarrow (a.ix, ix)$*
- `ex GiNaC::dirac_ONE` (unsigned char rl=0)  
*Create a Clifford unity object.*
  - static unsigned `GiNaC::get_dim_uint` (const `ex` &e)
  - `ex GiNaC::clifford_unit` (const `ex` &mu, const `ex` &metr, unsigned char rl=0)  
*Create a Clifford unit object.*
  - `ex GiNaC::dirac_gamma` (const `ex` &mu, unsigned char rl=0)  
*Create a Dirac gamma object.*
  - `ex GiNaC::dirac_gamma5` (unsigned char rl=0)  
*Create a Dirac gamma5 object.*
  - `ex GiNaC::dirac_gammaL` (unsigned char rl=0)  
*Create a Dirac gammaL object.*
  - `ex GiNaC::dirac_gammaR` (unsigned char rl=0)  
*Create a Dirac gammaR object.*
  - `ex GiNaC::dirac_slash` (const `ex` &e, const `ex` &dim, unsigned char rl=0)  
*Create a term of the form  $e_\mu * \gamma \sim \mu$  with a unique index  $\mu$ .*
  - static unsigned char `GiNaC::get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
  - static `ex GiNaC::trace_string` (exvector::const\_iterator ix, size\_t num)  
*Take trace of a string of an even number of Dirac gammas given a vector of indices.*
  - `ex GiNaC::dirac_trace` (const `ex` &e, const std::set< unsigned char > &rls, const `ex` &trONE=4)  
*Calculate dirac traces over the specified set of representation labels.*
  - `ex GiNaC::dirac_trace` (const `ex` &e, const `lst` &rl, const `ex` &trONE=4)  
*Calculate dirac traces over the specified list of representation labels.*
  - `ex GiNaC::dirac_trace` (const `ex` &e, unsigned char rl=0, const `ex` &trONE=4)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
  - `ex GiNaC::canonicalize_clifford` (const `ex` &e)  
*Bring all products of clifford objects in an expression into a canonical order.*
  - `ex GiNaC::clifford_star_bar` (const `ex` &e, bool do\_bar, unsigned `options`)  
*An auxillary function performing `clifford_star()` and `clifford_bar()`.*
  - `ex GiNaC::clifford_prime` (const `ex` &e)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
  - `ex GiNaC::remove_dirac_ONE` (const `ex` &e, unsigned char rl=0, unsigned `options`=0)  
*Replaces `dirac_ONE`'s (with a `representation_label` no less than `rl`) in `e` with 1.*
  - int `GiNaC::clifford_max_label` (const `ex` &e, bool ignore\_ONE=false)  
*Returns the maximal representation label of a clifford object if `e` contains at least one, otherwise returns -1.*
  - `ex GiNaC::clifford_norm` (const `ex` &e)  
*Calculation of the norm in the Clifford algebra.*
  - `ex GiNaC::clifford_inverse` (const `ex` &e)  
*Calculation of the inverse in the Clifford algebra.*
  - `ex GiNaC::lst_to_clifford` (const `ex` &v, const `ex` &mu, const `ex` &metr, unsigned char rl=0)  
*List or vector conversion into the Clifford vector.*
  - `ex GiNaC::lst_to_clifford` (const `ex` &v, const `ex` &e)  
*List or vector conversion into the Clifford vector.*
  - static `ex GiNaC::get_clifford_comp` (const `ex` &e, const `ex` &c, bool root=true)  
*Auxiliary structure to define a function for stripping one Clifford unit from vectors.*
  - `lst GiNaC::clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)  
*An inverse function to `lst_to_clifford()`.*
  - `ex GiNaC::clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)

*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*

- [ex GiNaC::clifford\\_moebius\\_map](#) (const [ex](#) &M, const [ex](#) &v, const [ex](#) &G, unsigned char rl=0)

*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

## Variables

- [GiNaC::tensor](#)

## 7.9.1 Detailed Description

Implementation of [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

## 7.10 clifford.h File Reference

Interface to [GiNaC](#)'s clifford algebra (Dirac gamma) objects.

```
#include "indexed.h"
#include "tensor.h"
#include "symbol.h"
#include "idx.h"
#include <set>
```

## Classes

- class [GiNaC::clifford](#)  
*This class holds an object representing an element of the Clifford algebra (the Dirac gamma matrices).*
- class [GiNaC::diracone](#)  
*This class represents the Clifford algebra unity element.*
- class [GiNaC::cliffordunit](#)  
*This class represents the Clifford algebra generators (units).*
- class [GiNaC::diracgamma](#)  
*This class represents the Dirac gamma Lorentz vector.*
- class [GiNaC::diracgamma5](#)  
*This class represents the Dirac gamma5 object which anticommutes with all other gammas.*
- class [GiNaC::diracgammaL](#)  
*This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).*
- class [GiNaC::diracgammaR](#)  
*This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`clifford`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`diracone`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`cliffordunit`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`diracgamma`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`diracgamma5`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`diracgammaL`)
- `GiNaC::GINAC_DECLARE_UNARCHIVER` (`diracgammaR`)
- `bool GiNaC::is_clifford_tinfo` (`const return_type_t &ti`)  
*Check whether a given `return_type_t` object (as returned by `return_type_tinfo()`) is that of a clifford object (with an arbitrary representation label).*
- `ex GiNaC::dirac_ONE` (`unsigned char rl=0`)  
*Create a Clifford unity object.*
- `ex GiNaC::clifford_unit` (`const ex &mu`, `const ex &metr`, `unsigned char rl=0`)  
*Create a Clifford unit object.*
- `ex GiNaC::dirac_gamma` (`const ex &mu`, `unsigned char rl=0`)  
*Create a Dirac gamma object.*
- `ex GiNaC::dirac_gamma5` (`unsigned char rl=0`)  
*Create a Dirac gamma5 object.*
- `ex GiNaC::dirac_gammaL` (`unsigned char rl=0`)  
*Create a Dirac gammaL object.*
- `ex GiNaC::dirac_gammaR` (`unsigned char rl=0`)  
*Create a Dirac gammaR object.*
- `ex GiNaC::dirac_slash` (`const ex &e`, `const ex &dim`, `unsigned char rl=0`)  
*Create a term of the form  $e_{\mu} * \gamma^{\sim\mu}$  with a unique index  $\mu$ .*
- `ex GiNaC::dirac_trace` (`const ex &e`, `const std::set< unsigned char > &rls`, `const ex &trONE=4`)  
*Calculate dirac traces over the specified set of representation labels.*
- `ex GiNaC::dirac_trace` (`const ex &e`, `const lst &rls`, `const ex &trONE=4`)  
*Calculate dirac traces over the specified list of representation labels.*
- `ex GiNaC::dirac_trace` (`const ex &e`, `unsigned char rl=0`, `const ex &trONE=4`)  
*Calculate the trace of an expression containing gamma objects with a specified representation label.*
- `ex GiNaC::canonicalize_clifford` (`const ex &e`)  
*Bring all products of clifford objects in an expression into a canonical order.*
- `ex GiNaC::clifford_prime` (`const ex &e`)  
*Automorphism of the Clifford algebra, simply changes signs of all clifford units.*
- `ex GiNaC::clifford_star_bar` (`const ex &e`, `bool do_bar`, `unsigned options`)  
*An auxillary function performing `clifford_star()` and `clifford_bar()`.*
- `ex GiNaC::clifford_bar` (`const ex &e`)  
*Main anti-automorphism of the Clifford algebra: makes reversion and changes signs of all clifford units.*
- `ex GiNaC::clifford_star` (`const ex &e`)  
*Reversion of the Clifford algebra, reverse the order of all clifford units in `ncmul`.*
- `ex GiNaC::remove_dirac_ONE` (`const ex &e`, `unsigned char rl=0`, `unsigned options=0`)  
*Replaces `dirac_ONE`'s (with a representation\_label no less than `rl`) in `e` with 1.*
- `int GiNaC::clifford_max_label` (`const ex &e`, `bool ignore_ONE=false`)  
*Returns the maximal representation label of a clifford object if `e` contains at least one, otherwise returns -1.*
- `ex GiNaC::clifford_norm` (`const ex &e`)  
*Calculation of the norm in the Clifford algebra.*
- `ex GiNaC::clifford_inverse` (`const ex &e`)  
*Calculation of the inverse in the Clifford algebra.*
- `ex GiNaC::lst_to_clifford` (`const ex &v`, `const ex &mu`, `const ex &metr`, `unsigned char rl=0`)  
*List or vector conversion into the Clifford vector.*

- `ex GiNaC::lst_to_clifford` (const `ex` &v, const `ex` &e)  
*List or vector conversion into the Clifford vector.*
- `lst GiNaC::clifford_to_lst` (const `ex` &e, const `ex` &c, bool algebraic=true)  
*An inverse function to `lst_to_clifford()`.*
- `ex GiNaC::clifford_moebius_map` (const `ex` &a, const `ex` &b, const `ex` &c, const `ex` &d, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*Calculations of Moebius transformations (conformal map) defined by a 2x2 Clifford matrix (a b|c d) in linear spaces with arbitrary signature.*
- `ex GiNaC::clifford_moebius_map` (const `ex` &M, const `ex` &v, const `ex` &G, unsigned char rl=0)  
*The second form of Moebius transformations defined by a 2x2 Clifford matrix M This function takes the transformation matrix M as a single entity.*

### 7.10.1 Detailed Description

Interface to `GiNaC`'s clifford algebra (Dirac gamma) objects.

## 7.11 color.cpp File Reference

Implementation of `GiNaC`'s color (SU(3) Lie algebra) objects.

```
#include "color.h"
#include "idx.h"
#include "ncmul.h"
#include "symmetry.h"
#include "operators.h"
#include "numeric.h"
#include "mul.h"
#include "power.h"
#include "symbol.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace `GiNaC`

### Macros

- `#define TEST_PERMUTATION(A, B, C, P)`
- `#define CMPINDICES(A, B, C) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))`

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`su3one`, `tensor`, `print_func< print_dflt >(&su3one::do_print)`). `print_func< print_latex >(&su3one::do_print_latex)`) `GINAC_IMPLEMENT_↵`  
`REGISTERED_CLASS_OPT(su3t`
- `GiNaC::print_func< print_dflt > (&su3t::do_print)`. `print_func< print_latex >(&su3t`
- `GiNaC::GINAC_BIND_UNARCHIVER` (`color`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3one`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3t`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3f`)
- `GiNaC::GINAC_BIND_UNARCHIVER` (`su3d`)
- static `ex` `GiNaC::permute_free_index_to_front` (const `exvector` &iv3, const `exvector` &iv2, int &sig)  
*Given a vector iv3 of three indices and a vector iv2 of two indices that is a subset of iv3, return the (free) index that is in iv3 but not in iv2 and the sign introduced by permuting that index to the front.*
- `ex` `GiNaC::color_ONE` (unsigned char rl=0)  
*Create the su(3) unity element.*
- `ex` `GiNaC::color_T` (const `ex` &a, unsigned char rl=0)  
*Create an su(3) generator.*
- `ex` `GiNaC::color_f` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) antisymmetric structure constant.*
- `ex` `GiNaC::color_d` (const `ex` &a, const `ex` &b, const `ex` &c)  
*Create an su(3) symmetric structure constant.*
- `ex` `GiNaC::color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination d.a.b.c+l\*f.a.b.c.*
- static bool `GiNaC::is_color_tinfo` (const `return_type_t` &ti)  
*Check whether a given tinfo key (as returned by `return_type_tinfo()`) is that of a color object (with an arbitrary representation label).*
- static unsigned char `GiNaC::get_representation_label` (const `return_type_t` &ti)  
*Extract representation label from tinfo key (as returned by `return_type_tinfo()`).*
- `ex` `GiNaC::color_trace` (const `ex` &e, const `std::set< unsigned char >` &rls)  
*Calculate color traces over the specified set of representation labels.*
- `ex` `GiNaC::color_trace` (const `ex` &e, const `lst` &rl)  
*Calculate color traces over the specified list of representation labels.*
- `ex` `GiNaC::color_trace` (const `ex` &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.11.1 Detailed Description

Implementation of `GiNaC`'s color (SU(3) Lie algebra) objects.

### 7.11.2 Macro Definition Documentation

#### 7.11.2.1 TEST\_PERMUTATION

```
#define TEST_PERMUTATION(
 A,
 B,
 C,
 P)
```

#### Value:

```
if (iv3[B].is_equal(iv2[0]) && iv3[C].is_equal(iv2[1])) { \
 sig = P; \
 return iv3[A]; \
}
```

### 7.11.2.2 CMPINDICES

```
#define CMPINDICES(
 A,
 B,
 C) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))
```

## 7.12 color.h File Reference

Interface to [GiNaC](#)'s color (SU(3) Lie algebra) objects.

```
#include "indexed.h"
#include "tensor.h"
#include <set>
```

### Classes

- class [GiNaC::color](#)  
*This class holds a generator  $T_a$  or the unity element of the Lie algebra of SU(3), as used for calculations in quantum chromodynamics.*
- class [GiNaC::su3one](#)  
*This class represents the su(3) unity element.*
- class [GiNaC::su3t](#)  
*This class represents an su(3) generator.*
- class [GiNaC::su3f](#)  
*This class represents the tensor of antisymmetric su(3) structure constants.*
- class [GiNaC::su3d](#)  
*This class represents the tensor of symmetric su(3) structure constants.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([color](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3one](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3t](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3f](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([su3d](#))
- [ex GiNaC::color\\_ONE](#) (unsigned char rl=0)  
*Create the su(3) unity element.*
- [ex GiNaC::color\\_T](#) (const [ex](#) &a, unsigned char rl=0)  
*Create an su(3) generator.*
- [ex GiNaC::color\\_f](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c)  
*Create an su(3) antisymmetric structure constant.*
- [ex GiNaC::color\\_d](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &c)  
*Create an su(3) symmetric structure constant.*

- `ex GiNaC::color_h` (const `ex` &a, const `ex` &b, const `ex` &c)  
*This returns the linear combination  $d.a.b.c + l*f.a.b.c$ .*
- `ex GiNaC::color_trace` (const `ex` &e, const `std::set< unsigned char >` &rls)  
*Calculate color traces over the specified set of representation labels.*
- `ex GiNaC::color_trace` (const `ex` &e, const `lst` &rl)  
*Calculate color traces over the specified list of representation labels.*
- `ex GiNaC::color_trace` (const `ex` &e, unsigned char rl=0)  
*Calculate the trace of an expression containing color objects with a specified representation label.*

### 7.12.1 Detailed Description

Interface to `GiNaC`'s color (SU(3) Lie algebra) objects.

## 7.13 compiler.h File Reference

Definition of optimizing macros.

### Macros

- `#define unlikely`(cond) (cond)
- `#define likely`(cond) (cond)
- `#define attribute_deprecated`

### 7.13.1 Detailed Description

Definition of optimizing macros.

### 7.13.2 Macro Definition Documentation

#### 7.13.2.1 unlikely

```
#define unlikely(
 cond) (cond)
```

Referenced by `GiNaC::add::eval()`, `GiNaC::mul::eval()`, and `GiNaC::expairseq::evalchildren()`.

#### 7.13.2.2 likely

```
#define likely(
 cond) (cond)
```

Referenced by `GiNaC::add::combine_ex_with_coeff_to_pair()`, `GiNaC::mul::eval()`, and `GiNaC::power::eval()`.

### 7.13.2.3 `attribute_deprecated`

```
#define attribute_deprecated
```

## 7.14 `constant.cpp` File Reference

Implementation of [GiNaC](#)'s constant types and some special constants.

```
#include "constant.h"
#include "numeric.h"
#include "ex.h"
#include "archive.h"
#include "utils.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([constant](#), [basic](#), [print\\_func< print\\_context >\(&constant::do\\_print\)](#), [print\\_func< print\\_latex >\(&constant::do\\_print\\_latex\)](#), [print\\_func< print\\_tree >\(&constant::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&constant::do\\_print\\_python\\_repr\)](#)) `constant`
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([constant](#))

### Variables

- `const` [constant](#) [GiNaC::Pi](#) ("Pi", [PiEvalf](#), "\\pi", [domain::positive](#))  
*Pi.*
- `const` [constant](#) [GiNaC::Euler](#) ("Euler", [EulerEvalf](#), "\\gamma\_E", [domain::positive](#))  
*Euler's constant.*
- `const` [constant](#) [GiNaC::Catalan](#) ("Catalan", [CatalanEvalf](#), "G", [domain::positive](#))  
*Catalan's constant.*

### 7.14.1 Detailed Description

Implementation of [GiNaC](#)'s constant types and some special constants.

## 7.15 constant.h File Reference

Interface to [GiNaC](#)'s constant types and some special constants.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <string>
```

### Classes

- class [GiNaC::constant](#)  
*This class holds constants, symbols with specific numerical value.*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef [ex](#)(\* [GiNaC::evalffunctype](#)) ()

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([constant](#))

### 7.15.1 Detailed Description

Interface to [GiNaC](#)'s constant types and some special constants.

## 7.16 container.h File Reference

Wrapper template for making [GiNaC](#) classes out of STL containers.

```
#include "ex.h"
#include "print.h"
#include "archive.h"
#include "assertion.h"
#include "compiler.h"
#include <algorithm>
#include <iterator>
#include <list>
#include <memory>
#include <stdexcept>
#include <vector>
```

## Classes

- class [GiNaC::container\\_storage< C >](#)  
*Helper template for encapsulating the [reserve\(\)](#) mechanics of STL containers.*
- class [GiNaC::container< C >](#)  
*Wrapper template for making [GiNaC](#) classes out of STL containers.*

## Namespaces

- namespace [GiNaC](#)

### 7.16.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of STL containers.

## 7.17 crc32.h File Reference

CRC32 hash function.

## Namespaces

- namespace [GiNaC](#)

## Functions

- static unsigned [GiNaC::crc32](#) (const char \*[c](#), const unsigned [len](#), const unsigned crcinit)

## Variables

- static unsigned const [GiNaC::crctab](#) [256]

### 7.17.1 Detailed Description

CRC32 hash function.

Shamelessly stolen from GNU coreutils (cksum.c).

## 7.18 ex.cpp File Reference

Implementation of [GiNaC](#)'s light-weight expression handles.

```
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "matrix.h"
#include "power.h"
#include "lst.h"
#include "relational.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### 7.18.1 Detailed Description

Implementation of [GiNaC](#)'s light-weight expression handles.

## 7.19 ex.h File Reference

Interface to [GiNaC](#)'s light-weight expression handles.

```
#include "basic.h"
#include "ptr.h"
#include <functional>
#include <iosfwd>
#include <iterator>
#include <memory>
#include <stack>
```

### Classes

- class [GiNaC::library\\_init](#)  
*Helper class to initialize the library.*
- class [GiNaC::ex](#)  
*Lightweight wrapper for [GiNaC](#)'s symbolic objects.*
- class [GiNaC::const\\_iterator](#)
- struct [GiNaC::internal::\\_iter\\_rep](#)
- class [GiNaC::const\\_preorder\\_iterator](#)
- class [GiNaC::const\\_postorder\\_iterator](#)
- struct [GiNaC::ex\\_is\\_less](#)

- struct [GiNaC::ex\\_is\\_equal](#)
- struct [GiNaC::op0\\_is\\_equal](#)
- struct [GiNaC::ex\\_swap](#)
- class [GiNaC::pointer\\_to\\_map\\_function](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_1arg< T1 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_2args< T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_map\\_function\\_3args< T1, T2, T3 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function< C >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_1arg< C, T1 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_2args< C, T1, T2 >](#)
- class [GiNaC::pointer\\_to\\_member\\_to\\_map\\_function\\_3args< C, T1, T2, T3 >](#)
- struct [std::hash< GiNaC::ex >](#)

*Specialization of `std::hash()` for `ex` objects.*

- struct [std::equal\\_to< GiNaC::ex >](#)

*Specialization of `std::equal_to()` for `ex` objects.*

## Namespaces

- namespace [GiNaC](#)
- namespace [GiNaC::internal](#)
- namespace [std](#)

## Functions

- bool [GiNaC::are\\_ex\\_trivially\\_equal](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Compare two objects of class quickly without doing a deep tree traversal.*
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exvector](#) &e)
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exset](#) &e)
- [std::ostream & GiNaC::operator<<](#) (std::ostream &os, const [exmap](#) &e)
- [size\\_t GiNaC::nops](#) (const [ex](#) &thisex)
- [ex GiNaC::expand](#) (const [ex](#) &thisex, unsigned [options](#)=0)
- [ex GiNaC::conjugate](#) (const [ex](#) &thisex)
- [ex GiNaC::real\\_part](#) (const [ex](#) &thisex)
- [ex GiNaC::imag\\_part](#) (const [ex](#) &thisex)
- bool [GiNaC::has](#) (const [ex](#) &thisex, const [ex](#) &pattern, unsigned [options](#)=0)
- bool [GiNaC::find](#) (const [ex](#) &thisex, const [ex](#) &pattern, [exset](#) &found)
- bool [GiNaC::is\\_polynomial](#) (const [ex](#) &thisex, const [ex](#) &vars)
- int [GiNaC::degree](#) (const [ex](#) &thisex, const [ex](#) &s)
- int [GiNaC::ldegree](#) (const [ex](#) &thisex, const [ex](#) &s)
- [ex GiNaC::coeff](#) (const [ex](#) &thisex, const [ex](#) &s, int [n](#)=1)
- [ex GiNaC::numer](#) (const [ex](#) &thisex)
- [ex GiNaC::denom](#) (const [ex](#) &thisex)
- [ex GiNaC::numer\\_denom](#) (const [ex](#) &thisex)
- [ex GiNaC::normal](#) (const [ex](#) &thisex)
- [ex GiNaC::to\\_rational](#) (const [ex](#) &thisex, [exmap](#) &repl)
- [ex GiNaC::to\\_polynomial](#) (const [ex](#) &thisex, [exmap](#) &repl)
- [ex GiNaC::collect](#) (const [ex](#) &thisex, const [ex](#) &s, bool [distributed](#)=false)
- [ex GiNaC::eval](#) (const [ex](#) &thisex)
- [ex GiNaC::evalf](#) (const [ex](#) &thisex)
- [ex GiNaC::evalm](#) (const [ex](#) &thisex)
- [ex GiNaC::eval\\_integ](#) (const [ex](#) &thisex)
- [ex GiNaC::diff](#) (const [ex](#) &thisex, const [symbol](#) &s, unsigned [nth](#)=1)

- `ex GiNaC::series` (const `ex` &thisex, const `ex` &r, int `order`, unsigned `options`=0)
- `bool GiNaC::match` (const `ex` &thisex, const `ex` &pattern, `exmap` &repl\_lst)
- `ex GiNaC::simplify_indexed` (const `ex` &thisex, unsigned `options`=0)
- `ex GiNaC::simplify_indexed` (const `ex` &thisex, const `scalar_products` &sp, unsigned `options`=0)
- `ex GiNaC::symmetrize` (const `ex` &thisex)
- `ex GiNaC::symmetrize` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::antisymmetrize` (const `ex` &thisex)
- `ex GiNaC::antisymmetrize` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::symmetrize_cyclic` (const `ex` &thisex)
- `ex GiNaC::symmetrize_cyclic` (const `ex` &thisex, const `lst` &l)
- `ex GiNaC::op` (const `ex` &thisex, size\_t i)
- `ex GiNaC::lhs` (const `ex` &thisex)
- `ex GiNaC::rhs` (const `ex` &thisex)
- `bool GiNaC::is_zero` (const `ex` &thisex)
- `void GiNaC::swap` (`ex` &e1, `ex` &e2)
- `ex GiNaC::subs` (const `ex` &thisex, const `exmap` &m, unsigned `options`=0)
- `ex GiNaC::subs` (const `ex` &thisex, const `lst` &ls, const `lst` &lr, unsigned `options`=0)
- `ex GiNaC::subs` (const `ex` &thisex, const `ex` &e, unsigned `options`=0)
- `template<class T >`  
`bool GiNaC::is_a` (const `ex` &obj)  
*Check if ex is a handle to a T, including base classes.*
- `template<class T >`  
`bool GiNaC::is_exactly_a` (const `ex` &obj)  
*Check if ex is a handle to a T, not including base classes.*
- `template<class T >`  
`const T & GiNaC::ex_to` (const `ex` &e)  
*Return a reference to the basic-derived class T object embedded in an expression.*
- `template<> void std::swap` (`GiNaC::ex` &a, `GiNaC::ex` &b)  
*Specialization of `std::swap()` for ex objects.*

## Variables

- static `library_init GiNaC::library_initializer`  
*For construction of flyweights, etc.*
- const `basic * GiNaC::_num0_bp`

### 7.19.1 Detailed Description

Interface to `GiNaC`'s light-weight expression handles.

## 7.20 excompiler.cpp File Reference

Functions to facilitate the conversion of a `ex` to a function pointer suited for fast numerical integration.

```
#include "excompiler.h"
#include "ex.h"
#include "lst.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
```

```
#include <cstdlib>
#include <fstream>
#include <ios>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym, [FUNCP\\_1P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym1, const [symbol](#) &sym2, [FUNCP\\_2P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [lst](#) &exprs, const [lst](#) &syms, [FUNCP\\_CUBA](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_1P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_2P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_CUBA](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 7.20.1 Detailed Description

Functions to facilitate the conversion of a [ex](#) to a function pointer suited for fast numerical integration.

## 7.21 excompiler.h File Reference

Functions to facilitate the conversion of a [ex](#) to a function pointer suited for fast numerical integration.

```
#include "lst.h"
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef double(\* [GiNaC::FUNCP\\_1P](#)) (double)  
*Function pointer with one function parameter.*
- typedef double(\* [GiNaC::FUNCP\\_2P](#)) (double, double)  
*Function pointer with two function parameters.*
- typedef void(\* [GiNaC::FUNCP\\_CUBA](#)) (const int \*, const double[], const int \*, double[])  
*Function pointer for use with the CUBA library ( <http://www.feynarts.de/cuba>).*

## Functions

- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym, [FUNCP\\_1P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [ex](#) &expr, const [symbol](#) &sym1, const [symbol](#) &sym2, [FUNCP\\_2P](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::compile\\_ex](#) (const [lst](#) &exprs, const [lst](#) &syms, [FUNCP\\_CUBA](#) &fp, const std::string filename="")  
*Takes an expression and produces a function pointer to the compiled and linked C code equivalent in double precision.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_1P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_1P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_2P](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_2P to the contained function.*
- void [GiNaC::link\\_ex](#) (const std::string filename, [FUNCP\\_CUBA](#) &fp)  
*Opens an existing so-file and returns a function pointer of type FUNCP\_CUBA to the contained function.*
- void [GiNaC::unlink\\_ex](#) (const std::string filename)  
*Closes all linked .so files that have the supplied filename.*

### 7.21.1 Detailed Description

Functions to facilitate the conversion of a [ex](#) to a function pointer suited for fast numerical integration.

## 7.22 expair.cpp File Reference

Implementation of expression pairs (building blocks of [expairseq](#)).

```
#include "expair.h"
#include "operators.h"
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

### 7.22.1 Detailed Description

Implementation of expression pairs (building blocks of [expairseq](#)).

## 7.23 `expair.h` File Reference

Definition of expression pairs (building blocks of `expairseq`).

```
#include "ex.h"
#include "numeric.h"
#include "print.h"
```

### Classes

- class [GiNaC::expair](#)  
*A pair of expressions.*
- struct [GiNaC::expair\\_is\\_less](#)  
*Function object for insertion into third argument of STL's `sort()` etc.*
- struct [GiNaC::expair\\_rest\\_is\\_less](#)  
*Function object not caring about the numerical coefficients for insertion into third argument of STL's `sort()`.*
- struct [GiNaC::expair\\_swap](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- void [GiNaC::swap](#) ([expair](#) &e1, [expair](#) &e2)

#### 7.23.1 Detailed Description

Definition of expression pairs (building blocks of `expairseq`).

## 7.24 `expairseq.cpp` File Reference

Implementation of sequences of expression pairs.

```
#include "expairseq.h"
#include "lst.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "wildcard.h"
#include "archive.h"
#include "operators.h"
#include "utils.h"
#include "hash_seed.h"
#include "indexed.h"
#include "compiler.h"
#include <algorithm>
#include <iostream>
#include <iterator>
#include <memory>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([expairseq](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[expairseq::do\\_print](#)). [print\\_func](#)< [print\\_tree](#)>(&[expairseq::do\\_print\\_tree](#))) [class](#) [epp\\_is\\_less](#)
- [epvector](#) \* [GiNaC::conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*

### 7.24.1 Detailed Description

Implementation of sequences of expression pairs.

## 7.25 expairseq.h File Reference

Interface to sequences of expression pairs.

```
#include "expair.h"
#include "indexed.h"
#include <vector>
```

## Classes

- class [GiNaC::expairseq](#)  
*A sequence of class expair.*
- class [GiNaC::make\\_flat\\_inserter](#)  
*Class to handle the renaming of dummy indices.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< [expair](#) > [GiNaC::epvector](#)  
*expair-vector*
- typedef [epvector::iterator](#) [GiNaC::epp](#)  
*expair-vector pointer*

## Functions

- [epvector](#) \* [GiNaC::conjugateepvector](#) (const [epvector](#) &)  
*Complex conjugate every element of an epvector.*

### 7.25.1 Detailed Description

Interface to sequences of expression pairs.

## 7.26 exprseq.cpp File Reference

Implementation of [GiNaC](#)'s exprseq.

```
#include "exprseq.h"
```

### Namespaces

- namespace [GiNaC](#)

### 7.26.1 Detailed Description

Implementation of [GiNaC](#)'s exprseq.

## 7.27 exprseq.h File Reference

Definition of [GiNaC](#)'s exprseq.

```
#include "container.h"
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef [container](#)< std::vector > [GiNaC::exprseq](#)

### 7.27.1 Detailed Description

Definition of [GiNaC](#)'s exprseq.

## 7.28 factor.cpp File Reference

Polynomial factorization (implementation).

```
#include "factor.h"
#include "ex.h"
#include "numeric.h"
#include "operators.h"
#include "inifcns.h"
#include "symbol.h"
#include "relational.h"
#include "power.h"
#include "mul.h"
#include "normal.h"
#include "add.h"
#include <type_traits>
#include <algorithm>
#include <limits>
#include <list>
#include <vector>
#include <stack>
#include <cln/cln.h>
```

### Namespaces

- namespace [GiNaC](#)

### Macros

- `#define` [DCOUT](#)(str)
- `#define` [DCOUTVAR](#)(var)
- `#define` [DCOUT2](#)(str, var)
- `#define` [USE\\_SAME\\_DEGREE\\_FACTOR](#)

### Functions

- `ex` [GiNaC::factor](#) (const `ex` &`poly`, unsigned `options`)  
*Interface function to the outside world.*

### 7.28.1 Detailed Description

Polynomial factorization (implementation).

The interface function `factor()` at the end of this file is defined in the [GiNaC](#) namespace. All other utility functions and classes are defined in an additional anonymous namespace.

Factorization starts by doing a square free factorization and making the coefficients integer. Then, depending on the number of free variables it proceeds either in dedicated univariate or multivariate factorization code.

Univariate factorization does a modular factorization via Berlekamp's algorithm and distinct degree factorization. Hensel lifting is used at the end.

Multivariate factorization uses the univariate factorization (applying a evaluation homomorphism first) and Hensel lifting raises the answer to the multivariate domain. The Hensel lifting code is completely distinct from the code used by the univariate factorization.

Algorithms used can be found in [Wan] An Improved Multivariate Polynomial Factoring Algorithm, P.S.Wang, Mathematics of Computation, Vol. 32, No. 144 (1978) 1215–1231. [GCL] Algorithms for Computer Algebra, K.O.Geddes, S.R.Czapor, G.Labahn, Springer Verlag, 1992. [Mig] Some Useful Bounds, M.Mignotte, In "Computer Algebra, Symbolic and Algebraic Computation" (B.Buchberger et al., eds.), pp. 259-263, Springer-Verlag, New York, 1982.

## 7.28.2 Macro Definition Documentation

### 7.28.2.1 DCOUT

```
#define DCOUT(
 str)
```

### 7.28.2.2 DCOUTVAR

```
#define DCOUTVAR(
 var)
```

### 7.28.2.3 DCOUT2

```
#define DCOUT2(
 str,
 var)
```

### 7.28.2.4 USE\_SAME\_DEGREE\_FACTOR

```
#define USE_SAME_DEGREE_FACTOR
```

## 7.28.3 Variable Documentation

### 7.28.3.1 value

```
const bool value = false [static]
```

Referenced by [GiNaC::archive\\_node::add\\_bool\(\)](#), [GiNaC::archive\\_node::add\\_ex\(\)](#), [GiNaC::archive\\_node::add\\_string\(\)](#), [GiNaC::archive\\_node::add\\_unsigned\(\)](#), [GiNaC::Li2\\_\(\)](#), [GiNaC::matrix::set\(\)](#), and [GiNaC::subsvalue\(\)](#).

### 7.28.3.2 r

```
size_t r [private]
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), [GiNaC::collect\\_common\\_factors\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::numeric::csgn\(\)](#), [GiNaC::numeric::denom\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::idx\\_symmetrization\(\)](#), [GiNaC::matrix::inverse\(\)](#), [GiNaC::iquo\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::numeric::numer\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::matrix::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::numeric::print\\_numeric\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::matrix::rank\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::return\\_type\\_tinfo\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::refcounted::set\\_refcount\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::numeric::step\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::matrix::trace\(\)](#), [GiNaC::matrix::transpose\(\)](#), [GiNaC::unit\\_matrix\(\)](#), and [GiNaC::zeta1\\_evalf\(\)](#).

## 7.28.3.3 c

```
size_t c [private]
```

Referenced by [GiNaC::abs\\_print\\_csrc\\_float\(\)](#), [GiNaC::abs\\_print\\_latex\(\)](#), [GiNaC::symmetry::add\(\)](#), [GiNaC::bernoulli\(\)](#), [GiNaC::matrix::charpoly\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::color\\_d\(\)](#), [GiNaC::color\\_f\(\)](#), [GiNaC::color\\_h\(\)](#), [GiNaC::expairseq::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_ex\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::expairseq::combine\\_overall\\_coeff\(\)](#), [GiNaC::mul::combine\\_overall\\_coeff\(\)](#), [GiNaC::expairseq::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::add::combine\\_pair\\_with\\_coeff\\_to\\_pair\(\)](#), [GiNaC::mul::combine\\_pair\\_v](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::conjugate\\_print\\_latex\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::crc32\(\)](#), [GiNaC::pseries::derivative\(\)](#), [GiNaC::matrix::determinant\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::add::do\\_print\(\)](#), [GiNaC::basic::do\\_print\(\)](#), [GiNaC::constant::do\\_print\(\)](#), [GiNaC::container< C >::do\\_print\(\)](#), [GiNaC::expairseq::do\\_print\(\)](#), [GiNaC::fderivative::do\\_print\(\)](#), [GiNaC::idx::do\\_print\(\)](#), [GiNaC::varidx::do\\_print\(\)](#), [GiNaC::spinidx::do\\_print\(\)](#), [GiNaC::indexed::do\\_print\(\)](#), [GiNaC::integral::do\\_print\(\)](#), [GiNaC::integration\\_kernel::do\\_print\(\)](#), [GiNaC::basic\\_log\\_kernel::do\\_print\(\)](#), [GiNaC::multiple\\_polylog\\_kernel::do\\_print\(\)](#), [GiNaC::ELi\\_kernel::do\\_print\(\)](#), [GiNaC::Ebar\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::do\\_print\(\)](#), [GiNaC::Kronecker\\_dz\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_kernel::do\\_print\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::do\\_print\(\)](#), [GiNaC::modular\\_form\\_kernel::do\\_print\(\)](#), [GiNaC::user\\_defined\\_kernel::do\\_print\(\)](#), [GiNaC::matrix::do\\_print\(\)](#), [GiNaC::mul::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\(\)](#), [GiNaC::numeric::do\\_print\(\)](#), [GiNaC::pseries::do\\_print\(\)](#), [GiNaC::relational::do\\_print\(\)](#), [GiNaC::symbol::do\\_print\(\)](#), [GiNaC::symmetry::do\\_print\(\)](#), [GiNaC::wildcard::do\\_print\(\)](#), [GiNaC::ncmul::do\\_print\\_csrc\(\)](#), [GiNaC::add::do\\_print\\_csrc\(\)](#), [GiNaC::fderivative::do\\_print\\_csrc\(\)](#), [GiNaC::idx::do\\_print\\_csrc\(\)](#), [GiNaC::mul::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\(\)](#), [GiNaC::power::do\\_print\\_csrc\(\)](#), [GiNaC::numeric::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::power::do\\_print\\_csrc\\_cl\\_N\(\)](#), [GiNaC::clifford::do\\_print\\_dflt\(\)](#), [GiNaC::power::do\\_print\\_dflt\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::add::do\\_print\\_latex\(\)](#), [GiNaC::clifford::do\\_print\\_latex\(\)](#), [GiNaC::constant::do\\_print\\_latex\(\)](#), [GiNaC::idx::do\\_print\\_latex\(\)](#), [GiNaC::spinidx::do\\_print\\_latex\(\)](#), [GiNaC::indexed::do\\_print\\_latex\(\)](#), [GiNaC::integral::do\\_print\\_latex\(\)](#), [GiNaC::matrix::do\\_print\\_latex\(\)](#), [GiNaC::mul::do\\_print\\_latex\(\)](#), [GiNaC::numeric::do\\_print\\_latex\(\)](#), [GiNaC::power::do\\_print\\_latex\(\)](#), [GiNaC::pseries::do\\_print\\_latex\(\)](#), [GiNaC::symbol::do\\_print\\_latex\(\)](#), [GiNaC::container< C >::do\\_print\\_python\(\)](#), [GiNaC::power::do\\_print\\_python\(\)](#), [GiNaC::pseries::do\\_print\\_python\(\)](#), [GiNaC::add::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_python\\_repr\(\)](#), [GiNaC::constant::do\\_print\\_python\\_repr\(\)](#), [GiNaC::container< C >::do\\_print\\_python\\_repr\(\)](#), [GiNaC::matrix::do\\_print\\_python\\_repr\(\)](#), [GiNaC::mul::do\\_print\\_python\\_repr\(\)](#), [GiNaC::numeric::do\\_print\\_python\\_repr\(\)](#), [GiNaC::power::do\\_print\\_python\\_repr\(\)](#), [GiNaC::pseries::do\\_print\\_python\\_repr\(\)](#), [GiNaC::relational::do\\_print\\_python\\_repr\(\)](#), [GiNaC::symbol::do\\_print\\_python\\_repr\(\)](#), [GiNaC::wildcard::do\\_print\\_python\\_repr\(\)](#), [GiNaC::basic::do\\_print\\_tree\(\)](#), [GiNaC::clifford::do\\_print\\_tree\(\)](#), [GiNaC::constant::do\\_print\\_tree\(\)](#), [GiNaC::container< C >::do\\_print\\_tr](#), [GiNaC::expairseq::do\\_print\\_tree\(\)](#), [GiNaC::fderivative::do\\_print\\_tree\(\)](#), [GiNaC::idx::do\\_print\\_tree\(\)](#), [GiNaC::varidx::do\\_print\\_tree\(\)](#), [GiNaC::spinidx::do\\_print\\_tree\(\)](#), [GiNaC::indexed::do\\_print\\_tree\(\)](#), [GiNaC::numeric::do\\_print\\_tree\(\)](#), [GiNaC::pseries::do\\_print\\_tree\(\)](#), [GiNaC::symbol::do\\_print\\_tree\(\)](#), [GiNaC::symmetry::do\\_print\\_tree\(\)](#), [GiNaC::wildcard::do\\_print\\_tree\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::factorial\\_print\\_dflt\\_latex\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::imag\\_part\\_print\\_latex\(\)](#), [GiNaC::add::integer\\_content\(\)](#), [GiNaC::matrix::inverse\(\)](#), [GiNaC::lcmcoeff\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::Isolve\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::matrix::mul\(\)](#), [GiNaC::matrix::mul\\_scalar\(\)](#), [GiNaC::print\\_funcutor::operator\(\)](#), [GiNaC::print\\_ptrfun\\_handler< T, C >::operator](#), [GiNaC::print\\_memfun\\_handler< T, C >::operator\(\)](#), [GiNaC::error\\_and\\_integral\\_is\\_less::operator\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::pseries::power\\_const\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::basic::print\(\)](#), [GiNaC::ex::print\(\)](#), [GiNaC::fderivative::print\(\)](#), [GiNaC::function::print\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::print\(\)](#), [GiNaC::expair::print\(\)](#), [GiNaC::add::print\\_add\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), [GiNaC::basic::print\\_dispatch\(\)](#), [GiNaC::matrix::print\\_elements\(\)](#), [GiNaC::idx::print\\_index\(\)](#), [GiNaC::indexed::print\\_indexed\(\)](#), [GiNaC::print\\_integer\\_csrc\(\)](#), [GiNaC::numeric::print\\_numeric\(\)](#), [GiNaC::print\\_operator\(\)](#), [GiNaC::mul::print\\_overall\\_coeff\(\)](#), [GiNaC::power::print\\_power\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [GiNaC::pseries::print\\_series\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::indexed::printindices\(\)](#), [GiNaC::expairseq::printpair\(\)](#), [GiNaC::expairseq::printseq\(\)](#), [GiNaC::container< C >::printseq\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::real\\_part\\_print\\_latex\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::symbolic\\_matrix\(\)](#), [GiNaC::matrix::transpose\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::unit\\_matrix\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

## 7.28.3.4 m

```
mvec m [private]
```

Referenced by [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::charpoly\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [GiNaC::cols\(\)](#), [GiNaC::convert\\_H\\_to\\_Li\(\)](#), [GiNaC::determinant\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::tensdelta::eval\\_indexed\(\)](#), [GiNaC::tensmetric::eval\\_indexed\(\)](#), [GiNaC::evalf\(\)](#), [GiNaC::add::evalm\(\)](#), [GiNaC::mul::evalm\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::H\\_series\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::nops\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::rank\(\)](#), [GiNaC::rank\(\)](#), [GiNaC::reduced\\_matrix\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::rows\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::smod\(\)](#), [GiNaC::sub\\_matrix\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::subs\(\)](#), [GiNaC::symbol::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::tgamma\\_series\(\)](#), [GiNaC::trace\(\)](#), [GiNaC::transpose\(\)](#), [GiNaC::zeta1\\_deriv\(\)](#), [GiNaC::zeta1\\_eval\(\)](#), [GiNaC::zeta1\\_print\\_latex\(\)](#), [GiNaC::zeta2\\_deriv\(\)](#), [GiNaC::zeta2\\_eval\(\)](#), and [GiNaC::zeta2\\_print\\_latex\(\)](#).

### 7.28.3.5 lr

```
umodpoly lr[2] [private]
```

Referenced by [GiNaC::subs\(\)](#), and [GiNaC::ex::subs\(\)](#).

### 7.28.3.6 cache

```
vector<vector<umodpoly> > cache [private]
```

### 7.28.3.7 factors

```
upvec factors [private]
```

Referenced by [GiNaC::ncmul::count\\_factors\(\)](#), [GiNaC::ncmul::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::mul::mul\(\)](#), [GiNaC::sqrfree\(\)](#), and [GiNaC::sqrfree\\_yun\(\)](#).

### 7.28.3.8 one

```
umodpoly one [private]
```

Referenced by [GiNaC::integration\\_kernel::get\\_numerical\\_value\\_impl\(\)](#), and [GiNaC::iterated\\_integral\\_evalf\\_impl\(\)](#).

## 7.28.3.9 n

```
size_t n [private]
```

Referenced by [GiNaC::class\\_info< OPT >::tree\\_node::add\\_child\(\)](#), [GiNaC::archive::add\\_node\(\)](#), [GiNaC::basic::archive\(\)](#), [GiNaC::clifford::archive\(\)](#), [GiNaC::color::archive\(\)](#), [GiNaC::constant::archive\(\)](#), [GiNaC::container< C >::archive\(\)](#), [GiNaC::expairseq::archive\(\)](#), [GiNaC::fderivative::archive\(\)](#), [GiNaC::function::archive\(\)](#), [GiNaC::idx::archive\(\)](#), [GiNaC::varidx::archive\(\)](#), [GiNaC::spinidx::archive\(\)](#), [GiNaC::indexed::archive\(\)](#), [GiNaC::integral::archive\(\)](#), [GiNaC::matrix::archive\(\)](#), [GiNaC::numeric::archive\(\)](#), [GiNaC::power::archive\(\)](#), [GiNaC::pseries::archive\(\)](#), [GiNaC::relational::archive\(\)](#), [GiNaC::symbol::archive\(\)](#), [GiNaC::symmetry::archive\(\)](#), [GiNaC::minkmetric::archive\(\)](#), [GiNaC::tensepsilon::archive\(\)](#), [GiNaC::wildcard::archive\(\)](#), [GiNaC::archive::archive\(\)](#), [GiNaC::bernoulli\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::basic::coeff\(\)](#), [GiNaC::ex::coeff\(\)](#), [GiNaC::add::coeff\(\)](#), [GiNaC::mul::coeff\(\)](#), [GiNaC::ncmul::coeff\(\)](#), [GiNaC::numeric::coeff\(\)](#), [GiNaC::power::coeff\(\)](#), [GiNaC::pseries::coeff\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::coeff\(\)](#), [GiNaC::coeff\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::coefficient\\_an\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::const\\_postorder\\_iterator::const\\_postorder\\_iterator\(\)](#), [GiNaC::const\\_preorder\\_iterator::const\\_preorder\\_iterator\(\)](#), [GiNaC::composition\\_generator::coolmulti::coolmulti\(\)](#), [GiNaC::matrix::determinant\\_minor\(\)](#), [GiNaC::dirichlet\\_character\(\)](#), [GiNaC::matrix::division\\_free\\_elimination\(\)](#), [GiNaC::doublefactorial\(\)](#), [GiNaC::class\\_info< OPT >::dump\\_tree\(\)](#), [GiNaC::matrix::echelon\\_form\(\)](#), [GiNaC::power::eval\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::factorial\(\)](#), [GiNaC::fibonacci\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::matrix::fraction\\_free\\_elimination\(\)](#), [GiNaC::function\\_options::function\\_options\(\)](#), [GiNaC::function\\_options::function\\_options\(\)](#), [GiNaC::matrix::gauss\\_elimination\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::golden\\_ratio\\_hash\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::ifactor\(\)](#), [GiNaC::power::imag\\_part\(\)](#), [GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field\(\)](#), [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::log2\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::basic::map\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::multinomial\\_coefficient\(\)](#), [GiNaC::add::normal\(\)](#), [GiNaC::mul::normal\(\)](#), [GiNaC::pseries::normal\(\)](#), [GiNaC::const\\_iterator::operator+\(\)](#), [GiNaC::const\\_iterator::operator+=\(\(\)\)](#), [GiNaC::const\\_iterator::operator-\(\)](#), [GiNaC::const\\_iterator::operator-=\(\)](#), [GiNaC::const\\_iterator::operator\[\]\(\)](#), [GiNaC::primitive\\_dirichlet\\_character\(\)](#), [GiNaC::psi2\\_deriv\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::clifford::read\\_archive\(\)](#), [GiNaC::color::read\\_archive\(\)](#), [GiNaC::container< C >::read\\_archive\(\)](#), [GiNaC::constant::read\\_archive\(\)](#), [GiNaC::expairseq::read\\_archive\(\)](#), [GiNaC::fderivative::read\\_archive\(\)](#), [GiNaC::function::read\\_archive\(\)](#), [GiNaC::idx::read\\_archive\(\)](#), [GiNaC::varidx::read\\_archive\(\)](#), [GiNaC::spinidx::read\\_archive\(\)](#), [GiNaC::indexed::read\\_archive\(\)](#), [GiNaC::integral::read\\_archive\(\)](#), [GiNaC::matrix::read\\_archive\(\)](#), [GiNaC::numeric::read\\_archive\(\)](#), [GiNaC::power::read\\_archive\(\)](#), [GiNaC::pseries::read\\_archive\(\)](#), [GiNaC::relational::read\\_archive\(\)](#), [GiNaC::symbol::read\\_archive\(\)](#), [GiNaC::symmetry::read\\_archive\(\)](#), [GiNaC::minkmetric::read\\_archive\(\)](#), [GiNaC::tensepsilon::read\\_archive\(\)](#), [GiNaC::wildcard::read\\_archive\(\)](#), [GiNaC::power::real\\_part\(\)](#), [GiNaC::container\\_storage< C >::read\\_archive\(\)](#), [GiNaC::container\\_storage< C >::reserve\(\)](#), [GiNaC::rotate\\_left\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::symbol::set\\_name\(\)](#), [GiNaC::function\\_options::set\\_name\(\)](#), [GiNaC::symbol::set\\_TeX\\_name\(\)](#), [GiNaC::matrix::solve\(\)](#), [GiNaC::sqfree\\_parfrac\(\)](#), [GiNaC::function\\_options::test\\_and\\_set\\_nparams\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::ex::traverse\\_postorder\(\)](#), [GiNaC::ex::traverse\\_preorder\(\)](#), [GiNaC::symmetry::validate\(\)](#), [GiNaC::write\\_real\\_float\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

## 7.28.3.10 len

```
size_t len [private]
```

Referenced by [GiNaC::crc32\(\)](#).

## 7.28.3.11 last

```
size_t last [private]
```

Referenced by [GiNaC::antisymmetrize\(\)](#), [GiNaC::expairseq::combine\\_same\\_terms\\_sorted\\_seq\(\)](#), [GiNaC::expairseq::construct\\_from\(\)](#), [GiNaC::cyclic\\_permutation\(\)](#), [GiNaC::mul::eval\(\)](#), [GiNaC::expairseq::evalchildren\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::find\\_free\\_and\\_dummy\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [GiNaC::permutation\\_sign\(\)](#), [GiNaC::shaker\\_sort\(\)](#), [GiNaC::expairseq::subchildren\(\)](#), [GiNaC::symm\(\)](#), [GiNaC::symmetrize\(\)](#), and [GiNaC::symmetrize\\_cyclic\(\)](#).

### 7.28.3.12 k

```
vector<int> k [private]
```

Referenced by [GiNaC::bernoulli\(\)](#), [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::binomial\(\)](#), [GiNaC::scalar\\_products::debugprint\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::EllipticE\\_deriv\(\)](#), [GiNaC::EllipticE\\_eval\(\)](#), [GiNaC::EllipticE\\_evalf\(\)](#), [GiNaC::EllipticE\\_print\\_latex\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_deriv\(\)](#), [GiNaC::EllipticK\\_eval\(\)](#), [GiNaC::EllipticK\\_evalf\(\)](#), [GiNaC::EllipticK\\_print\\_latex\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::get\\_sign\(\)](#), [GiNaC::log2\(\)](#), [GiNaC::matrix::markowitz\\_elimination\(\)](#), [GiNaC::basic\\_partition\\_generator::mpartition2::mpartition2\(\)](#), [GiNaC::basic\\_partition\\_generator::composition\\_generator::coolmulti::next\\_permutation\(\)](#), [GiNaC::multi\\_iterator\\_ordered< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_counter\\_indv< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_permutation< T >::operator++\(\)](#), [GiNaC::multi\\_iterator\\_shuffle< T >::operator++\(\)](#), [GiNaC::matrix::pivot\(\)](#), [GiNaC::resultant\(\)](#), [GiNaC::ELi\\_kernel::series\\_coeff\\_impl\(\)](#), [GiNaC::Ebar\\_kernel::series\\_coeff\\_impl\(\)](#), and [GiNaC::sqrfree\\_parfrac\(\)](#).

### 7.28.3.13 poly

```
const ex poly
```

Referenced by [GiNaC::matrix::charpoly\(\)](#), and [GiNaC::factor\(\)](#).

### 7.28.3.14 x

```
const ex x
```

Referenced by [GiNaC::abs\(\)](#), [GiNaC::acos\(\)](#), [GiNaC::acos\\_conjugate\(\)](#), [GiNaC::acos\\_deriv\(\)](#), [GiNaC::acos\\_eval\(\)](#), [GiNaC::acos\\_evalf\(\)](#), [GiNaC::acosh\(\)](#), [GiNaC::acosh\\_conjugate\(\)](#), [GiNaC::acosh\\_deriv\(\)](#), [GiNaC::acosh\\_eval\(\)](#), [GiNaC::acosh\\_evalf\(\)](#), [GiNaC::adaptivesimpson\(\)](#), [GiNaC::asin\(\)](#), [GiNaC::asin\\_conjugate\(\)](#), [GiNaC::asin\\_deriv\(\)](#), [GiNaC::asin\\_eval\(\)](#), [GiNaC::asin\\_evalf\(\)](#), [GiNaC::asin\\_info\(\)](#), [GiNaC::asinh\(\)](#), [GiNaC::asinh\\_conjugate\(\)](#), [GiNaC::asinh\\_deriv\(\)](#), [GiNaC::asinh\\_eval\(\)](#), [GiNaC::asinh\\_evalf\(\)](#), [GiNaC::atan\(\)](#), [GiNaC::atan2\\_deriv\(\)](#), [GiNaC::atan2\\_eval\(\)](#), [GiNaC::atan2\\_evalf\(\)](#), [GiNaC::atan2\\_info\(\)](#), [GiNaC::atan\\_conjugate\(\)](#), [GiNaC::atan\\_deriv\(\)](#), [GiNaC::atan\\_eval\(\)](#), [GiNaC::atan\\_evalf\(\)](#), [GiNaC::atan\\_info\(\)](#), [GiNaC::atanh\(\)](#), [GiNaC::atanh\\_conjugate\(\)](#), [GiNaC::atanh\\_deriv\(\)](#), [GiNaC::atanh\\_eval\(\)](#), [GiNaC::atanh\\_evalf\(\)](#), [GiNaC::Bernoulli\\_polynomial\(\)](#), [GiNaC::beta\\_deriv\(\)](#), [GiNaC::beta\\_eval\(\)](#), [GiNaC::beta\\_evalf\(\)](#), [GiNaC::binomial\\_conjugate\(\)](#), [GiNaC::binomial\\_eval\(\)](#), [GiNaC::binomial\\_evalf\(\)](#), [GiNaC::binomial\\_real\\_part\(\)](#), [GiNaC::binomial\\_sym\(\)](#), [GiNaC::lanczos\\_coeffs::calc\\_lanczos\\_A\(\)](#), [GiNaC::basic::collect\(\)](#), [GiNaC::container< C >::conjugate\(\)](#), [GiNaC::expairseq::conjugate\(\)](#), [GiNaC::matrix::conjugate\(\)](#), [GiNaC::mul::conjugate\(\)](#), [GiNaC::conjugateepvector\(\)](#), [GiNaC::ex::content\(\)](#), [GiNaC::convert\\_H\\_to\\_Li\(\)](#), [GiNaC::cos\(\)](#), [GiNaC::cos\\_conjugate\(\)](#), [GiNaC::cos\\_deriv\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cos\\_evalf\(\)](#), [GiNaC::cos\\_imag\\_part\(\)](#), [GiNaC::cos\\_real\\_part\(\)](#), [GiNaC::cosh\(\)](#), [GiNaC::cosh\\_conjugate\(\)](#), [GiNaC::cosh\\_deriv\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::cosh\\_evalf\(\)](#), [GiNaC::cosh\\_imag\\_part\(\)](#), [GiNaC::cosh\\_real\\_part\(\)](#), [GiNaC::csgn\(\)](#), [GiNaC::decomp\\_rational\(\)](#), [GiNaC::denom\(\)](#), [GiNaC::divide\(\)](#), [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::eta\\_conjugate\(\)](#), [GiNaC::eta\\_eval\(\)](#), [GiNaC::eta\\_evalf\(\)](#), [GiNaC::eta\\_imag\\_part\(\)](#), [GiNaC::eta\\_series\(\)](#), [GiNaC::tensepsilon::eval\\_indexed\(\)](#), [GiNaC::exp\(\)](#), [GiNaC::exp\\_conjugate\(\)](#), [GiNaC::exp\\_deriv\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::exp\\_evalf\(\)](#), [GiNaC::exp\\_imag\\_part\(\)](#), [GiNaC::exp\\_info\(\)](#), [GiNaC::exp\\_power\(\)](#), [GiNaC::exp\\_real\\_part\(\)](#), [GiNaC::factorial\\_conjugate\(\)](#), [GiNaC::factorial\\_eval\(\)](#), [GiNaC::factorial\\_evalf\(\)](#), [GiNaC::factorial\\_print\\_dfl\\_t\\_latex\(\)](#), [GiNaC::factorial\\_real\\_part\(\)](#), [GiNaC::find\\_common\\_factor\(\)](#), [GiNaC::frac\\_cancel\(\)](#), [GiNaC::fsolve\(\)](#), [GiNaC::G\(\)](#), [GiNaC::G2\\_eval\(\)](#), [GiNaC::G2\\_evalf\(\)](#), [GiNaC::G3\\_eval\(\)](#), [GiNaC::G3\\_evalf\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::generalised\\_Bernoulli\\_number\(\)](#), [GiNaC::get\\_first\\_symbol\(\)](#), [GiNaC::guess\\_precision\(\)](#), [GiNaC::H\\_deriv\(\)](#), [GiNaC::H\\_eval\(\)](#), [GiNaC::H\\_evalf\(\)](#), [GiNaC::H\\_print\\_latex\(\)](#), [GiNaC::H\\_series\(\)](#), [GiNaC::make\\_flat\\_inserter::handle\\_factor\(\)](#), [GiNaC::hasindex\(\)](#), [GiNaC::haswild\(\)](#), [GiNaC::heur\\_gcd\\_z\(\)](#), [GiNaC::imag\(\)](#), [GiNaC::interpolate\(\)](#), [GiNaC::inverse\(\)](#), [GiNaC::is\\_cinteger\(\)](#), [GiNaC::is\\_crational\(\)](#), [GiNaC::is\\_even\(\)](#), [GiNaC::is\\_integer\(\)](#), [GiNaC::is\\_negative\(\)](#), [GiNaC::is\\_nonneg\\_integer\(\)](#), [GiNaC::is\\_odd\(\)](#), [GiNaC::is\\_pos\\_integer\(\)](#), [GiNaC::is\\_positive\(\)](#), [GiNaC::is\\_prime\(\)](#), [GiNaC::is\\_rational\(\)](#), [GiNaC::is\\_real\(\)](#), [GiNaC::is\\_the\\_function\(\)](#), [GiNaC::is\\_the\\_function< G\\_SERIAL >\(\)](#),

[GiNaC::is\\_the\\_function< iterated\\_integral\\_SERIAL >\(\)](#), [GiNaC::is\\_the\\_function< psi\\_SERIAL >\(\)](#), [GiNaC::is\\_the\\_function< zeta\\_SERIAL >\(\)](#), [GiNaC::is\\_zero\(\)](#), [GiNaC::isqrt\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::lgamma\(\)](#), [GiNaC::lgamma\(\)](#), [GiNaC::lgamma\\_conjugate\(\)](#), [GiNaC::lgamma\\_deriv\(\)](#), [GiNaC::lgamma\\_eval\(\)](#), [GiNaC::lgamma\\_evalf\(\)](#), [GiNaC::Li2\(\)](#), [GiNaC::Li2\\_conjugate\(\)](#), [GiNaC::Li2\\_deriv\(\)](#), [GiNaC::Li2\\_eval\(\)](#), [GiNaC::Li2\\_evalf\(\)](#), [GiNaC::Li2\\_projection\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li3\\_eval\(\)](#), [GiNaC::Li\\_deriv\(\)](#), [GiNaC::Li\\_eval\(\)](#), [GiNaC::Li\\_evalf\(\)](#), [GiNaC::Li\\_print\\_latex\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\(\)](#), [GiNaC::log\\_conjugate\(\)](#), [GiNaC::log\\_deriv\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::log\\_evalf\(\)](#), [GiNaC::log\\_imag\\_part\(\)](#), [GiNaC::log\\_info\(\)](#), [GiNaC::log\\_real\\_part\(\)](#), [GiNaC::make\\_real\\_float\(\)](#), [GiNaC::matrix::matrix\(\)](#), [GiNaC::numer\(\)](#), [GiNaC::sym\\_desc::operator<\(\)](#), [GiNaC::Order\\_conjugate\(\)](#), [GiNaC::Order\\_eval\(\)](#), [GiNaC::Order\\_imag\\_part\(\)](#), [GiNaC::Order\\_power\(\)](#), [GiNaC::Order\\_real\\_part\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::pow\(\)](#), [GiNaC::prem\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::ex::primpart\(\)](#), [GiNaC::print\\_integer\\_csrc\(\)](#), [GiNaC::print\\_real\\_cl\\_N\(\)](#), [GiNaC::print\\_real\\_csrc\(\)](#), [GiNaC::print\\_real\\_number\(\)](#), [GiNaC::print\\_sym\\_pow\(\)](#), [GiNaC::psi1\\_deriv\(\)](#), [GiNaC::psi1\\_eval\(\)](#), [GiNaC::psi1\\_evalf\(\)](#), [GiNaC::psi2\\_deriv\(\)](#), [GiNaC::psi2\\_eval\(\)](#), [GiNaC::psi2\\_evalf\(\)](#), [GiNaC::quo\(\)](#), [GiNaC::read\\_real\\_float\(\)](#), [GiNaC::real\(\)](#), [GiNaC::rem\(\)](#), [GiNaC::S\\_deriv\(\)](#), [GiNaC::S\\_eval\(\)](#), [GiNaC::S\\_evalf\(\)](#), [GiNaC::S\\_print\\_latex\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::sin\(\)](#), [GiNaC::sin\\_conjugate\(\)](#), [GiNaC::sin\\_deriv\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sin\\_evalf\(\)](#), [GiNaC::sin\\_imag\\_part\(\)](#), [GiNaC::sin\\_real\\_part\(\)](#), [GiNaC::sinh\(\)](#), [GiNaC::sinh\\_conjugate\(\)](#), [GiNaC::sinh\\_deriv\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::sinh\\_evalf\(\)](#), [GiNaC::sinh\\_imag\\_part\(\)](#), [GiNaC::sinh\\_real\\_part\(\)](#), [GiNaC::sprem\(\)](#), [GiNaC::sqrfree\(\)](#), [GiNaC::sqrfree\\_parfrac\(\)](#), [GiNaC::sqrfree\\_yun\(\)](#), [GiNaC::sqrt\(\)](#), [GiNaC::sr\\_gcd\(\)](#), [GiNaC::step\(\)](#), [GiNaC::tan\(\)](#), [GiNaC::tan\\_conjugate\(\)](#), [GiNaC::tan\\_deriv\(\)](#), [GiNaC::tan\\_eval\(\)](#), [GiNaC::tan\\_evalf\(\)](#), [GiNaC::tan\\_imag\\_part\(\)](#), [GiNaC::tan\\_real\\_part\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\(\)](#), [GiNaC::tanh\\_conjugate\(\)](#), [GiNaC::tanh\\_deriv\(\)](#), [GiNaC::tanh\\_eval\(\)](#), [GiNaC::tanh\\_evalf\(\)](#), [GiNaC::tanh\\_imag\\_part\(\)](#), [GiNaC::tanh\\_real\\_part\(\)](#), [GiNaC::tanh\\_series\(\)](#), [GiNaC::tgamma\(\)](#), [GiNaC::tgamma\(\)](#), [GiNaC::tgamma\\_conjugate\(\)](#), [GiNaC::tgamma\\_deriv\(\)](#), [GiNaC::tgamma\\_eval\(\)](#), [GiNaC::tgamma\\_evalf\(\)](#), [GiNaC::to\\_double\(\)](#), [GiNaC::to\\_int\(\)](#), [GiNaC::to\\_long\(\)](#), [GiNaC::trig\\_info\(\)](#), [GiNaC::ex::unit\(\)](#), [GiNaC::unit\\_matrix\(\)](#), [GiNaC::ex::unitcontprim\(\)](#), [GiNaC::zeta\(\)](#), [GiNaC::zeta1\\_evalf\(\)](#), [GiNaC::zeta2\\_evalf\(\)](#), [GiNaC::zetaderiv\\_deriv\(\)](#), and [GiNaC::zetaderiv\\_eval\(\)](#).

### 7.28.3.15 evalpoint

```
int evalpoint
```

### 7.28.3.16 R

```
cl_modint_ring R
```

### 7.28.3.17 syms\_wox

```
const exset syms_wox
```

### 7.28.3.18 unit

```
ex unit
```

Referenced by [GiNaC::kronecker\\_symbol\(\)](#), [GiNaC::ex::primpart\(\)](#), and [GiNaC::ex::unitcontprim\(\)](#).

### 7.28.3.19 cont

```
ex cont
```

Referenced by [GiNaC::ex::content\(\)](#), [GiNaC::container< C >::imag\\_part\(\)](#), and [GiNaC::container< C >::real\\_part\(\)](#).

**7.28.3.20 pp**

ex pp

**7.28.3.21 vn**

ex vn

**7.28.3.22 vnlst**

exvector vnlst

**7.28.3.23 modulus**

numeric modulus

**7.28.3.24 syms**

exset syms

Referenced by [GiNaC::lsolve\(\)](#).

**7.28.3.25 options**

unsigned options

Referenced by [GiNaC::abs\\_expand\(\)](#), [GiNaC::mul::algebraic\\_subs\\_mul\(\)](#), [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::csgn\\_series\(\)](#), [GiNaC::determinant\(\)](#), [GiNaC::exp\\_expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::expand\(\)](#), [GiNaC::basic::expand\(\)](#), [GiNaC::ex::expand\(\)](#), [GiNaC::add::expand\(\)](#), [GiNaC::expairseq::expand\(\)](#), [GiNaC::function::expand\(\)](#), [GiNaC::indexed::expand\(\)](#), [GiNaC::integral::expand\(\)](#), [GiNaC::mul::expand\(\)](#), [GiNaC::ncmul::expand\(\)](#), [GiNaC::power::expand\(\)](#), [GiNaC::pseries::expand\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::expand\(\)](#), [GiNaC::power::expand\\_add\(\)](#), [GiNaC::power::expand\\_add\\_2\(\)](#), [GiNaC::power::expand\\_mul\(\)](#), [GiNaC::expairseq::expandchildren\(\)](#), [GiNaC::mul::expandchildren\(\)](#), [GiNaC::ncmul::expandchildren\(\)](#), [GiNaC::factor\(\)](#), [GiNaC::gcd\(\)](#), [GiNaC::basic::has\(\)](#), [GiNaC::mul::has\(\)](#), [GiNaC::power::has\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::has\(\)](#), [GiNaC::ex::has\(\)](#), [GiNaC::has\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::log\\_expand\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::lsolve\(\)](#), [GiNaC::expand\\_map\\_function::operator\(\)](#), [GiNaC::registered\\_class\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::function\\_options::print\\_func\(\)](#), [GiNaC::registered\\_class\\_options::print\\_func\(\)](#), [GiNaC::registered\\_class\\_options::print\\_func\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::set\\_print\\_context\(\)](#), [GiNaC::set\\_print\\_func\(\)](#), [GiNaC::set\\_print\\_func\(\)](#), [GiNaC::set\\_print\\_options\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), [GiNaC::simplify\\_indexed\(\)](#), [GiNaC::step\\_series\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::subs\(\)](#), [GiNaC::basic::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::clifford::subs\(\)](#), [GiNaC::container< C >::subs\(\)](#), [GiNaC::expairseq::subs\(\)](#), [GiNaC::idx::subs\(\)](#), [GiNaC::matrix::subs\(\)](#), [GiNaC::numeric::subs\(\)](#), [GiNaC::power::subs\(\)](#), [GiNaC::pseries::subs\(\)](#), [GiNaC::relational::subs\(\)](#), [GiNaC::structure< T, ComparisonPolicy >::subs\(\)](#), [GiNaC::symbol::subs\(\)](#), [GiNaC::ex::subs\(\)](#), [GiNaC::basic::subs\\_one\\_level\(\)](#), [GiNaC::container< C >::subschildren\(\)](#), [GiNaC::expairseq::subschildren\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

## 7.29 factor.h File Reference

Polynomial factorization.

### Namespaces

- namespace [GiNaC](#)

### Functions

- [ex](#) [GiNaC::factor](#) (const [ex](#) &[poly](#), unsigned [options](#))  
*Interface function to the outside world.*

### 7.29.1 Detailed Description

Polynomial factorization.

## 7.30 fail.cpp File Reference

Implementation of class signaling failure of operation.

```
#include "fail.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fail](#), [basic](#), [print\\_func](#)< [print\\_context](#) >(&[fail::do\\_print](#)). [print\\_func](#)< [print\\_tree](#) >(&[fail::do\\_print\\_tree](#))) [GINAC\\_BIND\\_UNARCHIVER](#)([fail](#))

### 7.30.1 Detailed Description

Implementation of class signaling failure of operation.

Considered somewhat obsolete (most of this can be replaced by exceptions).

## 7.31 fail.h File Reference

Interface to class signaling failure of operation.

```
#include "basic.h"
#include "archive.h"
```

### Classes

- class [GiNaC::fail](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([fail](#))

### 7.31.1 Detailed Description

Interface to class signaling failure of operation.

Considered obsolete somewhat obsolete (most of this can be replaced by exceptions).

## 7.32 fderivative.cpp File Reference

Implementation of abstract derivatives of functions.

```
#include "fderivative.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([fderivative](#), [function](#), [print\\_func< print\\_context >\(&fderivative::do\\_print\)](#), [print\\_func< print\\_latex >\(&fderivative::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&fderivative::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&fderivative::do\\_print\\_tree\)](#)) [fderivative](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([fderivative](#))

### 7.32.1 Detailed Description

Implementation of abstract derivatives of functions.

## 7.33 fderivative.h File Reference

Interface to abstract derivatives of functions.

```
#include "function.h"
#include <set>
```

### Classes

- class [GiNaC::fderivative](#)  
*This class represents the (abstract) derivative of a symbolic function.*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::multiset< unsigned > [GiNaC::paramset](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([fderivative](#))

### 7.33.1 Detailed Description

Interface to abstract derivatives of functions.

## 7.34 flags.h File Reference

Collection of all flags used through the [GiNaC](#) framework.

## Classes

- class [GiNaC::expand\\_options](#)  
*Flags to control the behavior of `expand()`.*
- class [GiNaC::has\\_options](#)  
*Flags to control the behavior of `has()`.*
- class [GiNaC::subs\\_options](#)  
*Flags to control the behavior of `subs()`.*
- class [GiNaC::domain](#)  
*Domain of an object.*
- class [GiNaC::series\\_options](#)  
*Flags to control series expansion.*
- class [GiNaC::determinant\\_algo](#)  
*Switch to control algorithm for determinant computation.*
- class [GiNaC::solve\\_algo](#)  
*Switch to control algorithm for linear system solving.*
- class [GiNaC::status\\_flags](#)  
*Flags to store information about the state of an object.*
- class [GiNaC::info\\_flags](#)  
*Possible attributes an object can have.*
- class [GiNaC::return\\_types](#)
- class [GiNaC::remember\\_strategies](#)  
*Strategies how to clean up the function remember cache.*
- class [GiNaC::factor\\_options](#)  
*Flags to control the polynomial factorization.*

## Namespaces

- namespace [GiNaC](#)

### 7.34.1 Detailed Description

Collection of all flags used through the [GiNaC](#) framework.

## 7.35 `function.cpp` File Reference

Implementation of class of symbolic functions.

```
#include "function.h"
#include "operators.h"
#include "fderivative.h"
#include "ex.h"
#include "lst.h"
#include "symmetry.h"
#include "print.h"
#include "power.h"
#include "archive.h"
#include "inifcns.h"
#include "utils.h"
```

```
#include "hash_seed.h"
#include "remember.h"
#include <iostream>
#include <limits>
#include <list>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (function)

### 7.35.1 Detailed Description

Implementation of class of symbolic functions.

## 7.36 function.h File Reference

Interface to class of symbolic functions.

```
#include "exprseq.h"
#include <string>
#include <vector>
```

## Classes

- class [GiNaC::function\\_options](#)
- class [GiNaC::do\\_taylor](#)

*Exception class thrown by classes which provide their own series expansion to signal that ordinary Taylor expansion is safe.*

- class [GiNaC::function](#)

*The class function is used to implement builtin functions like sin, cos... and user defined functions.*

## Namespaces

- namespace [GiNaC](#)

## Macros

- `#define DECLARE_FUNCTION_1P(NAME)`
- `#define DECLARE_FUNCTION_2P(NAME)`
- `#define DECLARE_FUNCTION_3P(NAME)`
- `#define DECLARE_FUNCTION_4P(NAME)`
- `#define DECLARE_FUNCTION_5P(NAME)`
- `#define DECLARE_FUNCTION_6P(NAME)`
- `#define DECLARE_FUNCTION_7P(NAME)`
- `#define DECLARE_FUNCTION_8P(NAME)`
- `#define DECLARE_FUNCTION_9P(NAME)`
- `#define DECLARE_FUNCTION_10P(NAME)`
- `#define DECLARE_FUNCTION_11P(NAME)`
- `#define DECLARE_FUNCTION_12P(NAME)`
- `#define DECLARE_FUNCTION_13P(NAME)`
- `#define DECLARE_FUNCTION_14P(NAME)`
- `#define REGISTER_FUNCTION(NAME, OPT)`
- `#define is_ex_the_function(OBJ, FUNCNAME) (GiNaC::is_the_function<FUNCNAME##_SERIAL>(OBJ))`

## Typedefs

- `typedef ex(* GiNaC::eval_funcp) ()`
- `typedef ex(* GiNaC::evalf_funcp) ()`
- `typedef ex(* GiNaC::conjugate_funcp) ()`
- `typedef ex(* GiNaC::real_part_funcp) ()`
- `typedef ex(* GiNaC::imag_part_funcp) ()`
- `typedef ex(* GiNaC::expand_funcp) ()`
- `typedef ex(* GiNaC::derivative_funcp) ()`
- `typedef ex(* GiNaC::expl_derivative_funcp) ()`
- `typedef ex(* GiNaC::power_funcp) ()`
- `typedef ex(* GiNaC::series_funcp) ()`
- `typedef void(* GiNaC::print_funcp) ()`
- `typedef bool(* GiNaC::info_funcp) ()`
- `typedef ex(* GiNaC::eval_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::evalf_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::conjugate_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::real_part_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::imag_part_funcp_1) (const ex &)`
- `typedef ex(* GiNaC::expand_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::derivative_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::expl_derivative_funcp_1) (const ex &, const symbol &)`
- `typedef ex(* GiNaC::power_funcp_1) (const ex &, const ex &)`
- `typedef ex(* GiNaC::series_funcp_1) (const ex &, const relational &, int, unsigned)`
- `typedef void(* GiNaC::print_funcp_1) (const ex &, const print_context &)`
- `typedef bool(* GiNaC::info_funcp_1) (const ex &, unsigned)`
- `typedef ex(* GiNaC::eval_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::evalf_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::conjugate_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::real_part_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::imag_part_funcp_2) (const ex &, const ex &)`
- `typedef ex(* GiNaC::expand_funcp_2) (const ex &, const ex &, unsigned)`
- `typedef ex(* GiNaC::derivative_funcp_2) (const ex &, const ex &, unsigned)`
- `typedef ex(* GiNaC::expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)`
- `typedef ex(* GiNaC::power_funcp_2) (const ex &, const ex &, const ex &)`

- [illegible]



- [illegible]



- [illegible]

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (function)
- `template<typename T >`  
`bool GiNaC::is\_the\_function (const ex &x)`

### 7.36.1 Detailed Description

Interface to class of symbolic functions.

### 7.36.2 Macro Definition Documentation

#### 7.36.2.1 DECLARE\_FUNCTION\_1P

```
#define DECLARE_FUNCTION_1P(
 NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 1; \
template< typename T1 > const GiNaC::function NAME(const T1 & p1) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1)); \
}
```

#### 7.36.2.2 DECLARE\_FUNCTION\_2P

```
#define DECLARE_FUNCTION_2P(
 NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 2; \
template< typename T1, typename T2 > const GiNaC::function NAME(const T1 & p1, const T2 & p2) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2)); \
}
```

#### 7.36.2.3 DECLARE\_FUNCTION\_3P

```
#define DECLARE_FUNCTION_3P(
 NAME)
```

##### Value:

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 3; \
template< typename T1, typename T2, typename T3 > const GiNaC::function NAME(const T1 & p1, const T2 & p2, \
 const T3 & p3) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3)); \
}
```

## 7.36.2.4 DECLARE\_FUNCTION\_4P

```
#define DECLARE_FUNCTION_4P (
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 4; \
template< typename T1, typename T2, typename T3, typename T4 > const GiNaC::function NAME(const T1 & p1,
 const T2 & p2, const T3 & p3, const T4 & p4) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3), GiNaC::ex(p4)
); \
}
```

## 7.36.2.5 DECLARE\_FUNCTION\_5P

```
#define DECLARE_FUNCTION_5P (
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 5; \
template< typename T1, typename T2, typename T3, typename T4, typename T5 > const GiNaC::function NAME(
 const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5)); \
}
```

## 7.36.2.6 DECLARE\_FUNCTION\_6P

```
#define DECLARE_FUNCTION_6P (
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 6; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 > const
 GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
 T6 & p6) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6)); \
}
```

## 7.36.2.7 DECLARE\_FUNCTION\_7P

```
#define DECLARE_FUNCTION_7P (
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 7; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 > const
 GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
 T6 & p6, const T7 & p7) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7)); \
}
```

### 7.36.2.8 DECLARE\_FUNCTION\_8P

```
#define DECLARE_FUNCTION_8P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 8; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8 > const GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4,
 const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8)); \
}
```

### 7.36.2.9 DECLARE\_FUNCTION\_9P

```
#define DECLARE_FUNCTION_9P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 9; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9 > const GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3,
 const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9)); \
}
```

### 7.36.2.10 DECLARE\_FUNCTION\_10P

```
#define DECLARE_FUNCTION_10P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 10; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9, typename T10 > const GiNaC::function NAME(const T1 & p1, const T2 & p2,
 const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 & p8, const T9 &
 p9, const T10 & p10) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
 GiNaC::ex(p10)); \
}
```

### 7.36.2.11 DECLARE\_FUNCTION\_11P

```
#define DECLARE_FUNCTION_11P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 11; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9, typename T10, typename T11 > const GiNaC::function NAME(const T1 & p1,
 const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7, const T8 &
 p8, const T9 & p9, const T10 & p10, const T11 & p11) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
 GiNaC::ex(p10), GiNaC::ex(p11)); \
}
```

## 7.36.2.12 DECLARE\_FUNCTION\_12P

```
#define DECLARE_FUNCTION_12P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 12; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9, typename T10, typename T11, typename T12 > const GiNaC::function NAME(const
 T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const T6 & p6, const T7 & p7,
 const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 & p12) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
 GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12)); \
}
```

## 7.36.2.13 DECLARE\_FUNCTION\_13P

```
#define DECLARE_FUNCTION_13P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 13; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9, typename T10, typename T11, typename T12, typename T13 > const
 GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
 T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &
 p12, const T13 & p13) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
 GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13)); \
}
```

## 7.36.2.14 DECLARE\_FUNCTION\_14P

```
#define DECLARE_FUNCTION_14P(
 NAME)
```

**Value:**

```
class NAME##_SERIAL { public: static unsigned serial; }; \
const unsigned NAME##_NPARAMS = 14; \
template< typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7,
 typename T8, typename T9, typename T10, typename T11, typename T12, typename T13, typename T14 > const
 GiNaC::function NAME(const T1 & p1, const T2 & p2, const T3 & p3, const T4 & p4, const T5 & p5, const
 T6 & p6, const T7 & p7, const T8 & p8, const T9 & p9, const T10 & p10, const T11 & p11, const T12 &
 p12, const T13 & p13, const T14 & p14) { \
 return GiNaC::function(NAME##_SERIAL::serial, GiNaC::ex(p1), GiNaC::ex(p2), GiNaC::ex(p3),
 GiNaC::ex(p4), GiNaC::ex(p5), GiNaC::ex(p6), GiNaC::ex(p7), GiNaC::ex(p8), GiNaC::ex(p9),
 GiNaC::ex(p10), GiNaC::ex(p11), GiNaC::ex(p12), GiNaC::ex(p13), GiNaC::ex(p14)); \
}
```

## 7.36.2.15 REGISTER\_FUNCTION

```
#define REGISTER_FUNCTION(
 NAME,
 OPT)
```

**Value:**

```
unsigned NAME##_SERIAL::serial = \
 GiNaC::function::register_new(GiNaC::function_options(#NAME, NAME##_NPARAMS).OPT);
```

### 7.36.2.16 is\_ex\_the\_function

```
#define is_ex_the_function(
 OBJ,
 FUNCNAME) (GiNaC::is_the_function<FUNCNAME##_SERIAL>(OBJ))
```

Referenced by [GiNaC::abs\\_eval\(\)](#), [GiNaC::cos\\_eval\(\)](#), [GiNaC::cosh\\_eval\(\)](#), [GiNaC::exp\\_eval\(\)](#), [GiNaC::is\\_order\\_function\(\)](#), [GiNaC::log\\_eval\(\)](#), [GiNaC::replace\\_with\\_symbol\(\)](#), [GiNaC::sin\\_eval\(\)](#), [GiNaC::sinh\\_eval\(\)](#), [GiNaC::tan\\_eval\(\)](#), and [GiNaC::tanh\\_eval\(\)](#).

## 7.37 ginac.h File Reference

This include file includes all other public [GiNaC](#) headers.

```
#include "version.h"
#include "basic.h"
#include "ex.h"
#include "normal.h"
#include "archive.h"
#include "print.h"
#include "constant.h"
#include "fail.h"
#include "integral.h"
#include "lst.h"
#include "matrix.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "structure.h"
#include "symbol.h"
#include "pseries.h"
#include "wildcard.h"
#include "symmetry.h"
#include "expair.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "exprseq.h"
#include "function.h"
#include "ncmul.h"
#include "inifcns.h"
#include "fderivative.h"
#include "operators.h"
#include "hash_map.h"
#include "idx.h"
#include "indexed.h"
#include "tensor.h"
#include "color.h"
#include "clifford.h"
#include "factor.h"
#include "integration_kernel.h"
#include "excompiler.h"
#include "parser.h"
```

### 7.37.1 Detailed Description

This include file includes all other public [GiNaC](#) headers.

## 7.38 hash\_map.h File Reference

Replacement for map<> using hash tables.

```
#include <unordered_map>
```

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- `template<typename T, class Hash = std::hash<ex>, class KeyEqual = std::equal_to<ex>, class Allocator = std::allocator<std::pair<const ex, T>>>>`  
using [GiNaC::exhashmap](#) = std::unordered\_map<[ex](#), T, Hash, KeyEqual, Allocator>

### 7.38.1 Detailed Description

Replacement for map<> using hash tables.

## 7.39 hash\_seed.h File Reference

Type-specific hash seed.

```
#include <typeinfo>
#include <cstring>
#include "utils.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static unsigned [GiNaC::make\\_hash\\_seed](#) (const std::type\_info &tinfo)  
*We need a hash function which gives different values for objects of different types.*

### 7.39.1 Detailed Description

Type-specific hash seed.

## 7.40 idx.cpp File Reference

Implementation of [GiNaC](#)'s indices.

```
#include "idx.h"
#include "symbol.h"
#include "lst.h"
#include "relational.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <sstream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([idx](#), [basic](#), [print\\_func< print\\_context >\(&idx::do\\_print\)](#), [print\\_func< print\\_latex >\(&idx::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&idx::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&idx::do\\_print\\_tree\)](#)) [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)([varidx](#)
- [GiNaC::print\\_func< print\\_context > \(&varidx::do\\_print\)](#), [print\\_func< print\\_latex >\(&varidx](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([idx](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([varidx](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([spinidx](#))
- [bool GiNaC::is\\_dummy\\_pair](#) (const [idx](#) &i1, const [idx](#) &i2)  
*Check whether two indices form a dummy pair.*
- [bool GiNaC::is\\_dummy\\_pair](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Check whether two expressions form a dummy index pair.*
- [void GiNaC::find\\_free\\_and\\_dummy](#) ([exvector::const\\_iterator](#) it, [exvector::const\\_iterator](#) itend, [exvector](#) &out←  
\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- [ex GiNaC::minimal\\_dim](#) (const [ex](#) &dim1, const [ex](#) &dim2)  
*Return the minimum of two index dimensions.*

### Variables

- [GiNaC::idx](#)

### 7.40.1 Detailed Description

Implementation of [GiNaC](#)'s indices.

## 7.41 idx.h File Reference

Interface to [GiNaC](#)'s indices.

```
#include "ex.h"
#include "numeric.h"
```

### Classes

- class [GiNaC::idx](#)  
*This class holds one index of an indexed object.*
- class [GiNaC::varidx](#)  
*This class holds an index with a variance (co- or contravariant).*
- class [GiNaC::spinidx](#)  
*This class holds a spinor index that can be dotted or undotted and that also has a variance.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([idx](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([varidx](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinidx](#))
- bool [GiNaC::is\\_dummy\\_pair](#) (const [idx](#) &i1, const [idx](#) &i2)  
*Check whether two indices form a dummy pair.*
- bool [GiNaC::is\\_dummy\\_pair](#) (const [ex](#) &e1, const [ex](#) &e2)  
*Check whether two expressions form a dummy index pair.*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (exvector::const\_iterator it, exvector::const\_iterator itend, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_free\\_and\\_dummy](#) (const [exvector](#) &v, [exvector](#) &out\_free, [exvector](#) &out\_dummy)  
*Given a vector of indices, split them into two vectors, one containing the free indices, the other containing the dummy indices (numeric indices are neither free nor dummy ones).*
- void [GiNaC::find\\_dummy\\_indices](#) (const [exvector](#) &v, [exvector](#) &out\_dummy)  
*Given a vector of indices, find the dummy indices.*
- size\_t [GiNaC::count\\_dummy\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- size\_t [GiNaC::count\\_free\\_indices](#) (const [exvector](#) &v)  
*Count the number of dummy index pairs in an index vector.*
- [ex](#) [GiNaC::minimal\\_dim](#) (const [ex](#) &dim1, const [ex](#) &dim2)  
*Return the minimum of two index dimensions.*

### 7.41.1 Detailed Description

Interface to [GiNaC](#)'s indices.

## 7.42 indexed.cpp File Reference

Implementation of [GiNaC](#)'s indexed expressions.

```
#include "indexed.h"
#include "idx.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "power.h"
#include "relational.h"
#include "symmetry.h"
#include "operators.h"
#include "lst.h"
#include "archive.h"
#include "symbol.h"
#include "utils.h"
#include "integral.h"
#include "matrix.h"
#include "inifcns.h"
#include <iostream>
#include <limits>
#include <sstream>
#include <stdexcept>
```

### Classes

- struct [GiNaC::idx\\_is\\_equal\\_ignore\\_dim](#)
- struct [GiNaC::is\\_summation\\_idx](#)
- struct [GiNaC::ex\\_base\\_is\\_less](#)
- class [GiNaC::terminfo](#)

*This structure stores the original and symmetrized versions of terms obtained during the simplification of sums.*

- class [GiNaC::terminfo\\_is\\_less](#)
- class [GiNaC::symminfo](#)

*This structure stores the individual symmetrized terms obtained during the simplification of sums.*

- class [GiNaC::symminfo\\_is\\_less\\_by\\_symmterm](#)
- class [GiNaC::symminfo\\_is\\_less\\_by\\_orig](#)

### Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`indexed`, `exprseq`, `print_func< print_context >(&indexed::do_print)`, `print_func< print_latex >(&indexed::do_print_latex)`, `print_func< print_tree >(&indexed::do_print_tree)`) `indexed`
- `GiNaC::GINAC_BIND_UNARCHIVER` (`indexed`)
- static bool `GiNaC::indices_consistent` (const `exvector` &v1, const `exvector` &v2)  
*Check whether two sorted index vectors are consistent (i.e.*
- template<class T >  
`size_t GiNaC::number_of_type` (const `exvector` &v)
- template<class T >  
`static ex GiNaC::rename_dummy_indices` (const `ex` &e, `exvector` &global\_dummy\_indices, `exvector` &local\_↵  
`_dummy_indices`)  
*Rename dummy indices in an expression.*
- static void `GiNaC::find_variant_indices` (const `exvector` &v, `exvector` &variant\_indices)  
*Given a set of indices, extract those of class varidx.*
- bool `GiNaC::reposition_dummy_indices` (`ex` &e, `exvector` &variant\_dummy\_indices, `exvector` &moved\_↵  
`indices`)  
*Raise/lower dummy indices in a single indexed objects to canonicalize their variance.*
- static void `GiNaC::product_to_exvector` (const `ex` &e, `exvector` &v, bool &non\_commutative)
- template<class T >  
`ex GiNaC::idx_symmetrization` (const `ex` &r, const `exvector` &local\_dummy\_indices)
- `ex GiNaC::simplify_indexed` (const `ex` &e, `exvector` &free\_indices, `exvector` &dummy\_indices, const  
`scalar_products` &sp)  
*Simplify indexed expression, return list of free indices.*
- `ex GiNaC::simplify_indexed_product` (const `ex` &e, `exvector` &free\_indices, `exvector` &dummy\_indices, const  
`scalar_products` &sp)  
*Simplify product of indexed expressions (commutative, noncommutative and simple squares), return list of free in-  
indices.*
- bool `GiNaC::hasindex` (const `ex` &x, const `ex` &sym)
- `exvector GiNaC::get_all_dummy_indices_safely` (const `ex` &e)  
*More reliable version of the form.*
- `exvector GiNaC::get_all_dummy_indices` (const `ex` &e)  
*Returns all dummy indices from the exvector.*
- `lst GiNaC::rename_dummy_indices_uniquely` (const `exvector` &va, const `exvector` &vb)  
*Similar to above, where va and vb are the same and the return value is a list of two lists for substitution in b.*
- `ex GiNaC::rename_dummy_indices_uniquely` (const `exvector` &va, const `exvector` &vb, const `ex` &b)  
*Same as above, where va and vb contain the indices of a and b and are sorted.*
- `ex GiNaC::rename_dummy_indices_uniquely` (const `ex` &a, const `ex` &b)  
*Returns b with all dummy indices, which are common with a, renamed.*
- `ex GiNaC::rename_dummy_indices_uniquely` (`exvector` &va, const `ex` &b, bool modify\_va=false)  
*Returns b with all dummy indices, which are listed in va, renamed if modify\_va is set to TRUE all dummy indices of b  
will be appended to va.*
- `ex GiNaC::expand_dummy_sum` (const `ex` &e, bool subs\_idx=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimen-  
sionality of the dummy index is a nonnegative integer.*

### 7.42.1 Detailed Description

Implementation of `GiNaC`'s indexed expressions.

## 7.43 indexed.h File Reference

Interface to [GiNaC](#)'s indexed expressions.

```
#include "exprseq.h"
#include "wildcard.h"
#include <map>
```

### Classes

- class [GiNaC::indexed](#)  
*This class holds an indexed expression.*
- class [GiNaC::spmapkey](#)
- class [GiNaC::scalar\\_products](#)  
*Helper class for storing information about known scalar products which are to be automatically replaced by [simplify\\_indexed\(\)](#).*

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef std::map< [spmapkey](#), [ex](#) > [GiNaC::spmap](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([indexed](#))
- [exvector](#) [GiNaC::get\\_all\\_dummy\\_indices](#) (const [ex](#) &[e](#))  
*Returns all dummy indices from the exvector.*
- [exvector](#) [GiNaC::get\\_all\\_dummy\\_indices\\_safely](#) (const [ex](#) &[e](#))  
*More reliable version of the form.*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) ([exvector](#) &[va](#), const [ex](#) &[b](#), bool [modify\\_va](#)=false)  
*Returns [b](#) with all dummy indices, which are listed in [va](#), renamed if [modify\\_va](#) is set to TRUE all dummy indices of [b](#) will be appended to [va](#).*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [ex](#) &[a](#), const [ex](#) &[b](#))  
*Returns [b](#) with all dummy indices, which are common with [a](#), renamed.*
- [ex](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &[va](#), const [exvector](#) &[vb](#), const [ex](#) &[b](#))  
*Same as above, where [va](#) and [vb](#) contain the indices of [a](#) and [b](#) and are sorted.*
- [lst](#) [GiNaC::rename\\_dummy\\_indices\\_uniquely](#) (const [exvector](#) &[va](#), const [exvector](#) &[vb](#))  
*Similar to above, where [va](#) and [vb](#) are the same and the return value is a list of two lists for substitution in [b](#).*
- [ex](#) [GiNaC::expand\\_dummy\\_sum](#) (const [ex](#) &[e](#), bool [subs\\_idx](#)=false)  
*This function returns the given expression with expanded sums for all dummy index summations, where the dimensionality of the dummy index is a nonnegative integer.*

### 7.43.1 Detailed Description

Interface to [GiNaC](#)'s indexed expressions.

## 7.44 inifcns.cpp File Reference

Implementation of [GiNaC](#)'s initially known functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "lst.h"
#include "fderivative.h"
#include "matrix.h"
#include "mul.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "pseries.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Classes

- class [GiNaC::symbolset](#)

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::conjugate\\_evalf](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_eval](#) (const [ex](#) &arg)
- static void [GiNaC::conjugate\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex](#) [GiNaC::conjugate\\_conjugate](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)
- static [ex](#) [GiNaC::conjugate\\_real\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::conjugate\\_imag\\_part](#) (const [ex](#) &arg)
- static bool [GiNaC::func\\_arg\\_info](#) (const [ex](#) &arg, unsigned inf)
- static bool [GiNaC::conjugate\\_info](#) (const [ex](#) &arg, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) (conjugate\_function, eval\_func([conjugate\\_eval](#)). evalf\_func([conjugate\\_evalf](#)).  
expl\_derivative\_func([conjugate\\_expl\\_derivative](#)). info\_func([conjugate\\_info](#)). print\_func< [print\\_latex](#)  
>([conjugate\\_print\\_latex](#)). conjugate\_func([conjugate\\_conjugate](#)). real\_part\_func([conjugate\\_real\\_part](#)).  
imag\_part\_func([conjugate\\_imag\\_part](#)). set\_name("conjugate","conjugate"))
- static [ex](#) [GiNaC::real\\_part\\_evalf](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_eval](#) (const [ex](#) &arg)
- static void [GiNaC::real\\_part\\_print\\_latex](#) (const [ex](#) &arg, const [print\\_context](#) &c)
- static [ex](#) [GiNaC::real\\_part\\_conjugate](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_real\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_imag\\_part](#) (const [ex](#) &arg)
- static [ex](#) [GiNaC::real\\_part\\_expl\\_derivative](#) (const [ex](#) &arg, const [symbol](#) &s)

- `GiNaC::REGISTER_FUNCTION` (`real_part_function`, `eval_func(real_part_eval)`. `evalf_func(real_part_evalf)`. `expl_derivative_func(real_part_expl_derivative)`. `print_func< print_latex >(real_part_print_latex)`. `conjugate_func(real_part_conjugate)`. `real_part_func(real_part_real_part)`. `imag_part_func(real_part_imag_part)`. `set_name("real_part","real_part")`)
- static `ex` `GiNaC::imag_part_evalf` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_eval` (`const ex &arg`)
- static void `GiNaC::imag_part_print_latex` (`const ex &arg`, `const print_context &c`)
- static `ex` `GiNaC::imag_part_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_real_part` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::imag_part_expl_derivative` (`const ex &arg`, `const symbol &s`)
- `GiNaC::REGISTER_FUNCTION` (`imag_part_function`, `eval_func(imag_part_eval)`. `evalf_func(imag_part_evalf)`. `expl_derivative_func(imag_part_expl_derivative)`. `print_func< print_latex >(imag_part_print_latex)`. `conjugate_func(imag_part_conjugate)`. `real_part_func(imag_part_real_part)`. `imag_part_func(imag_part_imag_part)`. `set_name("imag_part","imag_part")`)
- static `ex` `GiNaC::abs_evalf` (`const ex &arg`)
- static `ex` `GiNaC::abs_eval` (`const ex &arg`)
- static `ex` `GiNaC::abs_expand` (`const ex &arg`, `unsigned options`)
- static `ex` `GiNaC::abs_expl_derivative` (`const ex &arg`, `const symbol &s`)
- static void `GiNaC::abs_print_latex` (`const ex &arg`, `const print_context &c`)
- static void `GiNaC::abs_print_csrc_float` (`const ex &arg`, `const print_context &c`)
- static `ex` `GiNaC::abs_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::abs_real_part` (`const ex &arg`)
- static `ex` `GiNaC::abs_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::abs_power` (`const ex &arg`, `const ex &exp`)
- bool `GiNaC::abs_info` (`const ex &arg`, `unsigned inf`)
- `GiNaC::REGISTER_FUNCTION` (`abs`, `eval_func(abs_eval)`. `evalf_func(abs_evalf)`. `expand_func(abs_expand)`. `expl_derivative_func(abs_expl_derivative)`. `info_func(abs_info)`. `print_func< print_latex >(abs_print_latex)`. `print_func< print_csrc_float >(abs_print_csrc_float)`. `print_func< print_csrc_double >(abs_print_csrc_double)`. `conjugate_func(abs_conjugate)`. `real_part_func(abs_real_part)`. `imag_part_func(abs_imag_part)`. `power_func(abs_power)`)
- static `ex` `GiNaC::step_evalf` (`const ex &arg`)
- static `ex` `GiNaC::step_eval` (`const ex &arg`)
- static `ex` `GiNaC::step_series` (`const ex &arg`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::step_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::step_real_part` (`const ex &arg`)
- static `ex` `GiNaC::step_imag_part` (`const ex &arg`)
- `GiNaC::REGISTER_FUNCTION` (`step`, `eval_func(step_eval)`. `evalf_func(step_evalf)`. `series_func(step_series)`. `conjugate_func(step_conjugate)`. `real_part_func(step_real_part)`. `imag_part_func(step_imag_part)`)
- static `ex` `GiNaC::csgn_evalf` (`const ex &arg`)
- static `ex` `GiNaC::csgn_eval` (`const ex &arg`)
- static `ex` `GiNaC::csgn_series` (`const ex &arg`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::csgn_conjugate` (`const ex &arg`)
- static `ex` `GiNaC::csgn_real_part` (`const ex &arg`)
- static `ex` `GiNaC::csgn_imag_part` (`const ex &arg`)
- static `ex` `GiNaC::csgn_power` (`const ex &arg`, `const ex &exp`)
- `GiNaC::REGISTER_FUNCTION` (`csgn`, `eval_func(csgn_eval)`. `evalf_func(csgn_evalf)`. `series_func(csgn_series)`. `conjugate_func(csgn_conjugate)`. `real_part_func(csgn_real_part)`. `imag_part_func(csgn_imag_part)`. `power_func(csgn_power)`)
- static `ex` `GiNaC::eta_evalf` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_eval` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_series` (`const ex &x`, `const ex &y`, `const relational &rel`, `int order`, `unsigned options`)
- static `ex` `GiNaC::eta_conjugate` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_real_part` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::eta_imag_part` (`const ex &x`, `const ex &y`)

- `GiNaC::REGISTER_FUNCTION` (`eta`, `eval_func(eta_eval)`). `evalf_func(eta_evalf)`. `series_func(eta_series)`. `latex_name("\\eta")`. `set_symmetry(sy_symm(0, 1))`. `conjugate_func(eta_conjugate)`. `real_part_func(eta_real_part)`. `imag_part_func(eta_imag_part)`)
- static `ex` `GiNaC::Li2_evalf` (`const ex &x`)
- static `ex` `GiNaC::Li2_eval` (`const ex &x`)
- static `ex` `GiNaC::Li2_deriv` (`const ex &x`, unsigned `deriv_param`)
- static `ex` `GiNaC::Li2_series` (`const ex &x`, `const relational &rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::Li2_conjugate` (`const ex &x`)
- `GiNaC::REGISTER_FUNCTION` (`Li2`, `eval_func(Li2_eval)`). `evalf_func(Li2_evalf)`. `derivative_func(Li2_deriv)`. `series_func(Li2_series)`. `conjugate_func(Li2_conjugate)`. `latex_name("\\mathrm{Li}_2")`)
- static `ex` `GiNaC::Li3_eval` (`const ex &x`)
- `GiNaC::REGISTER_FUNCTION` (`Li3`, `eval_func(Li3_eval)`). `latex_name("\\mathrm{Li}_3")`)
- static `ex` `GiNaC::zetaderiv_eval` (`const ex &n`, `const ex &x`)
- static `ex` `GiNaC::zetaderiv_deriv` (`const ex &n`, `const ex &x`, unsigned `deriv_param`)
- `GiNaC::REGISTER_FUNCTION` (`zetaderiv`, `eval_func(zetaderiv_eval)`). `derivative_func(zetaderiv_deriv)`. `latex_name("\\zeta^{\\prime}")`)
- static `ex` `GiNaC::factorial_evalf` (`const ex &x`)
- static `ex` `GiNaC::factorial_eval` (`const ex &x`)
- static void `GiNaC::factorial_print_dflt_latex` (`const ex &x`, `const print_context &c`)
- static `ex` `GiNaC::factorial_conjugate` (`const ex &x`)
- static `ex` `GiNaC::factorial_real_part` (`const ex &x`)
- static `ex` `GiNaC::factorial_imag_part` (`const ex &x`)
- `GiNaC::REGISTER_FUNCTION` (`factorial`, `eval_func(factorial_eval)`). `evalf_func(factorial_evalf)`. `print_func< print_dflt >(factorial_print_dflt_latex)`. `print_func< print_latex >(factorial_print_dflt_latex)`. `conjugate_func(factorial_conjugate)`. `real_part_func(factorial_real_part)`. `imag_part_func(factorial_imag_part)`)
- static `ex` `GiNaC::binomial_evalf` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::binomial_sym` (`const ex &x`, `const numeric &y`)
- static `ex` `GiNaC::binomial_eval` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::binomial_conjugate` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::binomial_real_part` (`const ex &x`, `const ex &y`)
- static `ex` `GiNaC::binomial_imag_part` (`const ex &x`, `const ex &y`)
- `GiNaC::REGISTER_FUNCTION` (`binomial`, `eval_func(binomial_eval)`). `evalf_func(binomial_evalf)`. `conjugate_func(binomial_conjugate)`. `real_part_func(binomial_real_part)`. `imag_part_func(binomial_imag_part)`)
- static `ex` `GiNaC::Order_eval` (`const ex &x`)
- static `ex` `GiNaC::Order_series` (`const ex &x`, `const relational &r`, int `order`, unsigned `options`)
- static `ex` `GiNaC::Order_conjugate` (`const ex &x`)
- static `ex` `GiNaC::Order_real_part` (`const ex &x`)
- static `ex` `GiNaC::Order_imag_part` (`const ex &x`)
- static `ex` `GiNaC::Order_power` (`const ex &x`, `const ex &e`)
- static `ex` `GiNaC::Order_expl_derivative` (`const ex &arg`, `const symbol &s`)
- `GiNaC::REGISTER_FUNCTION` (`Order`, `eval_func(Order_eval)`). `series_func(Order_series)`. `latex_name("\\mathcal{O}")`. `expl_derivative_func(Order_expl_derivative)`. `conjugate_func(Order_conjugate)`. `real_part_func(Order_real_part)`. `imag_part_func(Order_imag_part)`. `power_func(Order_power)`)
- `ex` `GiNaC::lsolve` (`const ex &eqns`, `const ex &symbols`, unsigned `options=solve_algo::automatic`)

*Factorial function.*

- `const numeric` `GiNaC::fsolve` (`const ex &f`, `const symbol &x`, `const numeric &x1`, `const numeric &x2`)

*Find a real root of real-valued function f(x) numerically within a given interval.*

## Variables

- unsigned `GiNaC::force_include_tgamma` = `tgamma_SERIAL::serial`
- unsigned `GiNaC::force_include_zeta1` = `zeta1_SERIAL::serial`

### 7.44.1 Detailed Description

Implementation of [GiNaC](#)'s initially known functions.

## 7.45 inifcns.h File Reference

Interface to [GiNaC](#)'s initially known functions.

```
#include "numeric.h"
#include "function.h"
#include "ex.h"
```

### Classes

- class [GiNaC::zeta1\\_SERIAL](#)  
*Complex conjugate.*
- class [GiNaC::zeta2\\_SERIAL](#)  
*Alternating Euler sum or colored MZV.*
- class [GiNaC::G2\\_SERIAL](#)  
*Generalized multiple polylogarithm.*
- class [GiNaC::G3\\_SERIAL](#)  
*Generalized multiple polylogarithm with explicit imaginary parts.*
- class [GiNaC::psi1\\_SERIAL](#)  
*Polylogarithm and multiple polylogarithm.*
- class [GiNaC::psi2\\_SERIAL](#)  
*Derivatives of Psi-function (aka polygamma-functions).*
- class [GiNaC::iterated\\_integral2\\_SERIAL](#)  
*Complete elliptic integral of the first kind.*
- class [GiNaC::iterated\\_integral3\\_SERIAL](#)  
*Iterated integral with explicit truncation.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- `template<typename T1 >`  
`function GiNaC::zeta (const T1 &p1)`
- `template<typename T1 , typename T2 >`  
`function GiNaC::zeta (const T1 &p1, const T2 &p2)`
- `template<> bool GiNaC::is\_the\_function< zeta\_SERIAL > (const ex &x)`
- `template<typename T1 , typename T2 >`  
`function GiNaC::G (const T1 &x, const T2 &y)`
- `template<typename T1 , typename T2 , typename T3 >`  
`function GiNaC::G (const T1 &x, const T2 &s, const T3 &y)`
- `template<> bool GiNaC::is\_the\_function< G\_SERIAL > (const ex &x)`

- `template<typename T1 >`  
`function GiNaC::psi` (const T1 &p1)
- `template<typename T1 , typename T2 >`  
`function GiNaC::psi` (const T1 &p1, const T2 &p2)
- `template<> bool GiNaC::is_the_function< psi_SERIAL >` (const `ex` &x)
- `template<typename T1 , typename T2 >`  
`function GiNaC::iterated_integral` (const T1 &kernel\_lst, const T2 &lambda)
- `template<typename T1 , typename T2 , typename T3 >`  
`function GiNaC::iterated_integral` (const T1 &kernel\_lst, const T2 &lambda, const T3 &N\_trunc)
- `template<> bool GiNaC::is_the_function< iterated_integral_SERIAL >` (const `ex` &x)
- `ex GiNaC::lsolve` (const `ex` &eqns, const `ex` &symbols, unsigned `options=solve_algo::automatic`)  
*Factorial function.*
- const `numeric GiNaC::fsolve` (const `ex` &f, const `symbol` &x, const `numeric` &x1, const `numeric` &x2)  
*Find a real root of real-valued function f(x) numerically within a given interval.*
- `bool GiNaC::is_order_function` (const `ex` &e)  
*Check whether a function is the Order (O(n)) function.*
- `ex GiNaC::convert_H_to_Li` (const `ex` &parameterlst, const `ex` &arg)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*

### 7.45.1 Detailed Description

Interface to GiNaC's initially known functions.

## 7.46 inifcns\_elliptic.cpp File Reference

Implementation of some special functions related to elliptic curves.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include "integration_kernel.h"
#include "utils_multi_iterator.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- namespace `GiNaC`

## Functions

- static `ex` `GiNaC::EllipticK_evalf` (const `ex` &`k`)
- static `ex` `GiNaC::EllipticK_eval` (const `ex` &`k`)
- static `ex` `GiNaC::EllipticK_deriv` (const `ex` &`k`, unsigned `deriv_param`)
- static `ex` `GiNaC::EllipticK_series` (const `ex` &`k`, const `relational` &`rel`, int `order`, unsigned `options`)
- static void `GiNaC::EllipticK_print_latex` (const `ex` &`k`, const `print_context` &`c`)
- `GiNaC::REGISTER_FUNCTION` (`EllipticK`, `evalf_func`(`EllipticK_evalf`). `eval_func`(`EllipticK_eval`). `derivative`↔  
\_func(`EllipticK_deriv`). `series_func`(`EllipticK_series`). `print_func`< `print_latex` >( `EllipticK_print_latex`). `do_`↔  
`not_evalf_params`())
- static `ex` `GiNaC::EllipticE_evalf` (const `ex` &`k`)
- static `ex` `GiNaC::EllipticE_eval` (const `ex` &`k`)
- static `ex` `GiNaC::EllipticE_deriv` (const `ex` &`k`, unsigned `deriv_param`)
- static `ex` `GiNaC::EllipticE_series` (const `ex` &`k`, const `relational` &`rel`, int `order`, unsigned `options`)
- static void `GiNaC::EllipticE_print_latex` (const `ex` &`k`, const `print_context` &`c`)
- `GiNaC::REGISTER_FUNCTION` (`EllipticE`, `evalf_func`(`EllipticE_evalf`). `eval_func`(`EllipticE_eval`). `derivative`↔  
\_func(`EllipticE_deriv`). `series_func`(`EllipticE_series`). `print_func`< `print_latex` >( `EllipticE_print_latex`). `do_`↔  
`not_evalf_params`())
- static `ex` `GiNaC::iterated_integral_evalf_impl` (const `ex` &`kernel_lst`, const `ex` &`lambda`, const `ex` &`N_trunc`)
- static `ex` `GiNaC::iterated_integral2_evalf` (const `ex` &`kernel_lst`, const `ex` &`lambda`)
- static `ex` `GiNaC::iterated_integral3_evalf` (const `ex` &`kernel_lst`, const `ex` &`lambda`, const `ex` &`N_trunc`)
- static `ex` `GiNaC::iterated_integral2_eval` (const `ex` &`kernel_lst`, const `ex` &`lambda`)
- static `ex` `GiNaC::iterated_integral3_eval` (const `ex` &`kernel_lst`, const `ex` &`lambda`, const `ex` &`N_trunc`)

### 7.46.1 Detailed Description

Implementation of some special functions related to elliptic curves.

The functions are: complete elliptic integral of the first kind `EllipticK(k)` complete elliptic integral of the second kind `EllipticE(k)` iterated integral `iterated_integral(a,y)` or `iterated_integral(a,y,N_trunc)`

Some remarks:

- All formulae used can be looked up in the following publication: [WW] Numerical evaluation of iterated integrals related to elliptic Feynman integrals, M.Walden, S.Weinzierl, arXiv:2010.05271
- When these routines and methods are used for scientific work that leads to publication in a scientific journal, please refer to this program as : M.Walden, S.Weinzierl, "Numerical evaluation of iterated integrals related to elliptic Feynman integrals", arXiv:2010.05271
- As these routines build on the core part of `GiNaC`, it is also polite to acknowledge C. Bauer, A. Frink, R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language", J. Symbolic Computations 33, 1 (2002), cs.sc/0004015

## 7.47 inifcns\_gamma.cpp File Reference

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

```
#include "inifcns.h"
#include "constant.h"
#include "pseries.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "symmetry.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::lgamma\\_evalf](#) (const [ex](#) &[x](#))
- static [ex](#) [GiNaC::lgamma\\_eval](#) (const [ex](#) &[x](#))
 

*Evaluation of  $\lgamma(x)$ , the natural logarithm of the Gamma function.*
- static [ex](#) [GiNaC::lgamma\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex](#) [GiNaC::lgamma\\_series](#) (const [ex](#) &[arg](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- static [ex](#) [GiNaC::lgamma\\_conjugate](#) (const [ex](#) &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) ([lgamma](#), [eval\\_func](#)([lgamma\\_eval](#)). [evalf\\_func](#)([lgamma\\_evalf](#)). [derivative\\_↔\\_func](#)([lgamma\\_deriv](#)). [series\\_func](#)([lgamma\\_series](#)). [conjugate\\_func](#)([lgamma\\_conjugate](#)). [latex\\_name](#)("\\log \\Gamma"))
- static [ex](#) [GiNaC::tgamma\\_evalf](#) (const [ex](#) &[x](#))
- static [ex](#) [GiNaC::tgamma\\_eval](#) (const [ex](#) &[x](#))
 

*Evaluation of  $tgamma(x)$ , the true Gamma function.*
- static [ex](#) [GiNaC::tgamma\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex](#) [GiNaC::tgamma\\_series](#) (const [ex](#) &[arg](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- static [ex](#) [GiNaC::tgamma\\_conjugate](#) (const [ex](#) &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) ([tgamma](#), [eval\\_func](#)([tgamma\\_eval](#)). [evalf\\_func](#)([tgamma\\_evalf](#)). [derivative\\_↔\\_func](#)([tgamma\\_deriv](#)). [series\\_func](#)([tgamma\\_series](#)). [conjugate\\_func](#)([tgamma\\_conjugate](#)). [latex\\_↔\\_name](#)("\\Gamma"))
- static [ex](#) [GiNaC::beta\\_evalf](#) (const [ex](#) &[x](#), const [ex](#) &[y](#))
- static [ex](#) [GiNaC::beta\\_eval](#) (const [ex](#) &[x](#), const [ex](#) &[y](#))
- static [ex](#) [GiNaC::beta\\_deriv](#) (const [ex](#) &[x](#), const [ex](#) &[y](#), unsigned [deriv\\_param](#))
- static [ex](#) [GiNaC::beta\\_series](#) (const [ex](#) &[arg1](#), const [ex](#) &[arg2](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- [GiNaC::REGISTER\\_FUNCTION](#) ([beta](#), [eval\\_func](#)([beta\\_eval](#)). [evalf\\_func](#)([beta\\_evalf](#)). [derivative\\_↔\\_func](#)([beta\\_deriv](#)). [series\\_func](#)([beta\\_series](#)). [latex\\_name](#)("\\mathrm{B}"). [set\\_symmetry](#)([sy\\_symm](#)(0, 1)))
- static [ex](#) [GiNaC::psi1\\_evalf](#) (const [ex](#) &[x](#))
- static [ex](#) [GiNaC::psi1\\_eval](#) (const [ex](#) &[x](#))
 

*Evaluation of digamma-function  $\psi_1(x)$ .*
- static [ex](#) [GiNaC::psi1\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))

- static `ex GiNaC::psi1_series` (const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)
- static `ex GiNaC::psi2_evalf` (const `ex` &n, const `ex` &x)
- static `ex GiNaC::psi2_eval` (const `ex` &n, const `ex` &x)  
*Evaluation of polygamma-function  $\psi(n,x)$ .*
- static `ex GiNaC::psi2_deriv` (const `ex` &n, const `ex` &x, unsigned `deriv_param`)
- static `ex GiNaC::psi2_series` (const `ex` &n, const `ex` &arg, const `relational` &rel, int `order`, unsigned `options`)

### 7.47.1 Detailed Description

Implementation of Gamma-function, Beta-function, Polygamma-functions, and some related stuff.

## 7.48 inifcns\_nstdsums.cpp File Reference

Implementation of some special functions that have a representation as nested sums.

```
#include "inifcns.h"
#include "add.h"
#include "constant.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "operators.h"
#include "power.h"
#include "pseries.h"
#include "relational.h"
#include "symbol.h"
#include "utils.h"
#include "wildcard.h"
#include <cln/cln.h>
#include <sstream>
#include <stdexcept>
#include <vector>
#include <cmath>
```

### Namespaces

- namespace `GiNaC`

### Functions

- static `ex GiNaC::G2_evalf` (const `ex` &x\_, const `ex` &y)
- static `ex GiNaC::G2_eval` (const `ex` &x\_, const `ex` &y)
- static `ex GiNaC::G3_evalf` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex GiNaC::G3_eval` (const `ex` &x\_, const `ex` &s\_, const `ex` &y)
- static `ex GiNaC::Li_evalf` (const `ex` &m\_, const `ex` &x\_)
- static `ex GiNaC::Li_eval` (const `ex` &m\_, const `ex` &x\_)
- static `ex GiNaC::Li_series` (const `ex` &m, const `ex` &x, const `relational` &rel, int `order`, unsigned `options`)
- static `ex GiNaC::Li_deriv` (const `ex` &m\_, const `ex` &x\_, unsigned `deriv_param`)
- static void `GiNaC::Li_print_latex` (const `ex` &m\_, const `ex` &x\_, const `print_context` &c)

- `GiNaC::REGISTER_FUNCTION` (`Li`, `evalf_func(Li_evalf)`. `eval_func(Li_eval)`. `series_func(Li_series)`. `derivative_func(Li_deriv)`. `print_func< print_latex >(Li_print_latex)`. `do_not_evalf_params()`)
- static `ex` `GiNaC::S_evalf` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex` `GiNaC::S_eval` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`)
- static `ex` `GiNaC::S_series` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::S_deriv` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, unsigned `deriv_param`)
- static void `GiNaC::S_print_latex` (const `ex` &`n`, const `ex` &`p`, const `ex` &`x`, const `print_context` &`c`)
- `GiNaC::REGISTER_FUNCTION` (`S`, `evalf_func(S_evalf)`. `eval_func(S_eval)`. `series_func(S_series)`. `derivative_func(S_deriv)`. `print_func< print_latex >(S_print_latex)`. `do_not_evalf_params()`)
- static `ex` `GiNaC::H_evalf` (const `ex` &`x1`, const `ex` &`x2`)
- static `ex` `GiNaC::H_eval` (const `ex` &`m`, const `ex` &`x`)
- static `ex` `GiNaC::H_series` (const `ex` &`m`, const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::H_deriv` (const `ex` &`m`, const `ex` &`x`, unsigned `deriv_param`)
- static void `GiNaC::H_print_latex` (const `ex` &`m`, const `ex` &`x`, const `print_context` &`c`)
- `GiNaC::REGISTER_FUNCTION` (`H`, `evalf_func(H_evalf)`. `eval_func(H_eval)`. `series_func(H_series)`. `derivative_func(H_deriv)`. `print_func< print_latex >(H_print_latex)`. `do_not_evalf_params()`)
- `ex` `GiNaC::convert_H_to_Li` (const `ex` &`parameterlst`, const `ex` &`arg`)  
*Converts a given list containing parameters for H in Remiddi/Vermaseren notation into the corresponding GiNaC functions.*
- static `ex` `GiNaC::zeta1_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::zeta1_eval` (const `ex` &`m`)
- static `ex` `GiNaC::zeta1_deriv` (const `ex` &`m`, unsigned `deriv_param`)
- static void `GiNaC::zeta1_print_latex` (const `ex` &`m`, const `print_context` &`c`)
- static `ex` `GiNaC::zeta2_evalf` (const `ex` &`x`, const `ex` &`s`)
- static `ex` `GiNaC::zeta2_eval` (const `ex` &`m`, const `ex` &`s`)
- static `ex` `GiNaC::zeta2_deriv` (const `ex` &`m`, const `ex` &`s`, unsigned `deriv_param`)
- static void `GiNaC::zeta2_print_latex` (const `ex` &`m`, const `ex` &`s`, const `print_context` &`c`)

### 7.48.1 Detailed Description

Implementation of some special functions that have a representation as nested sums.

The functions are: classical polylogarithm  $\text{Li}(n,x)$  multiple polylogarithm  $\text{Li}(\{m_1,\dots,m_k\},\{x_1,\dots,x_k\})$   $G(\{a_1,\dots,a_k\},y)$  or  $G(\{a_1,\dots,a_k\},\{s_1,\dots,s_k\},y)$  Nielsen's generalized polylogarithm  $S(n,p,x)$  harmonic polylogarithm  $H(m,x)$  or  $H(\{m_1,\dots,m_k\},x)$  multiple zeta value  $\text{zeta}(m)$  or  $\text{zeta}(\{m_1,\dots,m_k\})$  alternating Euler sum  $\text{zeta}(m,s)$  or  $\text{zeta}(\{m_1,\dots,m_k\},\{s_1,\dots,s_k\})$

Some remarks:

- All formulae used can be looked up in the following publications: [Kol] Nielsen's Generalized Polylogarithms, K.S.Kolbig, SIAM J.Math.Anal. 17 (1986), pp. 1232-1258. [Cra] Fast Evaluation of Multiple Zeta Sums, R.E.Crandall, Math.Comp. 67 (1998), pp. 1163-1172. [ReV] Harmonic Polylogarithms, E.Remiddi, J.A. Vermaseren, Int.J.Mod.Phys. A15 (2000), pp. 725-754 [BBB] Special Values of Multiple Polylogarithms, J.Borwein, D.Bradley, D.Broadhurst, P.Lisonek, Trans.Amer.Math.Soc. 353/3 (2001), pp. 907-941 [VSW] Numerical evaluation of multiple polylogarithms, J.Vollinga, S.Weinzierl, hep-ph/0410259
- The order of parameters and arguments of Li and zeta is defined according to the nested sums representation. The parameters for H are understood as in [ReV]. They can be in expanded — only 0, 1 and -1 — or in compactified — a string with zeros in front of 1 or -1 is written as a single number — notation.
- All functions can be numerically evaluated with arguments in the whole complex plane. The parameters for Li, zeta and S must be positive integers. If you want to have an alternating Euler sum, you have to give the signs of the parameters as a second argument s to  $\text{zeta}(m,s)$  containing 1 and -1.

- The calculation of classical polylogarithms is speeded up by using Bernoulli numbers and look-up tables. S uses look-up tables as well. The zeta function applies the algorithms in [Cra] and [BBB] for speed up. Multiple polylogarithms use Hoelder convolution [BBB].
- The functions have no means to do a series expansion into nested sums. To do this, you have to convert these functions into the appropriate objects from the nestedsums library, do the expansion and convert the result back.
- Numerical testing of this implementation has been performed by doing a comparison of results between this software and the commercial M..... 4.1. Multiple zeta values have been checked by means of evaluations into simple zeta values. Harmonic polylogarithms have been checked by comparison to  $S(n,p,x)$  for corresponding parameter combinations and by continuity checks around  $|x|=1$  along with comparisons to corresponding zeta functions. Multiple polylogarithms were checked against H and zeta and by means of shuffle and quasi-shuffle relations.

## 7.49 inifcns\_trans.cpp File Reference

Implementation of transcendental (and trigonometric and hyperbolic) functions.

```
#include "inifcns.h"
#include "ex.h"
#include "constant.h"
#include "add.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "operators.h"
#include "relational.h"
#include "symbol.h"
#include "pseries.h"
#include "utils.h"
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- static [ex](#) [GiNaC::exp\\_evalf](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_eval](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_expand](#) (const [ex](#) &arg, unsigned [options](#))
- static [ex](#) [GiNaC::exp\\_deriv](#) (const [ex](#) &x, unsigned deriv\_param)
- static [ex](#) [GiNaC::exp\\_real\\_part](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_imag\\_part](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_conjugate](#) (const [ex](#) &x)
- static [ex](#) [GiNaC::exp\\_power](#) (const [ex](#) &x, const [ex](#) &a)
- static bool [GiNaC::exp\\_info](#) (const [ex](#) &x, unsigned inf)
- [GiNaC::REGISTER\\_FUNCTION](#) ([exp](#), [eval\\_func](#)([exp\\_eval](#)). [evalf\\_func](#)([exp\\_evalf](#)). [info\\_func](#)([exp\\_info](#)). [expand\\_func](#)([exp\\_expand](#)). [derivative\\_func](#)([exp\\_deriv](#)). [real\\_part\\_func](#)([exp\\_real\\_part](#)). [imag\\_part\\_func](#)([exp\\_imag\\_part](#)). [conjugate\\_func](#)([exp\\_conjugate](#)). [power\\_func](#)([exp\\_power](#)). [latex\\_name](#)("\\exp"))

- static `ex` `GiNaC::log_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::log_eval` (const `ex` &`x`)
- static `ex` `GiNaC::log_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::log_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::log_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::log_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::log_expand` (const `ex` &`arg`, unsigned `options`)
- static `ex` `GiNaC::log_conjugate` (const `ex` &`x`)
- static bool `GiNaC::log_info` (const `ex` &`x`, unsigned `inf`)
- `GiNaC::REGISTER_FUNCTION` (`log`, `eval_func(log_eval)`. `evalf_func(log_evalf)`. `info_func(log_info)`. `expand_func(log_expand)`. `derivative_func(log_deriv)`. `series_func(log_series)`. `real_part_func(log_real_part)`. `imag_part_func(log_imag_part)`. `conjugate_func(log_conjugate)`. `latex_name("\\ln")`)
- static `ex` `GiNaC::sin_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::sin_eval` (const `ex` &`x`)
- static `ex` `GiNaC::sin_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::sin_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::sin_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::sin_conjugate` (const `ex` &`x`)
- static bool `GiNaC::trig_info` (const `ex` &`x`, unsigned `inf`)
- `GiNaC::REGISTER_FUNCTION` (`sin`, `eval_func(sin_eval)`. `evalf_func(sin_evalf)`. `info_func(trig_info)`. `derivative_func(sin_deriv)`. `real_part_func(sin_real_part)`. `imag_part_func(sin_imag_part)`. `conjugate_↵  
func(sin_conjugate)`. `latex_name("\\sin")`)
- static `ex` `GiNaC::cos_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::cos_eval` (const `ex` &`x`)
- static `ex` `GiNaC::cos_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::cos_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::cos_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::cos_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`cos`, `eval_func(cos_eval)`. `info_func(trig_info)`. `evalf_func(cos_evalf)`. `derivative_func(cos_deriv)`. `real_part_func(cos_real_part)`. `imag_part_func(cos_imag_part)`. `conjugate_↵  
func(cos_conjugate)`. `latex_name("\\cos")`)
- static `ex` `GiNaC::tan_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::tan_eval` (const `ex` &`x`)
- static `ex` `GiNaC::tan_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::tan_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::tan_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::tan_series` (const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::tan_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`tan`, `eval_func(tan_eval)`. `evalf_func(tan_evalf)`. `info_func(trig_info)`. `derivative_func(tan_deriv)`. `series_func(tan_series)`. `real_part_func(tan_real_part)`. `imag_part_↵  
func(tan_imag_part)`. `conjugate_func(tan_conjugate)`. `latex_name("\\tan")`)
- static `ex` `GiNaC::asin_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::asin_eval` (const `ex` &`x`)
- static `ex` `GiNaC::asin_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::asin_conjugate` (const `ex` &`x`)
- static bool `GiNaC::asin_info` (const `ex` &`x`, unsigned `inf`)
- `GiNaC::REGISTER_FUNCTION` (`asin`, `eval_func(asin_eval)`. `evalf_func(asin_evalf)`. `info_func(asin_info)`. `derivative_func(asin_deriv)`. `conjugate_func(asin_conjugate)`. `latex_name("\\arcsin")`)
- static `ex` `GiNaC::acos_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::acos_eval` (const `ex` &`x`)
- static `ex` `GiNaC::acos_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::acos_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`acos`, `eval_func(acos_eval)`. `evalf_func(acos_evalf)`. `info_func(asin_info)`. `derivative_func(acos_deriv)`. `conjugate_func(acos_conjugate)`. `latex_name("\\arccos")`)
- static `ex` `GiNaC::atan_evalf` (const `ex` &`x`)

- static `ex` `GiNaC::atan_eval` (const `ex` &`x`)
- static `ex` `GiNaC::atan_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::atan_series` (const `ex` &`arg`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::atan_conjugate` (const `ex` &`x`)
- static bool `GiNaC::atan_info` (const `ex` &`x`, unsigned `inf`)
- `GiNaC::REGISTER_FUNCTION` (`atan`, `eval_func(atan_eval)`. `evalf_func(atan_evalf)`. `info_func(atan_info)`. `derivative_func(atan_deriv)`. `series_func(atan_series)`. `conjugate_func(atan_conjugate)`. `latex_name("\\arctan")`)
- static `ex` `GiNaC::atan2_evalf` (const `ex` &`y`, const `ex` &`x`)
- static `ex` `GiNaC::atan2_eval` (const `ex` &`y`, const `ex` &`x`)
- static `ex` `GiNaC::atan2_deriv` (const `ex` &`y`, const `ex` &`x`, unsigned `deriv_param`)
- static bool `GiNaC::atan2_info` (const `ex` &`y`, const `ex` &`x`, unsigned `inf`)
- `GiNaC::REGISTER_FUNCTION` (`atan2`, `eval_func(atan2_eval)`. `evalf_func(atan2_evalf)`. `info_func(atan2_info)`. `evalf_func(atan2_evalf)`. `derivative_func(atan2_deriv)`)
- static `ex` `GiNaC::sinh_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::sinh_eval` (const `ex` &`x`)
- static `ex` `GiNaC::sinh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::sinh_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::sinh_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::sinh_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`sinh`, `eval_func(sinh_eval)`. `evalf_func(sinh_evalf)`. `info_func(atan_info)`. `derivative_func(sinh_deriv)`. `real_part_func(sinh_real_part)`. `imag_part_func(sinh_imag_part)`. `conjugate_func(sinh_conjugate)`. `latex_name("\\sinh")`)
- static `ex` `GiNaC::cosh_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::cosh_eval` (const `ex` &`x`)
- static `ex` `GiNaC::cosh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::cosh_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::cosh_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::cosh_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`cosh`, `eval_func(cosh_eval)`. `evalf_func(cosh_evalf)`. `info_func(exp_info)`. `derivative_func(cosh_deriv)`. `real_part_func(cosh_real_part)`. `imag_part_func(cosh_imag_part)`. `conjugate_func(cosh_conjugate)`. `latex_name("\\cosh")`)
- static `ex` `GiNaC::tanh_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::tanh_eval` (const `ex` &`x`)
- static `ex` `GiNaC::tanh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::tanh_series` (const `ex` &`x`, const `relational` &`rel`, int `order`, unsigned `options`)
- static `ex` `GiNaC::tanh_real_part` (const `ex` &`x`)
- static `ex` `GiNaC::tanh_imag_part` (const `ex` &`x`)
- static `ex` `GiNaC::tanh_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`tanh`, `eval_func(tanh_eval)`. `evalf_func(tanh_evalf)`. `info_func(atan_info)`. `derivative_func(tanh_deriv)`. `series_func(tanh_series)`. `real_part_func(tanh_real_part)`. `imag_part_func(tanh_imag_part)`. `conjugate_func(tanh_conjugate)`. `latex_name("\\tanh")`)
- static `ex` `GiNaC::asinh_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::asinh_eval` (const `ex` &`x`)
- static `ex` `GiNaC::asinh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::asinh_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`asinh`, `eval_func(asinh_eval)`. `evalf_func(asinh_evalf)`. `info_func(atan_info)`. `derivative_func(asinh_deriv)`. `conjugate_func(asinh_conjugate)`)
- static `ex` `GiNaC::acosh_evalf` (const `ex` &`x`)
- static `ex` `GiNaC::acosh_eval` (const `ex` &`x`)
- static `ex` `GiNaC::acosh_deriv` (const `ex` &`x`, unsigned `deriv_param`)
- static `ex` `GiNaC::acosh_conjugate` (const `ex` &`x`)
- `GiNaC::REGISTER_FUNCTION` (`acosh`, `eval_func(acosh_eval)`. `evalf_func(acosh_evalf)`. `info_func(asin_info)`. `derivative_func(acosh_deriv)`. `conjugate_func(acosh_conjugate)`)
- static `ex` `GiNaC::atanh_evalf` (const `ex` &`x`)

- static [ex](#) [GiNaC::atanh\\_eval](#) (const [ex](#) &[x](#))
- static [ex](#) [GiNaC::atanh\\_deriv](#) (const [ex](#) &[x](#), unsigned [deriv\\_param](#))
- static [ex](#) [GiNaC::atanh\\_series](#) (const [ex](#) &[arg](#), const [relational](#) &[rel](#), int [order](#), unsigned [options](#))
- static [ex](#) [GiNaC::atanh\\_conjugate](#) (const [ex](#) &[x](#))
- [GiNaC::REGISTER\\_FUNCTION](#) ([atanh](#), [eval\\_func](#)([atanh\\_eval](#)). [evalf\\_func](#)([atanh\\_evalf](#)). [info\\_↔](#)  
[func](#)([asin\\_info](#)). [derivative\\_func](#)([atanh\\_deriv](#)). [series\\_func](#)([atanh\\_series](#)). [conjugate\\_func](#)([atanh\\_conjugate](#)))

### 7.49.1 Detailed Description

Implementation of transcendental (and trigonometric and hyperbolic) functions.

## 7.50 integral.cpp File Reference

Implementation of [GiNaC](#)'s symbolic integral.

```
#include "integral.h"
#include "numeric.h"
#include "symbol.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "inifcns.h"
#include "wildcard.h"
#include "archive.h"
#include "registrar.h"
#include "utils.h"
#include "operators.h"
#include "relational.h"
```

### Classes

- struct [GiNaC::error\\_and\\_integral](#)
- struct [GiNaC::error\\_and\\_integral\\_is\\_less](#)

### Namespaces

- namespace [GiNaC](#)

### Typedefs

- typedef map< [error\\_and\\_integral](#), [ex](#), [error\\_and\\_integral\\_is\\_less](#) > [GiNaC::lookup\\_map](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([integral](#), [basic](#), [print\\_func](#)< [print\\_dflt](#)  
>(&[integral::do\\_print](#)). [print\\_func](#)< [print\\_python](#) >(&[integral::do\\_print](#)). [print\\_func](#)< [print\\_latex](#)  
>(&[integral::do\\_print\\_latex](#))) [integral](#)
- [ex](#) [GiNaC::subvalue](#) (const [ex](#) &[var](#), const [ex](#) &[value](#), const [ex](#) &[fun](#))
- [ex](#) [GiNaC::adaptivesimpson](#) (const [ex](#) &[x](#), const [ex](#) &[a\\_in](#), const [ex](#) &[b\\_in](#), const [ex](#) &[f](#), const [ex](#) &[error](#))  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([integral](#))

### 7.50.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic integral.

## 7.51 integral.h File Reference

Interface to [GiNaC](#)'s symbolic integral.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::integral](#)  
*Symbolic integral.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([integral](#))
- [ex GiNaC::adaptivesimpson](#) (const [ex](#) &x, const [ex](#) &a\_in, const [ex](#) &b\_in, const [ex](#) &f, const [ex](#) &error)  
*Numeric integration routine based upon the "Adaptive Quadrature" one in "Numerical Analysis" by Burden and Faires.*

### 7.51.1 Detailed Description

Interface to [GiNaC](#)'s symbolic integral.

## 7.52 integration\_kernel.cpp File Reference

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "integration_kernel.h"
#include "add.h"
#include "mul.h"
#include "operators.h"
#include "power.h"
#include "relational.h"
#include "symbol.h"
#include "constant.h"
#include "numeric.h"
#include "function.h"
#include "pseries.h"
#include "utils.h"
#include "inifcns.h"
#include <iostream>
#include <stdexcept>
#include <cln/cln.h>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [ex GiNaC::ifactor](#) (const [numeric](#) &n)  
*Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p_1, \dots, pr), lst(a_1, \dots, ar) )$  such that  $n = p_1^{a_1} \dots p_r^{a_r}$ .*
- [bool GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const [numeric](#) &n)  
*Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.*
- [numeric GiNaC::kronecker\\_symbol](#) (const [numeric](#) &a, const [numeric](#) &n)  
*Returns the Kronecker symbol  $a: integer\ n: integer$ .*
- [numeric GiNaC::primitive\\_dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a)  
*Defines a primitive Dirichlet character through the Kronecker symbol.*
- [numeric GiNaC::dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a, const [numeric](#) &N)  
*Defines a Dirichlet character through the Kronecker symbol.*
- [numeric GiNaC::generalised\\_Bernoulli\\_number](#) (const [numeric](#) &k, const [numeric](#) &b)  
*The generalised Bernoulli number.*
- [ex GiNaC::Bernoulli\\_polynomial](#) (const [numeric](#) &k, const [ex](#) &x)  
*The Bernoulli polynomials.*
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (integration\_kernel, basic, print\_func< print\_context >(&integration\_kernel::do\_print)) integration\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (integration\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (basic\_log\_kernel, integration\_kernel, print\_func< print\_context >(&basic\_log\_kernel::do\_print)) basic\_log\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (basic\_log\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (multiple\_polylog\_kernel, integration\_kernel, print\_func< print\_context >(&multiple\_polylog\_kernel::do\_print)) multiple\_polylog\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (multiple\_polylog\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (ELi\_kernel, integration\_kernel, print\_func< print\_context >(&ELi\_kernel::do\_print)) ELi\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (ELi\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Ebar\_kernel, integration\_kernel, print\_func< print\_context >(&Ebar\_kernel::do\_print)) Ebar\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Ebar\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dtau\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dtau\_kernel::do\_print)) Kronecker\_dtau\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dtau\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Kronecker\_dz\_kernel, integration\_kernel, print\_func< print\_context >(&Kronecker\_dz\_kernel::do\_print)) Kronecker\_dz\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Kronecker\_dz\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_kernel::do\_print)) Eisenstein\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (Eisenstein\_h\_kernel, integration\_kernel, print\_func< print\_context >(&Eisenstein\_h\_kernel::do\_print)) Eisenstein\_h\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (Eisenstein\_h\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (modular\_form\_kernel, integration\_kernel, print\_func< print\_context >(&modular\_form\_kernel::do\_print)) modular\_form\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (modular\_form\_kernel)
- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (user\_defined\_kernel, integration\_kernel, print\_func< print\_context >(&user\_defined\_kernel::do\_print)) user\_defined\_kernel
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (user\_defined\_kernel)

### 7.52.1 Detailed Description

Implementation of [GiNaC](#)'s integration kernels for iterated integrals.

### 7.52.2 Variable Documentation

#### 7.52.2.1 qbar

```
ex qbar
```

Referenced by [GiNaC::divide\\_in\\_z\(\)](#), [GiNaC::ELi\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Ebar\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Kronecker\\_dtau\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::get\\_numerical\\_value\(\)](#), [GiNaC::modular\\_form\\_kernel::is\\_numeric\(\)](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), and [GiNaC::Kronecker\\_dz\\_kernel::series\\_coeff\\_impl\(\)](#).

#### 7.52.2.2 order

```
int order
```

Referenced by [GiNaC::atan\\_series\(\)](#), [GiNaC::atanh\\_series\(\)](#), [GiNaC::beta\\_series\(\)](#), [GiNaC::fderivative::do\\_print\\_latex\(\)](#), [GiNaC::EllipticE\\_series\(\)](#), [GiNaC::EllipticK\\_series\(\)](#), [GiNaC::modular\\_form\\_kernel::Laurent\\_series\(\)](#), [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::Laurent\\_series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::Laurent\\_series\(\)](#), [GiNaC::user\\_defined\\_kernel::Laurent\\_series\(\)](#), [GiNaC::lgamma\\_series\(\)](#), [GiNaC::Li2\\_series\(\)](#), [GiNaC::Li\\_series\(\)](#), [GiNaC::log\\_series\(\)](#), [GiNaC::Order\\_series\(\)](#), [GiNaC::psi1\\_series\(\)](#), [GiNaC::psi2\\_series\(\)](#), [GiNaC::Eisenstein\\_kernel::q\\_expansion\\_modular\\_form\(\)](#), [GiNaC::S\\_series\(\)](#), [GiNaC::ex::series\(\)](#), [GiNaC::series\(\)](#), [GiNaC::basic::series\(\)](#), [GiNaC::add::series\(\)](#), [GiNaC::fderivative::series\(\)](#), [GiNaC::function::series\(\)](#), [GiNaC::integral::series\(\)](#), [GiNaC::integration\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_kernel::series\(\)](#), [GiNaC::Eisenstein\\_h\\_kernel::series\(\)](#), [GiNaC::modular\\_form\\_kernel::series\(\)](#), [GiNaC::pseries::series\(\)](#), [GiNaC::structure< T, CompA, CompB>::series\(\)](#), [GiNaC::mul::series\(\)](#), [GiNaC::power::series\(\)](#), [GiNaC::symbol::series\(\)](#), [GiNaC::tan\\_series\(\)](#), [GiNaC::tanh\\_series\(\)](#), and [GiNaC::tgamma\\_series\(\)](#).

#### 7.52.2.3 cache\_vec

```
std::vector<ex> cache_vec [static], [protected]
```

#### 7.52.2.4 x

```
symbol x [static], [protected]
```

Referenced by [GiNaC::integration\\_kernel::Laurent\\_series\(\)](#), and [GiNaC::integration\\_kernel::series\\_coeff\(\)](#).

## 7.53 integration\_kernel.h File Reference

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

```
#include "basic.h"
#include "archive.h"
#include "numeric.h"
#include "lst.h"
#include <cln/complex.h>
#include <vector>
```

## Classes

- class [GiNaC::integration\\_kernel](#)  
The base class for integration kernels for iterated integrals.
- class [GiNaC::basic\\_log\\_kernel](#)  
The basic integration kernel with a logarithmic singularity at the origin.
- class [GiNaC::multiple\\_polylog\\_kernel](#)  
The integration kernel for multiple polylogarithms.
- class [GiNaC::ELi\\_kernel](#)  
The ELi-kernel.
- class [GiNaC::Ebar\\_kernel](#)  
The Ebar-kernel.
- class [GiNaC::Kronecker\\_dtau\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n)}(z_j, K\tau)$  in  $\tau$  (or equivalently in  $\bar{q}$ ).
- class [GiNaC::Kronecker\\_dz\\_kernel](#)  
The kernel corresponding to integrating the Kronecker coefficient function  $g^{(n-1)}(z - z_j, K\tau)$  in  $z$ .
- class [GiNaC::Eisenstein\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $E_{k,N,a,b,K}(\tau)$ .
- class [GiNaC::Eisenstein\\_h\\_kernel](#)  
The kernel corresponding to the Eisenstein series  $h_{k,N,r,s}(\tau)$ .
- class [GiNaC::modular\\_form\\_kernel](#)  
A kernel corresponding to a polynomial in Eisenstein series.
- class [GiNaC::user\\_defined\\_kernel](#)  
A user-defined integration kernel.

## Namespaces

- namespace [GiNaC](#)

## Functions

- [ex GiNaC::ifactor](#) (const [numeric](#) &n)  
Returns the decomposition of the positive integer  $n$  into prime numbers in the form  $lst( lst(p1,...,pr), lst(a1,...,ar) )$  such that  $n = p1^{a1} \dots pr^{ar}$ .
- [bool GiNaC::is\\_discriminant\\_of\\_quadratic\\_number\\_field](#) (const [numeric](#) &n)  
Returns true if the integer  $n$  is either one or the discriminant of a quadratic number field.
- [numeric GiNaC::kronecker\\_symbol](#) (const [numeric](#) &a, const [numeric](#) &n)  
Returns the Kronecker symbol  $a$ : integer  $n$ : integer.
- [numeric GiNaC::primitive\\_dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a)  
Defines a primitive Dirichlet character through the Kronecker symbol.
- [numeric GiNaC::dirichlet\\_character](#) (const [numeric](#) &n, const [numeric](#) &a, const [numeric](#) &N)  
Defines a Dirichlet character through the Kronecker symbol.
- [numeric GiNaC::generalised\\_Bernoulli\\_number](#) (const [numeric](#) &k, const [numeric](#) &b)  
The generalised Bernoulli number.
- [ex GiNaC::Bernoulli\\_polynomial](#) (const [numeric](#) &k, const [ex](#) &x)  
The Bernoulli polynomials.
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([integration\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([basic\\_log\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([multiple\\_polylog\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([ELi\\_kernel](#))

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([Ebar\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dtau\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([Kronecker\\_dz\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([Eisenstein\\_h\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([modular\\_form\\_kernel](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([user\\_defined\\_kernel](#))

### 7.53.1 Detailed Description

Interface to [GiNaC](#)'s integration kernels for iterated integrals.

## 7.54 Ist.cpp File Reference

Implementation of [GiNaC](#)'s Ist.

```
#include "lst.h"
#include "archive.h"
```

### Namespaces

- namespace [GiNaC](#)

### Variables

- `template<> GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T(Ist, basic, print_func< print\_context >(&Ist::do_print). print_func< print\_tree >(&Ist::do_print_tree))` `template<> bool` `Is GiNaC::GINAC\_BIND\_UNARCHIVER )(Ist)`

*Specialization of [container::info\(\)](#) for Ist.*

### 7.54.1 Detailed Description

Implementation of [GiNaC](#)'s Ist.

## 7.55 Ist.h File Reference

Definition of [GiNaC](#)'s Ist.

```
#include "container.h"
#include <list>
```

### Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef [container](#)< std::list > [GiNaC::lst](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([lst](#))

### 7.55.1 Detailed Description

Definition of [GiNaC](#)'s [lst](#).

## 7.56 matrix.cpp File Reference

Implementation of symbolic matrices.

```
#include "matrix.h"
#include "numeric.h"
#include "lst.h"
#include "idx.h"
#include "indexed.h"
#include "add.h"
#include "power.h"
#include "symbol.h"
#include "operators.h"
#include "normal.h"
#include "archive.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <map>
#include <sstream>
#include <stdexcept>
#include <string>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`matrix`, `basic`, `print_func` < `print_context` >(&`matrix::do_print`)). `print_func` < `print_latex` >(&`matrix::do_print_latex`). `print_func` < `print_tree` >(&`matrix::do_print_tree`). `print_func` < `print_python_repr` >(&`matrix::do_print_python_repr`)) `matrix`  
*Default ctor.*
- `GiNaC::GINAC_BIND_UNARCHIVER` (`matrix`)
- `ex GiNaC::lst_to_matrix` (const `lst` &`l`)  
*Convert list of lists to matrix.*
- `ex GiNaC::diag_matrix` (const `lst` &`l`)  
*Convert list of diagonal elements to matrix.*
- `ex GiNaC::diag_matrix` (std::initializer\_list< `ex` > `l`)
- `ex GiNaC::unit_matrix` (unsigned `r`, unsigned `c`)  
*Create an r times c unit matrix.*
- `ex GiNaC::symbolic_matrix` (unsigned `r`, unsigned `c`, const std::string &`base_name`, const std::string &`tex_↵` `base_name`)  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- `ex GiNaC::reduced_matrix` (const `matrix` &`m`, unsigned `r`, unsigned `c`)  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- `ex GiNaC::sub_matrix` (const `matrix` &`m`, unsigned `r`, unsigned `nr`, unsigned `c`, unsigned `nc`)  
*Return the nr times nc submatrix starting at position r, c of matrix m.*

### 7.56.1 Detailed Description

Implementation of symbolic matrices.

## 7.57 matrix.h File Reference

Interface to symbolic matrices.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include "compiler.h"
#include <string>
#include <vector>
```

## Classes

- class `GiNaC::matrix`  
*Symbolic matrices.*

## Namespaces

- namespace `GiNaC`

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([matrix](#))
- [size\\_t GiNaC::nops](#) (const [matrix](#) &[m](#))
- [ex GiNaC::expand](#) (const [matrix](#) &[m](#), unsigned [options](#)=0)
- [ex GiNaC::evalf](#) (const [matrix](#) &[m](#))
- unsigned [GiNaC::rows](#) (const [matrix](#) &[m](#))
- unsigned [GiNaC::cols](#) (const [matrix](#) &[m](#))
- [matrix GiNaC::transpose](#) (const [matrix](#) &[m](#))
- [ex GiNaC::determinant](#) (const [matrix](#) &[m](#), unsigned [options](#)=[determinant\\_algo::automatic](#))
- [ex GiNaC::trace](#) (const [matrix](#) &[m](#))
- [ex GiNaC::charpoly](#) (const [matrix](#) &[m](#), const [ex](#) &[lambda](#))
- [matrix GiNaC::inverse](#) (const [matrix](#) &[m](#))
- [matrix GiNaC::inverse](#) (const [matrix](#) &[m](#), unsigned [algo](#))
- unsigned [GiNaC::rank](#) (const [matrix](#) &[m](#))
- unsigned [GiNaC::rank](#) (const [matrix](#) &[m](#), unsigned [solve\\_algo](#))
- [ex GiNaC::lst\\_to\\_matrix](#) (const [lst](#) &[l](#))  
*Convert list of lists to matrix.*
- [ex GiNaC::diag\\_matrix](#) (const [lst](#) &[l](#))  
*Convert list of diagonal elements to matrix.*
- [ex GiNaC::diag\\_matrix](#) (std::initializer\_list< [ex](#) > [l](#))
- [ex GiNaC::unit\\_matrix](#) (unsigned [r](#), unsigned [c](#))  
*Create an r times c unit matrix.*
- [ex GiNaC::unit\\_matrix](#) (unsigned [x](#))  
*Create a x times x unit matrix.*
- [ex GiNaC::symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &[base\\_name](#), const std::string &[tex\\_↵](#)  
[base\\_name](#))  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*
- [ex GiNaC::reduced\\_matrix](#) (const [matrix](#) &[m](#), unsigned [r](#), unsigned [c](#))  
*Return the reduced matrix that is formed by deleting the rth row and cth column of matrix m.*
- [ex GiNaC::sub\\_matrix](#) (const [matrix](#) &[m](#), unsigned [r](#), unsigned [nr](#), unsigned [c](#), unsigned [nc](#))  
*Return the nr times nc submatrix starting at position r, c of matrix m.*
- [ex GiNaC::symbolic\\_matrix](#) (unsigned [r](#), unsigned [c](#), const std::string &[base\\_name](#))  
*Create an r times c matrix of newly generated symbols consisting of the given base name plus the numeric row/column position of each element.*

### 7.57.1 Detailed Description

Interface to symbolic matrices.

## 7.58 mul.cpp File Reference

Implementation of [GiNaC](#)'s products of expressions.

```
#include "mul.h"
#include "add.h"
#include "power.h"
#include "operators.h"
#include "matrix.h"
```

```
#include "indexed.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "symbol.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([mul](#), [expairseq](#), [print\\_func< print\\_context >\(&mul::do\\_print\)](#), [print\\_func< print\\_latex >\(&mul::do\\_print\\_latex\)](#), [print\\_func< print\\_csrc >\(&mul::do\\_print\\_csrc\)](#), [print\\_func< print\\_tree >\(&mul::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&mul::do\\_print\\_python\\_repr\)](#))  
[mul](#)
- [bool GiNaC::tryfactsubs](#) (const [ex](#) &origfactor, const [ex](#) &patternfactor, int &nummatches, [exmap](#) &repls)
- [bool GiNaC::algebraic\\_match\\_mul\\_with\\_mul](#) (const [mul](#) &e, const [ex](#) &pat, [exmap](#) &repls, int [factor](#), int &nummatches, const std::vector< bool > &subsed, std::vector< bool > &matched)  
*Checks whether e matches to the pattern pat and the (possibly to be updated) list of replacements repls.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([mul](#))

## 7.58.1 Detailed Description

Implementation of [GiNaC](#)'s products of expressions.

## 7.59 mul.h File Reference

Interface to [GiNaC](#)'s products of expressions.

```
#include "expairseq.h"
```

## Classes

- class [GiNaC::mul](#)  
*Product of expressions.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([mul](#))

### 7.59.1 Detailed Description

Interface to [GiNaC](#)'s products of expressions.

## 7.60 ncmul.cpp File Reference

Implementation of [GiNaC](#)'s non-commutative products of expressions.

```
#include "ncmul.h"
#include "ex.h"
#include "add.h"
#include "mul.h"
#include "clifford.h"
#include "matrix.h"
#include "archive.h"
#include "indexed.h"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef std::vector< std::size\_t > [GiNaC::uintvector](#)
- typedef std::vector< unsigned > [GiNaC::unsignedvector](#)
- typedef std::vector< [exvector](#) > [GiNaC::exvectorvector](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([ncmul](#), [exprseq](#), [print\\_func](#)< [print\\_context](#) >(&[ncmul::do\\_print](#)). [print\\_func](#)< [print\\_tree](#) >(&[ncmul::do\\_print\\_tree](#)). [print\\_func](#)< [print\\_csrc](#) >(&[ncmul::do\\_print\\_csrc](#)). [print\\_func](#)< [print\\_python\\_repr](#) >(&[ncmul::do\\_print\\_csrc](#))) [ncmul](#)
- [ex](#) [GiNaC::reeval\\_ncmul](#) (const [exvector](#) &[v](#))
- [ex](#) [GiNaC::hold\\_ncmul](#) (const [exvector](#) &[v](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([ncmul](#))

### 7.60.1 Detailed Description

Implementation of [GiNaC](#)'s non-commutative products of expressions.

## 7.61 ncmul.h File Reference

Interface to [GiNaC](#)'s non-commutative products of expressions.

```
#include "exprseq.h"
#include "archive.h"
```

### Classes

- class [GiNaC::ncmul](#)  
*Non-commutative product of expressions.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([ncmul](#))
- [ex GiNaC::reeval\\_ncmul](#) (const [exvector](#) &[v](#))
- [ex GiNaC::hold\\_ncmul](#) (const [exvector](#) &[v](#))

### 7.61.1 Detailed Description

Interface to [GiNaC](#)'s non-commutative products of expressions.

## 7.62 normal.cpp File Reference

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "normal.h"
#include "basic.h"
#include "ex.h"
#include "add.h"
#include "constant.h"
#include "expairseq.h"
#include "fail.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "numeric.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "matrix.h"
#include "pseries.h"
#include "symbol.h"
#include "utils.h"
#include "polynomial/chinrem_gcd.h"
#include <algorithm>
#include <map>
```

## Classes

- struct [GiNaC::sym\\_desc](#)  
*This structure holds information about the highest and lowest degrees in which a symbol appears in two multivariate polynomials "a" and "b".*
- class [GiNaC::gcdheu\\_failed](#)  
*Exception thrown by [heur\\_gcd\(\)](#) to signal failure.*
- struct [GiNaC::normal\\_map\\_function](#)  
*Function object to be applied by [basic::normal\(\)](#).*

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define [FAST\\_COMPARE](#) 1
- #define [USE\\_REMEMBER](#) 0
- #define [USE\\_TRIAL\\_DIVISION](#) 0
- #define [STATISTICS](#) 0

## Typedefs

- typedef std::vector< [sym\\_desc](#) > [GiNaC::sym\\_desc\\_vec](#)

## Functions

- static bool [GiNaC::get\\_first\\_symbol](#) (const [ex](#) &e, [ex](#) &x)  
*Return pointer to first symbol found in expression.*
- static void [GiNaC::add\\_symbol](#) (const [ex](#) &s, [sym\\_desc\\_vec](#) &v)
- static void [GiNaC::collect\\_symbols](#) (const [ex](#) &e, [sym\\_desc\\_vec](#) &v)
- static void [GiNaC::get\\_symbol\\_stats](#) (const [ex](#) &a, const [ex](#) &b, [sym\\_desc\\_vec](#) &v)  
*Collect statistical information about symbols in polynomials.*
- static [numeric](#) [GiNaC::lcmcoeff](#) (const [ex](#) &e, const [numeric](#) &l)
- static [numeric](#) [GiNaC::lcm\\_of\\_coefficients\\_denominators](#) (const [ex](#) &e)  
*Compute LCM of denominators of coefficients of a polynomial.*
- static [ex](#) [GiNaC::multiply\\_lcm](#) (const [ex](#) &e, const [numeric](#) &lcm)  
*Bring polynomial from  $Q[X]$  to  $Z[X]$  by multiplying in the previously determined LCM of the coefficient's denominators.*
- [ex](#) [GiNaC::quo](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex](#) [GiNaC::rem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex](#) [GiNaC::decomp\\_rational](#) (const [ex](#) &a, const [ex](#) &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- [ex](#) [GiNaC::prem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex](#) [GiNaC::sprem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- bool [GiNaC::divide](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) &q, bool check\_args)

- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
  - static bool `GiNaC::divide_in_z` (const `ex` &a, const `ex` &b, `ex` &q, `sym_desc_vec::const_iterator` var)
- Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Z[X]$ .*
  - static `ex` `GiNaC::sr_gcd` (const `ex` &a, const `ex` &b, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the subresultant PRS algorithm.*
  - static `ex` `GiNaC::interpolate` (const `ex` &gamma, const `numeric` &xi, const `ex` &x, int degree\_hint=1)
- $\xi$ -adic polynomial interpolation*
  - static bool `GiNaC::heur_gcd_z` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static bool `GiNaC::heur_gcd` (`ex` &res, const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, `sym_desc_vec::const_iterator` var)
- Compute GCD of multivariate polynomials using the heuristic GCD algorithm.*
  - static `ex` `GiNaC::gcd_pf_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
  - static `ex` `GiNaC::gcd_pf_mul` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
  - `ex` `GiNaC::gcd` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb, bool check\_args, unsigned `options`)
- Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
  - static `ex` `GiNaC::gcd_pf_pow_pow` (const `ex` &a, const `ex` &b, `ex` \*ca, `ex` \*cb)
  - `ex` `GiNaC::lcm` (const `ex` &a, const `ex` &b, bool check\_args)
- Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
  - static `epvector` `GiNaC::sqrfree_yun` (const `ex` &a, const `symbol` &x)
- Compute square-free factorization of multivariate polynomial  $a(x)$  using Yun's algorithm.*
  - `ex` `GiNaC::sqrfree` (const `ex` &a, const `lst` &l)
- Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
  - `ex` `GiNaC::sqrfree_parfrac` (const `ex` &a, const `symbol` &x)
- Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
  - static `ex` `GiNaC::replace_with_symbol` (const `ex` &e, `exmap` &repl, `exmap` &rev\_lookup, `lst` &modifier)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static `ex` `GiNaC::replace_with_symbol` (const `ex` &e, `exmap` &repl)
- Create a symbol for replacing the expression "e" (or return a previously assigned symbol).*
  - static `ex` `GiNaC::frac_cancel` (const `ex` &n, const `ex` &d)
- Fraction cancellation.*
  - static `ex` `GiNaC::find_common_factor` (const `ex` &e, `ex` &factor, `exmap` &repl)
- Remove the common factor in the terms of a sum 'e' by calculating the GCD, and multiply it into the expression 'factor' (which needs to be initialized to 1, unless you're accumulating factors).*
  - `ex` `GiNaC::collect_common_factors` (const `ex` &e)
- Collect common factors in sums.*
  - `ex` `GiNaC::resultant` (const `ex` &e1, const `ex` &e2, const `ex` &s)
- Resultant of two expressions  $e_1, e_2$  with respect to symbol  $s$ .*

## 7.62.1 Detailed Description

This file implements several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 7.62.2 Macro Definition Documentation

### 7.62.2.1 FAST\_COMPARE

```
#define FAST_COMPARE 1
```

## 7.62.2.2 USE\_REMEMBER

```
#define USE_REMEMBER 0
```

## 7.62.2.3 USE\_TRIAL\_DIVISION

```
#define USE_TRIAL_DIVISION 0
```

## 7.62.2.4 STATISTICS

```
#define STATISTICS 0
```

## 7.63 normal.h File Reference

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

```
#include "lst.h"
```

## Classes

- struct [GiNaC::gcd\\_options](#)  
*Flags to control the behavior of [gcd\(\)](#) and friends.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [ex GiNaC::quo](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Quotient  $q(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::rem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Remainder  $r(x)$  of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::decomp\\_rational](#) (const [ex](#) &a, const [ex](#) &x)  
*Decompose rational function  $a(x)=N(x)/D(x)$  into  $P(x)+n(x)/D(x)$  with  $\text{degree}(n, x) < \text{degree}(D, x)$ .*
- [ex GiNaC::prem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [ex GiNaC::sprem](#) (const [ex](#) &a, const [ex](#) &b, const [ex](#) &x, bool check\_args)  
*Sparse pseudo-remainder of polynomials  $a(x)$  and  $b(x)$  in  $Q[x]$ .*
- [bool GiNaC::divide](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) &q, bool check\_args)  
*Exact polynomial division of  $a(X)$  by  $b(X)$  in  $Q[X]$ .*
- [ex GiNaC::gcd](#) (const [ex](#) &a, const [ex](#) &b, [ex](#) \*ca, [ex](#) \*cb, bool check\_args, unsigned [options](#))  
*Compute GCD (Greatest Common Divisor) of multivariate polynomials  $a(X)$  and  $b(X)$  in  $Z[X]$ .*
- [ex GiNaC::lcm](#) (const [ex](#) &a, const [ex](#) &b, bool check\_args)  
*Compute LCM (Least Common Multiple) of multivariate polynomials in  $Z[X]$ .*
- [ex GiNaC::sqrfree](#) (const [ex](#) &a, const [lst](#) &l)  
*Compute a square-free factorization of a multivariate polynomial in  $Q[X]$ .*
- [ex GiNaC::sqrfree\\_parfrac](#) (const [ex](#) &a, const [symbol](#) &x)  
*Compute square-free partial fraction decomposition of rational function  $a(x)$ .*
- [ex GiNaC::collect\\_common\\_factors](#) (const [ex](#) &e)  
*Collect common factors in sums.*
- [ex GiNaC::resultant](#) (const [ex](#) &e1, const [ex](#) &e2, const [ex](#) &s)  
*Resultant of two expressions  $e1, e2$  with respect to symbol  $s$ .*

### 7.63.1 Detailed Description

This file defines several functions that work on univariate and multivariate polynomials and rational functions.

These functions include polynomial quotient and remainder, GCD and LCM computation, square-free factorization and rational function normalization.

## 7.64 numeric.cpp File Reference

This file contains the interface to the underlying bignum package.

```
#include "numeric.h"
#include "ex.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <cln/output.h>
#include <cln/integer_io.h>
#include <cln/integer_ring.h>
#include <cln/rational_io.h>
#include <cln/rational_ring.h>
#include <cln/lfloat_class.h>
#include <cln/lfloat_io.h>
#include <cln/real_io.h>
#include <cln/real_ring.h>
#include <cln/complex_io.h>
#include <cln/complex_ring.h>
#include <cln/numtheory.h>
```

### Classes

- class [GiNaC::lanczos\\_coeffs](#)

### Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`numeric`, `basic`, `print_func< print_context >(&numeric::do_print)`, `print_func< print_latex >(&numeric::do_print_latex)`, `print_func< print_csrc >(&numeric::do_print_csrc)`, `print_func< print_csrc_cl_N >(&numeric::do_print_csrc_cl_N)`, `print_func< print_tree >(&numeric::do_print_tree)`, `print_func< print_python_repr >(&numeric::do_print_python_repr)`)  
`numeric`  
*default ctor.*
- static const `cln::cl_F` `GiNaC::make_real_float` (`const cln::cl_idcoded_float &dec`)  
*Construct a floating point number from sign, mantissa, and exponent.*
- static const `cln::cl_F` `GiNaC::read_real_float` (`std::istream &s`)  
*Read serialized floating point number.*
- `GiNaC::GINAC_BIND_UNARCHIVER` (`numeric`)
- static void `GiNaC::write_real_float` (`std::ostream &s`, `const cln::cl_R &n`)
- static void `GiNaC::print_real_number` (`const print_context &c`, `const cln::cl_R &x`)  
*Helper function to print a real number in a nicer way than is CLN's default.*
- static void `GiNaC::print_integer_csrc` (`const print_context &c`, `const cln::cl_I &x`)  
*Helper function to print integer number in C++ source format.*
- static void `GiNaC::print_real_csrc` (`const print_context &c`, `const cln::cl_R &x`)  
*Helper function to print real number in C++ source format.*
- `template<typename T1 , typename T2 >`  
static bool `GiNaC::coerce` (`T1 &dst`, `const T2 &arg`)
- `template<> bool` `GiNaC::coerce< int, cln::cl_I >` (`int &dst`, `const cln::cl_I &arg`)  
*Check if CLN integer can be converted into int.*
- `template<> bool` `GiNaC::coerce< unsigned int, cln::cl_I >` (`unsigned int &dst`, `const cln::cl_I &arg`)
- static void `GiNaC::print_real_cl_N` (`const print_context &c`, `const cln::cl_R &x`)  
*Helper function to print real number in C++ source format using cl\_N types.*
- const `numeric` `GiNaC::exp` (`const numeric &x`)  
*Exponential function.*
- const `numeric` `GiNaC::log` (`const numeric &x`)  
*Natural logarithm.*
- const `numeric` `GiNaC::sin` (`const numeric &x`)  
*Numeric sine (trigonometric function).*
- const `numeric` `GiNaC::cos` (`const numeric &x`)  
*Numeric cosine (trigonometric function).*
- const `numeric` `GiNaC::tan` (`const numeric &x`)  
*Numeric tangent (trigonometric function).*
- const `numeric` `GiNaC::asin` (`const numeric &x`)  
*Numeric inverse sine (trigonometric function).*
- const `numeric` `GiNaC::acos` (`const numeric &x`)  
*Numeric inverse cosine (trigonometric function).*
- const `numeric` `GiNaC::atan` (`const numeric &x`)  
*Numeric arcustangent.*
- const `numeric` `GiNaC::atan` (`const numeric &y`, `const numeric &x`)  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const `numeric` `GiNaC::sinh` (`const numeric &x`)  
*Numeric hyperbolic sine (trigonometric function).*
- const `numeric` `GiNaC::cosh` (`const numeric &x`)  
*Numeric hyperbolic cosine (trigonometric function).*
- const `numeric` `GiNaC::tanh` (`const numeric &x`)  
*Numeric hyperbolic tangent (trigonometric function).*
- const `numeric` `GiNaC::asinh` (`const numeric &x`)

- Numeric inverse hyperbolic sine (trigonometric function).*
- const [numeric GiNaC::acosh](#) (const [numeric &x](#))
- Numeric inverse hyperbolic cosine (trigonometric function).*
- const [numeric GiNaC::atanh](#) (const [numeric &x](#))
- Numeric inverse hyperbolic tangent (trigonometric function).*
- static [cln::cl\\_N GiNaC::Li2\\_series](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))
- Numeric evaluation of Dilogarithm within circle of convergence (unit circle) using a power series.*
- static [cln::cl\\_N GiNaC::Li2\\_projection](#) (const [cln::cl\\_N &x](#), const [cln::float\\_format\\_t &prec](#))
- Folds Li2's argument inside a small rectangle to enhance convergence.*
- const [cln::cl\\_N GiNaC::Li2\\_](#) (const [cln::cl\\_N &value](#))
- Numeric evaluation of Dilogarithm.*
- const [numeric GiNaC::Li2](#) (const [numeric &x](#))
- const [numeric GiNaC::zeta](#) (const [numeric &x](#))
- Numeric evaluation of Riemann's Zeta function.*
- static [cln::float\\_format\\_t GiNaC::guess\\_precision](#) (const [cln::cl\\_N &x](#))
- const [cln::cl\\_N GiNaC::lgamma](#) (const [cln::cl\\_N &x](#))
- The Gamma function.*
- const [numeric GiNaC::lgamma](#) (const [numeric &x](#))
- const [cln::cl\\_N GiNaC::tgamma](#) (const [cln::cl\\_N &x](#))
- const [numeric GiNaC::tgamma](#) (const [numeric &x](#))
- const [numeric GiNaC::psi](#) (const [numeric &x](#))
- The psi function (aka polygamma function).*
- const [numeric GiNaC::psi](#) (const [numeric &n](#), const [numeric &x](#))
- The psi functions (aka polygamma functions).*
- const [numeric GiNaC::factorial](#) (const [numeric &n](#))
- Factorial combinatorial function.*
- const [numeric GiNaC::doublefactorial](#) (const [numeric &n](#))
- The double factorial combinatorial function.*
- const [numeric GiNaC::binomial](#) (const [numeric &n](#), const [numeric &k](#))
- The Binomial coefficients.*
- const [numeric GiNaC::bernoulli](#) (const [numeric &nn](#))
- Bernoulli number.*
- const [numeric GiNaC::fibonacci](#) (const [numeric &n](#))
- Fibonacci number.*
- const [numeric GiNaC::abs](#) (const [numeric &x](#))
- Absolute value.*
- const [numeric GiNaC::mod](#) (const [numeric &a](#), const [numeric &b](#))
- Modulus (in positive representation).*
- const [numeric GiNaC::smod](#) (const [numeric &a\\_](#), const [numeric &b\\_](#))
- Modulus (in symmetric representation).*
- const [numeric GiNaC::irem](#) (const [numeric &a](#), const [numeric &b](#))
- Numeric integer remainder.*
- const [numeric GiNaC::irem](#) (const [numeric &a](#), const [numeric &b](#), [numeric &q](#))
- Numeric integer remainder.*
- const [numeric GiNaC::iquo](#) (const [numeric &a](#), const [numeric &b](#))
- Numeric integer quotient.*
- const [numeric GiNaC::iquo](#) (const [numeric &a](#), const [numeric &b](#), [numeric &r](#))
- Numeric integer quotient.*
- const [numeric GiNaC::gcd](#) (const [numeric &a](#), const [numeric &b](#))
- Greatest Common Divisor.*
- const [numeric GiNaC::lcm](#) (const [numeric &a](#), const [numeric &b](#))

*Least Common Multiple.*

- const [numeric GiNaC::sqrt](#) (const [numeric](#) &x)

*Numeric square root.*

- const [numeric GiNaC::isqrt](#) (const [numeric](#) &x)

*Integer numeric square root.*

- [ex GiNaC::PiEvalf](#) ()

*Floating point evaluation of Archimedes' constant Pi.*

- [ex GiNaC::EulerEvalf](#) ()

*Floating point evaluation of Euler's constant gamma.*

- [ex GiNaC::CatalanEvalf](#) ()

*Floating point evaluation of Catalan's constant.*

- `std::ostream & GiNaC::operator<< (std::ostream &os, const \_numeric\_digits &e)`

## Variables

- const [numeric GiNaC::I](#) = [numeric](#)([cln::complex](#)([cln::cl\\_I](#)(0),[cln::cl\\_I](#)(1)))

*Imaginary unit.*

- [\\_numeric\\_digits GiNaC::Digits](#)

*Accuracy in decimal digits.*

## 7.64.1 Detailed Description

This file contains the interface to the underlying bignum package.

Its most important design principle is to completely hide the inner working of that other package from the user of [GiNaC](#). It must either provide implementation of arithmetic operators and numerical evaluation of special functions or implement the interface to the bignum package.

## 7.65 numeric.h File Reference

Makes the interface to the underlying bignum package available.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
#include <cln/complex.h>
#include <stdexcept>
#include <vector>
```

## Classes

- class [GiNaC::\\_numeric\\_digits](#)

*This class is used to instantiate a global singleton object Digits which behaves just like Maple's Digits.*

- class [GiNaC::pole\\_error](#)

*Exception class thrown when a singularity is encountered.*

- class [GiNaC::numeric](#)

*This class is a wrapper around CLN-numbers within the [GiNaC](#) class hierarchy.*

## Namespaces

- namespace [GiNaC](#)

## Typedefs

- typedef void(\* [GiNaC::digits\\_changed\\_callback](#)) (long)  
*Function pointer to implement callbacks in the case 'Digits' gets changed.*

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) (numeric)
- const [numeric GiNaC::exp](#) (const [numeric &x](#))  
*Exponential function.*
- const [numeric GiNaC::log](#) (const [numeric &x](#))  
*Natural logarithm.*
- const [numeric GiNaC::sin](#) (const [numeric &x](#))  
*Numeric sine (trigonometric function).*
- const [numeric GiNaC::cos](#) (const [numeric &x](#))  
*Numeric cosine (trigonometric function).*
- const [numeric GiNaC::tan](#) (const [numeric &x](#))  
*Numeric tangent (trigonometric function).*
- const [numeric GiNaC::asin](#) (const [numeric &x](#))  
*Numeric inverse sine (trigonometric function).*
- const [numeric GiNaC::acos](#) (const [numeric &x](#))  
*Numeric inverse cosine (trigonometric function).*
- const [numeric GiNaC::atan](#) (const [numeric &x](#))  
*Numeric arcustangent.*
- const [numeric GiNaC::atan](#) (const [numeric &y](#), const [numeric &x](#))  
*Numeric arcustangent of two arguments, analytically continued in a suitable way.*
- const [numeric GiNaC::sinh](#) (const [numeric &x](#))  
*Numeric hyperbolic sine (trigonometric function).*
- const [numeric GiNaC::cosh](#) (const [numeric &x](#))  
*Numeric hyperbolic cosine (trigonometric function).*
- const [numeric GiNaC::tanh](#) (const [numeric &x](#))  
*Numeric hyperbolic tangent (trigonometric function).*
- const [numeric GiNaC::asinh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic sine (trigonometric function).*
- const [numeric GiNaC::acosh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic cosine (trigonometric function).*
- const [numeric GiNaC::atanh](#) (const [numeric &x](#))  
*Numeric inverse hyperbolic tangent (trigonometric function).*
- const [numeric GiNaC::Li2](#) (const [numeric &x](#))
- const [numeric GiNaC::zeta](#) (const [numeric &x](#))  
*Numeric evaluation of Riemann's Zeta function.*
- const [numeric GiNaC::lgamma](#) (const [numeric &x](#))
- const [numeric GiNaC::tgamma](#) (const [numeric &x](#))
- const [numeric GiNaC::psi](#) (const [numeric &x](#))  
*The psi function (aka polygamma function).*
- const [numeric GiNaC::psi](#) (const [numeric &n](#), const [numeric &x](#))

*The psi functions (aka polygamma functions).*

- const [numeric GiNaC::factorial](#) (const [numeric](#) &n)

*Factorial combinatorial function.*

- const [numeric GiNaC::doublefactorial](#) (const [numeric](#) &n)

*The double factorial combinatorial function.*

- const [numeric GiNaC::binomial](#) (const [numeric](#) &n, const [numeric](#) &k)

*The Binomial coefficients.*

- const [numeric GiNaC::bernoulli](#) (const [numeric](#) &nn)

*Bernoulli number.*

- const [numeric GiNaC::fibonacci](#) (const [numeric](#) &n)

*Fibonacci number.*

- const [numeric GiNaC::isqrt](#) (const [numeric](#) &x)

*Integer numeric square root.*

- const [numeric GiNaC::sqrt](#) (const [numeric](#) &x)

*Numeric square root.*

- const [numeric GiNaC::abs](#) (const [numeric](#) &x)

*Absolute value.*

- const [numeric GiNaC::mod](#) (const [numeric](#) &a, const [numeric](#) &b)

*Modulus (in positive representation).*

- const [numeric GiNaC::smod](#) (const [numeric](#) &a\_, const [numeric](#) &b\_)

*Modulus (in symmetric representation).*

- const [numeric GiNaC::irem](#) (const [numeric](#) &a, const [numeric](#) &b)

*Numeric integer remainder.*

- const [numeric GiNaC::irem](#) (const [numeric](#) &a, const [numeric](#) &b, [numeric](#) &q)

*Numeric integer remainder.*

- const [numeric GiNaC::iquo](#) (const [numeric](#) &a, const [numeric](#) &b)

*Numeric integer quotient.*

- const [numeric GiNaC::iquo](#) (const [numeric](#) &a, const [numeric](#) &b, [numeric](#) &r)

*Numeric integer quotient.*

- const [numeric GiNaC::gcd](#) (const [numeric](#) &a, const [numeric](#) &b)

*Greatest Common Divisor.*

- const [numeric GiNaC::lcm](#) (const [numeric](#) &a, const [numeric](#) &b)

*Least Common Multiple.*

- const [numeric GiNaC::pow](#) (const [numeric](#) &x, const [numeric](#) &y)

- const [numeric GiNaC::inverse](#) (const [numeric](#) &x)

- [numeric GiNaC::step](#) (const [numeric](#) &x)

- int [GiNaC::csgn](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_zero](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_positive](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_negative](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_integer](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_pos\\_integer](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_nonneg\\_integer](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_even](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_odd](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_prime](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_rational](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_real](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_cinteger](#) (const [numeric](#) &x)

- bool [GiNaC::is\\_crational](#) (const [numeric](#) &x)

- int [GiNaC::to\\_int](#) (const [numeric](#) &x)

- long [GiNaC::to\\_long](#) (const [numeric](#) &x)

- double `GiNaC::to_double` (const numeric &x)
- const numeric `GiNaC::real` (const numeric &x)
- const numeric `GiNaC::imag` (const numeric &x)
- const numeric `GiNaC::numer` (const numeric &x)
- const numeric `GiNaC::denom` (const numeric &x)
- ex `GiNaC::PiEvalf` ()  
*Floating point evaluation of Archimedes' constant Pi.*
- ex `GiNaC::EulerEvalf` ()  
*Floating point evaluation of Euler's constant gamma.*
- ex `GiNaC::CatalanEvalf` ()  
*Floating point evaluation of Catalan's constant.*

### 7.65.1 Detailed Description

Makes the interface to the underlying bignum package available.

## 7.66 operators.cpp File Reference

Implementation of `GiNaC`'s overloaded operators.

```
#include "operators.h"
#include "numeric.h"
#include "add.h"
#include "mul.h"
#include "power.h"
#include "ncmul.h"
#include "relational.h"
#include "print.h"
#include "utils.h"
#include <iostream>
```

### Namespaces

- namespace `GiNaC`

### Enumerations

- enum { `GiNaC::callback_registered` = 1 }

## Functions

- static const [ex](#) [GiNaC::exadd](#) (const [ex](#) &lh, const [ex](#) &rh)  
*Used internally by [operator+\(\)](#) to add two [ex](#) objects.*
- static const [ex](#) [GiNaC::exmul](#) (const [ex](#) &lh, const [ex](#) &rh)  
*Used internally by [operator\\*\(\)](#) to multiply two [ex](#) objects.*
- static const [ex](#) [GiNaC::exminus](#) (const [ex](#) &lh)  
*Used internally by [operator-\(\)](#) and friends to change the sign of an argument.*
- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator\\*](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator/](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator\\*](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator/](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- [ex](#) & [GiNaC::operator+=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator-=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator\\*=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator/=](#) ([ex](#) &lh, const [ex](#) &rh)
- [numeric](#) & [GiNaC::operator+=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator-=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator\\*=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- [numeric](#) & [GiNaC::operator/=](#) ([numeric](#) &lh, const [numeric](#) &rh)
- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh)
- [ex](#) & [GiNaC::operator++](#) ([ex](#) &rh)  
*Expression prefix increment.*
- [ex](#) & [GiNaC::operator--](#) ([ex](#) &rh)  
*Expression prefix decrement.*
- const [ex](#) [GiNaC::operator++](#) ([ex](#) &lh, int)  
*Expression postfix increment.*
- const [ex](#) [GiNaC::operator--](#) ([ex](#) &lh, int)  
*Expression postfix decrement.*
- [numeric](#) & [GiNaC::operator++](#) ([numeric](#) &rh)  
*Numeric prefix increment.*
- [numeric](#) & [GiNaC::operator--](#) ([numeric](#) &rh)  
*Numeric prefix decrement.*
- const [numeric](#) [GiNaC::operator++](#) ([numeric](#) &lh, int)  
*Numeric postfix increment.*
- const [numeric](#) [GiNaC::operator--](#) ([numeric](#) &lh, int)  
*Numeric postfix decrement.*
- const [relational](#) [GiNaC::operator==](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator!=](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator<](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator<=](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator>](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [relational](#) [GiNaC::operator>=](#) (const [ex](#) &lh, const [ex](#) &rh)
- static int [GiNaC::my\\_ios\\_index](#) ()
- static void [GiNaC::my\\_ios\\_callback](#) (std::ios\_base::event ev, std::ios\_base &s, int i)
- static [print\\_context](#) \* [GiNaC::get\\_print\\_context](#) (std::ios\_base &s)

- static void [GiNaC::set\\_print\\_context](#) (std::ios\_base &s, const [print\\_context](#) &c)
- static unsigned [GiNaC::get\\_print\\_options](#) (std::ios\_base &s)
- static void [GiNaC::set\\_print\\_options](#) (std::ostream &s, unsigned [options](#))
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [ex](#) &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [exvector](#) &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [exset](#) &e)
- std::ostream & [GiNaC::operator<<](#) (std::ostream &os, const [exmap](#) &e)
- std::istream & [GiNaC::operator>>](#) (std::istream &is, [ex](#) &e)
- std::ostream & [GiNaC::dflt](#) (std::ostream &os)
- std::ostream & [GiNaC::latex](#) (std::ostream &os)
- std::ostream & [GiNaC::python](#) (std::ostream &os)
- std::ostream & [GiNaC::python\\_repr](#) (std::ostream &os)
- std::ostream & [GiNaC::tree](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_float](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_double](#) (std::ostream &os)
- std::ostream & [GiNaC::csrc\\_cl\\_N](#) (std::ostream &os)
- std::ostream & [GiNaC::index\\_dimensions](#) (std::ostream &os)
- std::ostream & [GiNaC::no\\_index\\_dimensions](#) (std::ostream &os)

### 7.66.1 Detailed Description

Implementation of [GiNaC](#)'s overloaded operators.

## 7.67 operators.h File Reference

Interface to [GiNaC](#)'s overloaded operators.

```
#include <iosfwd>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- const [ex](#) [GiNaC::operator+](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator-](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator\\*](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [ex](#) [GiNaC::operator/](#) (const [ex](#) &lh, const [ex](#) &rh)
- const [numeric](#) [GiNaC::operator+](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator-](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator\\*](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- const [numeric](#) [GiNaC::operator/](#) (const [numeric](#) &lh, const [numeric](#) &rh)
- [ex](#) & [GiNaC::operator+=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator-=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator\\*=](#) ([ex](#) &lh, const [ex](#) &rh)
- [ex](#) & [GiNaC::operator/=](#) ([ex](#) &lh, const [ex](#) &rh)

- `numeric & GiNaC::operator+= (numeric &lh, const numeric &rh)`
- `numeric & GiNaC::operator-= (numeric &lh, const numeric &rh)`
- `numeric & GiNaC::operator*= (numeric &lh, const numeric &rh)`
- `numeric & GiNaC::operator/= (numeric &lh, const numeric &rh)`
- `const ex GiNaC::operator+ (const ex &lh)`
- `const ex GiNaC::operator- (const ex &lh)`
- `const numeric GiNaC::operator+ (const numeric &lh)`
- `const numeric GiNaC::operator- (const numeric &lh)`
- `ex & GiNaC::operator++ (ex &rh)`  
*Expression prefix increment.*
- `ex & GiNaC::operator-- (ex &rh)`  
*Expression prefix decrement.*
- `const ex GiNaC::operator++ (ex &lh, int)`  
*Expression postfix increment.*
- `const ex GiNaC::operator-- (ex &lh, int)`  
*Expression postfix decrement.*
- `numeric & GiNaC::operator++ (numeric &rh)`  
*Numeric prefix increment.*
- `numeric & GiNaC::operator-- (numeric &rh)`  
*Numeric prefix decrement.*
- `const numeric GiNaC::operator++ (numeric &lh, int)`  
*Numeric postfix increment.*
- `const numeric GiNaC::operator-- (numeric &lh, int)`  
*Numeric postfix decrement.*
- `const relational GiNaC::operator== (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator!= (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator< (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator<= (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator> (const ex &lh, const ex &rh)`
- `const relational GiNaC::operator>= (const ex &lh, const ex &rh)`
- `std::ostream & GiNaC::operator<< (std::ostream &os, const ex &e)`
- `std::istream & GiNaC::operator>> (std::istream &is, ex &e)`
- `std::ostream & GiNaC::dflt (std::ostream &os)`
- `std::ostream & GiNaC::latex (std::ostream &os)`
- `std::ostream & GiNaC::python (std::ostream &os)`
- `std::ostream & GiNaC::python_repr (std::ostream &os)`
- `std::ostream & GiNaC::tree (std::ostream &os)`
- `std::ostream & GiNaC::csrc (std::ostream &os)`
- `std::ostream & GiNaC::csrc_float (std::ostream &os)`
- `std::ostream & GiNaC::csrc_double (std::ostream &os)`
- `std::ostream & GiNaC::csrc_cl_N (std::ostream &os)`
- `std::ostream & GiNaC::index_dimensions (std::ostream &os)`
- `std::ostream & GiNaC::no_index_dimensions (std::ostream &os)`

### 7.67.1 Detailed Description

Interface to `GiNaC`'s overloaded operators.

## 7.68 power.cpp File Reference

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "power.h"
#include "expairseq.h"
#include "add.h"
#include "mul.h"
#include "ncmul.h"
#include "numeric.h"
#include "constant.h"
#include "operators.h"
#include "inifcns.h"
#include "matrix.h"
#include "indexed.h"
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "relational.h"
#include "compiler.h"
#include <iostream>
#include <limits>
#include <stdexcept>
#include <vector>
#include <algorithm>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([power](#), [basic](#), [print\\_func< print\\_dflt >](#) (&[power::do\\_print\\_dflt](#)), [print\\_func< print\\_latex >](#) (&[power::do\\_print\\_latex](#)), [print\\_func< print\\_csrc >](#) (&[power::do\\_print\\_csrc](#)), [print\\_func< print\\_python >](#) (&[power::do\\_print\\_python](#)), [print\\_func< print\\_python\\_repr >](#) (&[power::do\\_print\\_python\\_repr](#)), [print\\_func< print\\_csrc\\_cl\\_N >](#) (&[power::do\\_print\\_csrc\\_cl\\_N](#))) [power](#)
- static void [GiNaC::print\\_sym\\_pow](#) (const [print\\_context](#) &[c](#), const [symbol](#) &[x](#), int [exp](#))
- bool [GiNaC::tryfactsubs](#) (const [ex](#) &[origfactor](#), const [ex](#) &[patternfactor](#), int &[nummatches](#), [exmap](#) &[repls](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([power](#))

### 7.68.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.69 power.h File Reference

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::power](#)

*This class holds a two-component object, a basis and an exponent representing exponentiation.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([power](#))
- [ex GiNaC::pow](#) (const [ex](#) &b, const [ex](#) &e)  
*Symbolic exponentiation.*
- `template<typename T1 , typename T2 >`  
[ex GiNaC::pow](#) (const T1 &b, const T2 &e)
- [ex GiNaC::sqrt](#) (const [ex](#) &a)  
*Square root expression.*

### 7.69.1 Detailed Description

Interface to [GiNaC](#)'s symbolic exponentiation ( $\text{basis}^{\text{exponent}}$ ).

## 7.70 print.cpp File Reference

Implementation of helper classes for expression output.

```
#include "print.h"
#include <iostream>
```

## Namespaces

- namespace [GiNaC](#)

## Variables

- unsigned [GiNaC::next\\_print\\_context\\_id](#) = 0  
*Next free ID for [print\\_context](#) types.*

### 7.70.1 Detailed Description

Implementation of helper classes for expression output.

## 7.71 print.h File Reference

Definition of helper classes for expression output.

```
#include "class_info.h"
#include <iosfwd>
#include <memory>
#include <string>
```

### Classes

- class [GiNaC::print\\_context\\_options](#)  
*This class stores information about a registered [print\\_context](#) class.*
- class [GiNaC::print\\_options](#)  
*Flags to control the behavior of a [print\\_context](#).*
- class [GiNaC::print\\_context](#)  
*Base class for [print\\_contexts](#).*
- class [GiNaC::print\\_dflt](#)  
*Context for default (ginsh-parsable) output.*
- class [GiNaC::print\\_latex](#)  
*Context for latex-parsable output.*
- class [GiNaC::print\\_python](#)  
*Context for python pretty-print output.*
- class [GiNaC::print\\_python\\_repr](#)  
*Context for python-parsable output.*
- class [GiNaC::print\\_tree](#)  
*Context for tree-like output for debugging.*
- class [GiNaC::print\\_csrc](#)  
*Base context for C source output.*
- class [GiNaC::print\\_csrc\\_float](#)  
*Context for C source output using float precision.*
- class [GiNaC::print\\_csrc\\_double](#)  
*Context for C source output using double precision.*
- class [GiNaC::print\\_csrc\\_cl\\_N](#)  
*Context for C source output using CLN numbers.*
- class [GiNaC::print\\_functor\\_impl](#)  
*Base class for [print\\_functor](#) handlers.*
- class [GiNaC::print\\_ptrfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for pointer-to-functions of class *T*, context type *C**
- class [GiNaC::print\\_memfun\\_handler< T, C >](#)  
*[print\\_functor](#) handler for member functions of class *T*, context type *C**
- class [GiNaC::print\\_functor](#)  
*This class represents a print method for a certain algebraic class and [print\\_context](#) type.*

### Namespaces

- namespace [GiNaC](#)

## Macros

- `#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname)`  
Common part of `GINAC_DECLARE_PRINT_CONTEXT_BASE` and `GINAC_DECLARE_PRINT_CONTEXT_DERIVED`.
- `#define GINAC_DECLARE_PRINT_CONTEXT_BASE(classname)`
- `#define GINAC_DECLARE_PRINT_CONTEXT(classname, surname)`  
Macro for inclusion in the declaration of a `print_context` class.
- `#define GINAC_IMPLEMENT_PRINT_CONTEXT(classname, surname)`  
Macro for inclusion in the implementation of each `print_context` class.

## Typedefs

- `typedef class_info< print_context_options > GiNaC::print_context_class_info`

## Functions

- `template<class T >`  
`bool GiNaC::is_a (const print_context &obj)`  
Check if `obj` is a `T`, including base classes.

## 7.71.1 Detailed Description

Definition of helper classes for expression output.

## 7.71.2 Macro Definition Documentation

### 7.71.2.1 GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON

```
#define GINAC_DECLARE_PRINT_CONTEXT_COMMON(
 classname)
```

#### Value:

```
public: \
 friend class function_options; \
 friend class registered_class_options; \
 static const GiNaC::print_context_class_info &get_class_info_static(); \
 classname();
```

Common part of `GINAC_DECLARE_PRINT_CONTEXT_BASE` and `GINAC_DECLARE_PRINT_CONTEXT_DERIVED`.

### 7.71.2.2 GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE

```
#define GINAC_DECLARE_PRINT_CONTEXT_BASE(
 classname)
```

#### Value:

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \
virtual const GiNaC::print_context_class_info &get_class_info() const { return
 classname::get_class_info_static(); } \
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); } \
virtual classname * duplicate() const { return new classname(*this); } \
private:
```

### 7.71.2.3 GINAC\_DECLARE\_PRINT\_CONTEXT

```
#define GINAC_DECLARE_PRINT_CONTEXT(
 classname,
 supertype)
```

#### Value:

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \
typedef supertype inherited; \
const GiNaC::print_context_class_info &get_class_info() const override { return
 classname::get_class_info_static(); } \
const char *class_name() const override { return classname::get_class_info_static().options.get_name();
 } \
classname * duplicate() const override { return new classname(*this); } \
private:
```

Macro for inclusion in the declaration of a `print_context` class.

It declares some functions that are common to all classes derived from 'print\_context' as well as all required stuff for the [GiNaC](#) registry.

### 7.71.2.4 GINAC\_IMPLEMENT\_PRINT\_CONTEXT

```
#define GINAC_IMPLEMENT_PRINT_CONTEXT(
 classname,
 supertype)
```

#### Value:

```
const GiNaC::print_context_class_info &classname::get_class_info_static() \
{ \
 static GiNaC::print_context_class_info reg_info =
 GiNaC::print_context_class_info(GiNaC::print_context_options(#classname, #supertype,
 GiNaC::next_print_context_id++)); \
 return reg_info; \
}
```

Macro for inclusion in the implementation of each `print_context` class.

## 7.72 pseries.cpp File Reference

Implementation of class for extended truncated power series and methods for series expansion.

```
#include "pseries.h"
#include "add.h"
#include "inifcns.h"
#include "lst.h"
#include "mul.h"
#include "power.h"
#include "relational.h"
#include "operators.h"
#include "symbol.h"
#include "integral.h"
#include "archive.h"
#include "utils.h"
#include <limits>
#include <numeric>
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([pseries](#), [basic](#), [print\\_func](#)< [print\\_context](#)>(&[pseries::do\\_print](#)). [print\\_func](#)< [print\\_latex](#)>(&[pseries::do\\_print\\_latex](#)). [print\\_func](#)< [print\\_tree](#)>(&[pseries::do\\_print\\_tree](#)). [print\\_func](#)< [print\\_python](#)>(&[pseries::do\\_print\\_python](#)). [print\\_func](#)< [print\\_python\\_repr](#)>(&[pseries::do\\_print\\_python\\_repr](#)) [pseries](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([pseries](#))

### 7.72.1 Detailed Description

Implementation of class for extended truncated power series and methods for series expansion.

## 7.73 pseries.h File Reference

Interface to class for extended truncated power series.

```
#include "basic.h"
#include "expairseq.h"
```

## Classes

- class [GiNaC::pseries](#)  
*This class holds a extended truncated power series (positive and negative integer powers).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([pseries](#))
- [ex GiNaC::series\\_to\\_poly](#) (const [ex](#) &e)  
*Convert the pseries object embedded in an expression to an ordinary polynomial in the expansion variable.*
- bool [GiNaC::is\\_terminating](#) (const [pseries](#) &s)

### 7.73.1 Detailed Description

Interface to class for extended truncated power series.

## 7.74 ptr.h File Reference

Reference-counted pointer template.

```
#include "assertion.h"
#include <cstddef>
#include <functional>
#include <iosfwd>
```

### Classes

- class [GiNaC::refcounted](#)  
*Base class for reference-counted objects.*
- class [GiNaC::ptr< T >](#)  
*Class of (intrusively) reference-counted pointers that support copy-on-write semantics.*
- struct [std::less< GiNaC::ptr< T > >](#)  
*Specialization of std::less for ptr<T> to enable ordering of ptr<T> objects (e.g.*

### Namespaces

- namespace [GiNaC](#)
- namespace [std](#)

### 7.74.1 Detailed Description

Reference-counted pointer template.

## 7.75 registrar.cpp File Reference

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

```
#include "registrar.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### 7.75.1 Detailed Description

[GiNaC](#)'s class registrar (for class basic and all classes derived from it).

## 7.76 registrar.h File Reference

GiNaC's class registrar (for class basic and all classes derived from it).

```
#include "class_info.h"
#include "print.h"
#include <list>
#include <string>
#include <typeinfo>
#include <vector>
```

### Classes

- struct [GiNaC::return\\_type\\_t](#)  
*To distinguish between different kinds of non-commutative objects.*
- class [GiNaC::registered\\_class\\_options](#)  
*This class stores information about a registered [GiNaC](#) class.*

### Namespaces

- namespace [GiNaC](#)

### Macros

- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS\\_COMMON](#)(classname)  
*Common part of [GINAC\\_DECLARE\\_REGISTERED\\_CLASS\\_NO\\_CTORS](#) and [GINAC\\_DECLARE\\_REGISTERED\\_CLASS](#).*
- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS\\_NO\\_CTORS](#)(classname, supertype)  
*Primary macro for inclusion in the declaration of each registered class.*
- #define [GINAC\\_DECLARE\\_REGISTERED\\_CLASS](#)(classname, supertype)  
*Macro for inclusion in the declaration of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS](#)(classname, supertype) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)));  
*Macro for inclusion in the implementation of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#)(classname, supertype, options) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)).options);  
*Macro for inclusion in the implementation of each registered class.*
- #define [GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT\\_T](#)(classname, supertype, options) [GiNaC::registered\\_class\\_info](#) classname::reg\_info = [GiNaC::registered\\_class\\_info](#)([GiNaC::registered\\_class\\_options](#)(#classname, #supertype, typeid(classname)).options);  
*Macro for inclusion in the implementation of each registered class.*

### Typedefs

- typedef [class\\_info](#)< [registered\\_class\\_options](#) > [GiNaC::registered\\_class\\_info](#)

## Functions

- `template<typename T>`  
`return_type_t GiNaC::make_return_type_t (const unsigned rl=0)`
- `template<class Alg, class Ctx, class T, class C>`  
`void GiNaC::set_print_func (void f(const T &, const C &c, unsigned))`  
*Add or replace a print method.*
- `template<class Alg, class Ctx, class T, class C>`  
`void GiNaC::set_print_func (void(T::*f)(const C &, unsigned))`  
*Add or replace a print method.*

### 7.76.1 Detailed Description

GiNaC's class registrar (for class basic and all classes derived from it).

### 7.76.2 Macro Definition Documentation

#### 7.76.2.1 GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON

```
#define GINAC_DECLARE_REGISTERED_CLASS_COMMON(
 classname)
```

##### Value:

```
private: \
 static GiNaC::registered_class_info reg_info; \
public: \
 static GiNaC::registered_class_info &get_class_info_static() { return reg_info; } \
 class visitor { \
 public: \
 virtual void visit(const classname &) = 0; \
 virtual ~visitor() {}; \
 };
```

Common part of GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS and GINAC\_DECLARE\_REGISTERED\_↵  
\_CLASS.

#### 7.76.2.2 GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS

```
#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(
 classname,
 supertype)
```

##### Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
typedef supertype inherited; \
virtual const GiNaC::registered_class_info &get_class_info() const { return \
 classname::get_class_info_static(); } \
virtual GiNaC::registered_class_info &get_class_info() { return classname::get_class_info_static(); } \
virtual const char *class_name() const { return classname::get_class_info_static().options.get_name(); } \
private:
```

Primary macro for inclusion in the declaration of each registered class.

### 7.76.2.3 GINAC\_DECLARE\_REGISTERED\_CLASS

```
#define GINAC_DECLARE_REGISTERED_CLASS(
 classname,
 supertype)
```

Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
template<class B, typename... Args> friend B & dynallocate(Args &&... args); \
typedef supertype inherited; \
classname(); \
classname * duplicate() const override { \
 classname * bp = new classname(*this); \
 bp->setflag(GiNaC::status_flags::dynallocated); \
 return bp; \
} \
void accept(GiNaC::visitor & v) const override \
{ \
 if (visitor *p = dynamic_cast<visitor *>(&v)) \
 p->visit(*this); \
 else \
 inherited::accept(v); \
} \
const GiNaC::registered_class_info &get_class_info() const override { return \
 classname::get_class_info_static(); } \
GiNaC::registered_class_info &get_class_info() override { return classname::get_class_info_static(); } \
const char *class_name() const override { return classname::get_class_info_static().options.get_name(); \
} \
protected: \
 int compare_same_type(const GiNaC::basic & other) const override; \
private:
```

Macro for inclusion in the declaration of each registered class.

It declares some functions that are common to all classes derived from 'basic' as well as all required stuff for the [GiNaC](#) class registry (mainly needed for archiving).

### 7.76.2.4 GINAC\_IMPLEMENT\_REGISTERED\_CLASS

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS(
 classname,
 supertype) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::status_flags::dynallocated, typeid(classname));
```

Macro for inclusion in the implementation of each registered class.

### 7.76.2.5 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(
 classname,
 supertype,
 options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::status_flags::dynallocated, typeid(classname), options);
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

### 7.76.2.6 GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T

```
#define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(
 classname,
 supename,
 options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC
#supename, typeid(classname)).options);
```

Macro for inclusion in the implementation of each registered class.

Additional options can be specified.

## 7.77 relational.cpp File Reference

Implementation of relations between expressions.

```
#include "relational.h"
#include "operators.h"
#include "numeric.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
#include <stdexcept>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([relational](#), [basic](#), [print\\_func< print\\_context >\(&relational::do\\_print\)](#), [print\\_func< print\\_tree >\(&relational::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&relational::do\\_print\\_python\\_repr\)](#)) [relational](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([relational](#))
- static void [GiNaC::print\\_operator](#) (const [print\\_context](#) &c, [relational::operators](#) o)

### 7.77.1 Detailed Description

Implementation of relations between expressions.

## 7.78 relational.h File Reference

Interface to relations between expressions.

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

## Classes

- class [GiNaC::relational](#)  
*This class holds a relation consisting of two expressions and a logical relation between them.*
- struct [GiNaC::relational::safe\\_bool\\_helper](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([relational](#))

### 7.78.1 Detailed Description

Interface to relations between expressions.

## 7.79 remember.cpp File Reference

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

```
#include "function.h"
#include "utils.h"
#include "remember.h"
#include <stdexcept>
```

## Namespaces

- namespace [GiNaC](#)

### 7.79.1 Detailed Description

Implementation of helper classes for using the remember option in [GiNaC](#) functions.

## 7.80 remember.h File Reference

Interface to helper classes for using the remember option in [GiNaC](#) functions.

```
#include <iosfwd>
#include <list>
#include <vector>
```

## Classes

- class [GiNaC::remember\\_table\\_entry](#)  
*A single entry in the remember table of a function.*
- class [GiNaC::remember\\_table\\_list](#)  
*A list of entries in the remember table having some least significant bits of the hashvalue in common.*
- class [GiNaC::remember\\_table](#)  
*The remember table is organized like an n-fold associative cache in a microprocessor.*

## Namespaces

- namespace [GiNaC](#)

### 7.80.1 Detailed Description

Interface to helper classes for using the remember option in [GiNaC](#) functions.

## 7.81 structure.h File Reference

Wrapper template for making [GiNaC](#) classes out of C++ structures.

```
#include "ex.h"
#include "ncmul.h"
#include "numeric.h"
#include "operators.h"
#include "print.h"
#include <functional>
```

## Classes

- class [GiNaC::compare\\_all\\_equal< T >](#)  
*Comparison policy: all structures of one type are equal.*
- class [GiNaC::compare\\_std\\_less< T >](#)  
*Comparison policy: use std::equal\_to/std::less (defaults to operators == and <) to compare structures.*
- class [GiNaC::compare\\_bitwise< T >](#)  
*Comparison policy: use bit-wise comparison to compare structures.*
- class [GiNaC::structure< T, ComparisonPolicy >](#)  
*Wrapper template for making [GiNaC](#) classes out of C++ structures.*

## Namespaces

- namespace [GiNaC](#)

### 7.81.1 Detailed Description

Wrapper template for making [GiNaC](#) classes out of C++ structures.

## 7.82 symbol.cpp File Reference

Implementation of [GiNaC](#)'s symbolic objects.

```
#include "symbol.h"
#include "lst.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include "inifcns.h"
#include <map>
#include <stdexcept>
#include <string>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) (symbol, basic, print\_func< [print\\_context](#) >(&symbol::do\_print). print\_func< [print\\_latex](#) >(&symbol::do\_print\_latex). print\_func< [print\\_tree](#) >(&symbol::do\_print\_tree). print\_func< [print\\_python\\_repr](#) >(&symbol::do\_print\_python\_repr)) symbol
- static const std::string & [GiNaC::get\\_default\\_TeX\\_name](#) (const std::string &name)  
*Return default TeX name for symbol.*
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (symbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (realsymbol)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) (possymbol)

### 7.82.1 Detailed Description

Implementation of [GiNaC](#)'s symbolic objects.

## 7.83 symbol.h File Reference

Interface to [GiNaC](#)'s symbolic objects.

```
#include "basic.h"
#include "ex.h"
#include "ptr.h"
#include "archive.h"
#include <string>
#include <typeinfo>
```

## Classes

- class [GiNaC::symbol](#)  
*Basic CAS symbol.*
- class [GiNaC::realsymbol](#)  
*Specialization of symbol to real domain.*
- class [GiNaC::possymbol](#)  
*Specialization of symbol to real positive domain.*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([symbol](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([realsymbol](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([possymbol](#))

### 7.83.1 Detailed Description

Interface to [GiNaC](#)'s symbolic objects.

## 7.84 symmetry.cpp File Reference

Implementation of [GiNaC](#)'s symmetry definitions.

```
#include "symmetry.h"
#include "lst.h"
#include "add.h"
#include "numeric.h"
#include "operators.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <functional>
#include <iostream>
#include <limits>
#include <stdexcept>
```

## Classes

- class [GiNaC::sy\\_is\\_less](#)
- class [GiNaC::sy\\_swap](#)

## Namespaces

- namespace [GiNaC](#)

## Functions

- `GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT` (`symmetry`, `basic`, `print_func` < `print_context` >(&`symmetry::do_print`). `print_func` < `print_tree` >(&`symmetry::do_print_tree`)) `symmetry`
- `GiNaC::GINAC_BIND_UNARCHIVER` (`symmetry`)
- static const `symmetry` & `GiNaC::index0` ()
- static const `symmetry` & `GiNaC::index1` ()
- static const `symmetry` & `GiNaC::index2` ()
- static const `symmetry` & `GiNaC::index3` ()
- const `symmetry` & `GiNaC::not_symmetric` ()
- const `symmetry` & `GiNaC::symmetric2` ()
- const `symmetry` & `GiNaC::symmetric3` ()
- const `symmetry` & `GiNaC::symmetric4` ()
- const `symmetry` & `GiNaC::antisymmetric2` ()
- const `symmetry` & `GiNaC::antisymmetric3` ()
- const `symmetry` & `GiNaC::antisymmetric4` ()
- int `GiNaC::canonicalize` (`exvector::iterator` v, const `symmetry` &`symm`)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- static `ex` `GiNaC::symm` (const `ex` &`e`, `exvector::const_iterator` first, `exvector::const_iterator` last, bool `asymmetric`)
- `ex` `GiNaC::symmetrize` (const `ex` &`e`, `exvector::const_iterator` first, `exvector::const_iterator` last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- `ex` `GiNaC::antisymmetrize` (const `ex` &`e`, `exvector::const_iterator` first, `exvector::const_iterator` last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- `ex` `GiNaC::symmetrize_cyclic` (const `ex` &`e`, `exvector::const_iterator` first, `exvector::const_iterator` last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.84.1 Detailed Description

Implementation of `GiNaC`'s symmetry definitions.

## 7.85 symmetry.h File Reference

Interface to `GiNaC`'s symmetry definitions.

```
#include "ex.h"
#include "archive.h"
#include <set>
```

## Classes

- class `GiNaC::symmetry`  
*This class describes the symmetry of a group of indices.*

## Namespaces

- namespace `GiNaC`

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([symmetry](#))
- [symmetry GiNaC::sy\\_none](#) ()
- [symmetry GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry GiNaC::sy\\_none](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry GiNaC::sy\\_symm](#) ()
- [symmetry GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry GiNaC::sy\\_symm](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry GiNaC::sy\\_anti](#) ()
- [symmetry GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry GiNaC::sy\\_anti](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- [symmetry GiNaC::sy\\_cycl](#) ()
- [symmetry GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2)
- [symmetry GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3)
- [symmetry GiNaC::sy\\_cycl](#) (const [symmetry](#) &c1, const [symmetry](#) &c2, const [symmetry](#) &c3, const [symmetry](#) &c4)
- const [symmetry](#) & [GiNaC::not\\_symmetric](#) ()
- const [symmetry](#) & [GiNaC::symmetric2](#) ()
- const [symmetry](#) & [GiNaC::symmetric3](#) ()
- const [symmetry](#) & [GiNaC::symmetric4](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric2](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric3](#) ()
- const [symmetry](#) & [GiNaC::antisymmetric4](#) ()
- int [GiNaC::canonicalize](#) (exvector::iterator v, const [symmetry](#) &symm)  
*Canonicalize the order of elements of an expression vector, according to the symmetry properties defined in a symmetry tree.*
- [ex GiNaC::symmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex GiNaC::symmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression over a set of objects (symbols, indices).*
- [ex GiNaC::antisymmetrize](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex GiNaC::antisymmetrize](#) (const [ex](#) &e, const [exvector](#) &v)  
*Antisymmetrize expression over a set of objects (symbols, indices).*
- [ex GiNaC::symmetrize\\_cyclic](#) (const [ex](#) &e, exvector::const\_iterator first, exvector::const\_iterator last)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*
- [ex GiNaC::symmetrize\\_cyclic](#) (const [ex](#) &e, const [exvector](#) &v)  
*Symmetrize expression by cyclic permutation over a set of objects (symbols, indices).*

### 7.85.1 Detailed Description

Interface to [GiNaC](#)'s symmetry definitions.

## 7.86 tensor.cpp File Reference

Implementation of [GiNaC](#)'s special tensors.

```
#include "tensor.h"
#include "idx.h"
#include "indexed.h"
#include "symmetry.h"
#include "relational.h"
#include "operators.h"
#include "lst.h"
#include "numeric.h"
#include "matrix.h"
#include "archive.h"
#include "utils.h"
#include <iostream>
#include <stdexcept>
#include <vector>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([tensdelta](#), [tensor](#), [print\\_func< print\\_dflt >](#)(&[tensdelta::do\\_print](#)), [print\\_func< print\\_latex >](#)(&[tensdelta::do\\_print\\_latex](#))) [GINAC\\_IMPLEMENT\\_](#)↵  
[REGISTERED\\_CLASS\\_OPT](#)([tensmetric](#)
- [GiNaC::print\\_func< print\\_dflt >](#) (&[tensmetric::do\\_print](#)). [print\\_func< print\\_latex >](#)(&[tensmetric](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([minkmetric](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([tensepsilon](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([tensdelta](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([tensmetric](#))
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([spinmetric](#))
- [ex GiNaC::delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex GiNaC::metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex GiNaC::lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex GiNaC::spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex GiNaC::lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.86.1 Detailed Description

Implementation of [GiNaC](#)'s special tensors.

## 7.87 tensor.h File Reference

Interface to [GiNaC](#)'s special tensors.

```
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::tensor](#)  
*This class holds one of [GiNaC](#)'s predefined special tensors such as the delta and the metric tensors.*
- class [GiNaC::tensdelta](#)  
*This class represents the delta tensor.*
- class [GiNaC::tensmetric](#)  
*This class represents a general metric tensor which can be used to raise/lower indices.*
- class [GiNaC::minkmetric](#)  
*This class represents a Minkowski metric tensor.*
- class [GiNaC::spinmetric](#)  
*This class represents an antisymmetric spinor metric tensor which can be used to raise/lower indices of 2-component Weyl spinors.*
- class [GiNaC::tensepsilon](#)  
*This class represents the totally antisymmetric epsilon tensor.*

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensdelta](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([minkmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([spinmetric](#))
- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([tensepsilon](#))
- [ex GiNaC::delta\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a delta tensor with specified indices.*
- [ex GiNaC::metric\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a symmetric metric tensor with specified indices.*
- [ex GiNaC::lorentz\\_g](#) (const [ex](#) &i1, const [ex](#) &i2, bool pos\_sig=false)  
*Create a Minkowski metric tensor with specified indices.*
- [ex GiNaC::spinor\\_metric](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create a spinor metric tensor with specified indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2)  
*Create an epsilon tensor in a Euclidean space with two indices.*
- [ex GiNaC::epsilon\\_tensor](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3)  
*Create an epsilon tensor in a Euclidean space with three indices.*
- [ex GiNaC::lorentz\\_eps](#) (const [ex](#) &i1, const [ex](#) &i2, const [ex](#) &i3, const [ex](#) &i4, bool pos\_sig=false)  
*Create an epsilon tensor in a Minkowski space with four indices.*

### 7.87.1 Detailed Description

Interface to [GiNaC](#)'s special tensors.

## 7.88 utils.cpp File Reference

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "ex.h"
#include "numeric.h"
#include "utils.h"
#include "version.h"
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- unsigned [GiNaC::log2](#) (unsigned n)  
*Integer binary logarithm.*
- const [numeric GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*

### Variables

- const int [GiNaC::version\\_major](#) = GINACLIB\_MAJOR\_VERSION
- const int [GiNaC::version\\_minor](#) = GINACLIB\_MINOR\_VERSION
- const int [GiNaC::version\\_micro](#) = GINACLIB\_MICRO\_VERSION
- const [numeric](#) \* [GiNaC::\\_num\\_120\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_120](#) = [ex](#)(\*[\\_num\\_120\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_60\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_60](#) = [ex](#)(\*[\\_num\\_60\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_48\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_48](#) = [ex](#)(\*[\\_num\\_48\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_30\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_30](#) = [ex](#)(\*[\\_num\\_30\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_25\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_25](#) = [ex](#)(\*[\\_num\\_25\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_24\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_24](#) = [ex](#)(\*[\\_num\\_24\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_20\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_20](#) = [ex](#)(\*[\\_num\\_20\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_18\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_18](#) = [ex](#)(\*[\\_num\\_18\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_15\\_p](#)
- const [ex](#) [GiNaC::\\_ex\\_15](#) = [ex](#)(\*[\\_num\\_15\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num\\_12\\_p](#)

- `const ex GiNaC::_ex_12 = ex(*_num_12_p)`
- `const numeric * GiNaC::_num_11_p`
- `const ex GiNaC::_ex_11 = ex(*_num_11_p)`
- `const numeric * GiNaC::_num_10_p`
- `const ex GiNaC::_ex_10 = ex(*_num_10_p)`
- `const numeric * GiNaC::_num_9_p`
- `const ex GiNaC::_ex_9 = ex(*_num_9_p)`
- `const numeric * GiNaC::_num_8_p`
- `const ex GiNaC::_ex_8 = ex(*_num_8_p)`
- `const numeric * GiNaC::_num_7_p`
- `const ex GiNaC::_ex_7 = ex(*_num_7_p)`
- `const numeric * GiNaC::_num_6_p`
- `const ex GiNaC::_ex_6 = ex(*_num_6_p)`
- `const numeric * GiNaC::_num_5_p`
- `const ex GiNaC::_ex_5 = ex(*_num_5_p)`
- `const numeric * GiNaC::_num_4_p`
- `const ex GiNaC::_ex_4 = ex(*_num_4_p)`
- `const numeric * GiNaC::_num_3_p`
- `const ex GiNaC::_ex_3 = ex(*_num_3_p)`
- `const numeric * GiNaC::_num_2_p`
- `const ex GiNaC::_ex_2 = ex(*_num_2_p)`
- `const numeric * GiNaC::_num_1_p`
- `const ex GiNaC::_ex_1 = ex(*_num_1_p)`
- `const numeric * GiNaC::_num_1_2_p`
- `const ex GiNaC::_ex_1_2 = ex(*_num_1_2_p)`
- `const numeric * GiNaC::_num_1_3_p`
- `const ex GiNaC::_ex_1_3 = ex(*_num_1_3_p)`
- `const numeric * GiNaC::_num_1_4_p`
- `const ex GiNaC::_ex_1_4 = ex(*_num_1_4_p)`
- `const numeric * GiNaC::_num0_p`
- `const ex GiNaC::_ex0 = ex(*_num0_p)`
- `const numeric * GiNaC::_num1_4_p`
- `const ex GiNaC::_ex1_4 = ex(*_num1_4_p)`
- `const numeric * GiNaC::_num1_3_p`
- `const ex GiNaC::_ex1_3 = ex(*_num1_3_p)`
- `const numeric * GiNaC::_num1_2_p`
- `const ex GiNaC::_ex1_2 = ex(*_num1_2_p)`
- `const numeric * GiNaC::_num1_p`
- `const ex GiNaC::_ex1 = ex(*_num1_p)`
- `const numeric * GiNaC::_num2_p`
- `const ex GiNaC::_ex2 = ex(*_num2_p)`
- `const numeric * GiNaC::_num3_p`
- `const ex GiNaC::_ex3 = ex(*_num3_p)`
- `const numeric * GiNaC::_num4_p`
- `const ex GiNaC::_ex4 = ex(*_num4_p)`
- `const numeric * GiNaC::_num5_p`
- `const ex GiNaC::_ex5 = ex(*_num5_p)`
- `const numeric * GiNaC::_num6_p`
- `const ex GiNaC::_ex6 = ex(*_num6_p)`
- `const numeric * GiNaC::_num7_p`
- `const ex GiNaC::_ex7 = ex(*_num7_p)`
- `const numeric * GiNaC::_num8_p`
- `const ex GiNaC::_ex8 = ex(*_num8_p)`
- `const numeric * GiNaC::_num9_p`
- `const ex GiNaC::_ex9 = ex(*_num9_p)`

- const [numeric](#) \* [GiNaC::\\_num10\\_p](#)
- const [ex](#) [GiNaC::\\_ex10](#) = [ex](#)(\*[\\_num10\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num11\\_p](#)
- const [ex](#) [GiNaC::\\_ex11](#) = [ex](#)(\*[\\_num11\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num12\\_p](#)
- const [ex](#) [GiNaC::\\_ex12](#) = [ex](#)(\*[\\_num12\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num15\\_p](#)
- const [ex](#) [GiNaC::\\_ex15](#) = [ex](#)(\*[\\_num15\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num18\\_p](#)
- const [ex](#) [GiNaC::\\_ex18](#) = [ex](#)(\*[\\_num18\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num20\\_p](#)
- const [ex](#) [GiNaC::\\_ex20](#) = [ex](#)(\*[\\_num20\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num24\\_p](#)
- const [ex](#) [GiNaC::\\_ex24](#) = [ex](#)(\*[\\_num24\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num25\\_p](#)
- const [ex](#) [GiNaC::\\_ex25](#) = [ex](#)(\*[\\_num25\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num30\\_p](#)
- const [ex](#) [GiNaC::\\_ex30](#) = [ex](#)(\*[\\_num30\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num48\\_p](#)
- const [ex](#) [GiNaC::\\_ex48](#) = [ex](#)(\*[\\_num48\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num60\\_p](#)
- const [ex](#) [GiNaC::\\_ex60](#) = [ex](#)(\*[\\_num60\\_p](#))
- const [numeric](#) \* [GiNaC::\\_num120\\_p](#)
- const [ex](#) [GiNaC::\\_ex120](#) = [ex](#)(\*[\\_num120\\_p](#))

### 7.88.1 Detailed Description

Implementation of several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89 utils.h File Reference

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

```
#include "assertion.h"
#include <functional>
#include <cstdint>
#include <string>
```

### Classes

- class [GiNaC::dunno](#)  
*Exception class thrown by functions to signal unimplemented functionality so the expression may just be .hold()*
- class [GiNaC::basic\\_partition\\_generator](#)  
*Base class for generating all bounded combinatorial partitions of an integer n with exactly m parts in non-decreasing order.*
- struct [GiNaC::basic\\_partition\\_generator::mpartition2](#)
- class [GiNaC::partition\\_with\\_zero\\_parts\\_generator](#)

*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (including zero parts) in non-decreasing order.*

- class [GiNaC::partition\\_generator](#)

*Generate all bounded combinatorial partitions of an integer  $n$  with exactly  $m$  parts (not including zero parts) in non-decreasing order.*

- class [GiNaC::composition\\_generator](#)

*Generate all compositions of a partition of an integer  $n$ , starting with the compositions which has non-decreasing order.*

- struct [GiNaC::composition\\_generator::coolmulti](#)
- struct [GiNaC::composition\\_generator::coolmulti::element](#)

## Namespaces

- namespace [GiNaC](#)

## Macros

- #define [DEFAULT\\_CTOR](#)(classname) classname::classname() { setflag(status\_flags::evaluated | status\_↔ flags::expanded); }
- #define [DEFAULT\\_COMPARE](#)(classname)
- #define [DEFAULT\\_PRINT](#)(classname, text)
- #define [DEFAULT\\_PRINT\\_LATEX](#)(classname, text, latex)

## Functions

- unsigned [GiNaC::log2](#) (unsigned  $n$ )  
*Integer binary logarithm.*
- unsigned [GiNaC::rotate\\_left](#) (unsigned  $n$ )  
*Rotate bits of unsigned value by one bit to the left.*
- template<class T >  
int [GiNaC::compare\\_pointers](#) (const T \*a, const T \*b)  
*Compare two pointers (just to establish some sort of canonical order).*
- unsigned [GiNaC::golden\\_ratio\\_hash](#) (uintptr\_t  $n$ )  
*Truncated multiplication with golden ratio, for computing hash values.*
- template<class It >  
int [GiNaC::permutation\\_sign](#) (It first, It last)
- template<class It, class Cmp, class Swap >  
int [GiNaC::permutation\\_sign](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Cmp, class Swap >  
void [GiNaC::shaker\\_sort](#) (It first, It last, Cmp comp, Swap swapit)
- template<class It, class Swap >  
void [GiNaC::cyclic\\_permutation](#) (It first, It last, It new\_first, Swap swapit)
- const numeric [GiNaC::multinomial\\_coefficient](#) (const std::vector< unsigned > &p)  
*Compute the multinomial coefficient  $n!/(p_1! * p_2! * \dots * p_k!)$  where  $n = p_1 + p_2 + \dots + p_k$ , i.e.*

### 7.89.1 Detailed Description

Interface to several small and furry utilities needed within [GiNaC](#) but not of any interest to the user of the library.

## 7.89.2 Macro Definition Documentation

### 7.89.2.1 DEFAULT\_CTOR

```
#define DEFAULT_CTOR(
 classname) classname::classname() { setflag(status_flags::evaluated | status_↵
flags::expanded); }
```

### 7.89.2.2 DEFAULT\_COMPARE

```
#define DEFAULT_COMPARE(
 classname)
```

#### Value:

```
int classname::compare_same_type(const basic & other) const \
{ \
 /* by default, the objects are always identical */ \
 return 0; \
}
```

### 7.89.2.3 DEFAULT\_PRINT

```
#define DEFAULT_PRINT(
 classname,
 text)
```

#### Value:

```
void classname::do_print(const print_context & c, unsigned level) const \
{ \
 c.s << text; \
}
```

### 7.89.2.4 DEFAULT\_PRINT\_LATEX

```
#define DEFAULT_PRINT_LATEX(
 classname,
 text,
 latex)
```

#### Value:

```
DEFAULT_PRINT(classname, text) \
void classname::do_print_latex(const print_latex & c, unsigned level) const \
{ \
 c.s << latex; \
}
```

## 7.90 utils\_multi\_iterator.h File Reference

Utilities for summing over multiple indices.

```
#include <cstddef>
#include <vector>
#include <ostream>
#include <iterator>
```

## Classes

- class [GiNaC::has\\_distance< T >](#)  
*SFINAE test for distance.*
- class [GiNaC::basic\\_multi\\_iterator< T >](#)  
*basic\_multi\_iterator is a base class.*
- class [GiNaC::multi\\_iterator\\_ordered< T >](#)  
*The class multi\_iterator\_ordered defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq< T >](#)  
*The class multi\_iterator\_ordered\_eq defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_ordered\\_eq\\_indv< T >](#)  
*The class multi\_iterator\_ordered\_eq\_indv defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_counter< T >](#)  
*The class multi\_iterator\_counter defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_counter\\_indv< T >](#)  
*The class multi\_iterator\_counter\_indv defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , such that.*
- class [GiNaC::multi\\_iterator\\_permutation< T >](#)  
*The class multi\_iterator\_permutation defines a multi\_iterator  $(i_1, i_2, \dots, i_k)$ , for which.*
- class [GiNaC::multi\\_iterator\\_shuffle< T >](#)  
*The class multi\_iterator\_shuffle defines a multi\_iterator, which runs over all shuffles of a and b.*
- class [GiNaC::multi\\_iterator\\_shuffle\\_prime< T >](#)  
*The class multi\_iterator\_shuffle\_prime defines a multi\_iterator, which runs over all shuffles of a and b, excluding the first one (a,b).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- `template<typename T >`  
`std::enable_if< has\_distance< T >::value, typename std::iterator_traits< T >::difference_type >::type`  
`GiNaC::format\_index\_value (const T &a, const T &b)`  
*For printing a multi-index: If the templates are used, where T is an iterator, printing the address where the iterator points to is not meaningful.*
- `template<typename T >`  
`std::enable_if<!has\_distance< T >::value, T >::type GiNaC::format\_index\_value (const T &a, const T &b)`  
*For all other cases we simply print the value.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const basic\_multi\_iterator< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi\_iterator\_ordered\_eq\_indv< T > &v)`  
*Output operator.*

- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_counter< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_counter_indv< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_permutation< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle< T > &v)`  
*Output operator.*
- `template<class T >`  
`std::ostream & GiNaC::operator<< (std::ostream &os, const multi_iterator_shuffle_prime< T > &v)`  
*Output operator.*

### 7.90.1 Detailed Description

Utilities for summing over multiple indices.

## 7.91 version.h File Reference

[GiNaC](#) library version information.

### Namespaces

- namespace [GiNaC](#)

### Macros

- `#define GINACLIB_MAJOR_VERSION 1`
- `#define GINACLIB_MINOR_VERSION 8`
- `#define GINACLIB_MICRO_VERSION 7`
- `#define GINAC_LT_CURRENT 12`
- `#define GINAC_LT_REVISION 6`
- `#define GINAC_LT_AGE 1`
- `#define GINACLIB_ARCHIVE_VERSION 3`
- `#define GINACLIB_ARCHIVE_AGE 3`
- `#define GINACLIB_STR_HELPER(x) #x`
- `#define GINACLIB_STR(x) GINACLIB_STR_HELPER(x)`
- `#define GINACLIB_VERSION`

### 7.91.1 Detailed Description

[GiNaC](#) library version information.

## 7.91.2 Macro Definition Documentation

### 7.91.2.1 GINACLIB\_MAJOR\_VERSION

```
#define GINACLIB_MAJOR_VERSION 1
```

### 7.91.2.2 GINACLIB\_MINOR\_VERSION

```
#define GINACLIB_MINOR_VERSION 8
```

### 7.91.2.3 GINACLIB\_MICRO\_VERSION

```
#define GINACLIB_MICRO_VERSION 7
```

### 7.91.2.4 GINAC\_LT\_CURRENT

```
#define GINAC_LT_CURRENT 12
```

### 7.91.2.5 GINAC\_LT\_REVISION

```
#define GINAC_LT_REVISION 6
```

### 7.91.2.6 GINAC\_LT\_AGE

```
#define GINAC_LT_AGE 1
```

### 7.91.2.7 GINACLIB\_ARCHIVE\_VERSION

```
#define GINACLIB_ARCHIVE_VERSION 3
```

### 7.91.2.8 GINACLIB\_ARCHIVE\_AGE

```
#define GINACLIB_ARCHIVE_AGE 3
```

### 7.91.2.9 GINACLIB\_STR\_HELPER

```
#define GINACLIB_STR_HELPER(
 x) #x
```

### 7.91.2.10 GINACLIB\_STR

```
#define GINACLIB_STR(
 x) GINACLIB_STR_HELPER(x)
```

### 7.91.2.11 GINACLIB\_VERSION

```
#define GINACLIB_VERSION
```

**Value:**

```
GINACLIB_STR(GINACLIB_MAJOR_VERSION) "." \
GINACLIB_STR(GINACLIB_MINOR_VERSION) "." \
GINACLIB_STR(GINACLIB_MICRO_VERSION)
```

## 7.92 wildcard.cpp File Reference

Implementation of [GiNaC](#)'s wildcard objects.

```
#include "wildcard.h"
#include "archive.h"
#include "utils.h"
#include "hash_seed.h"
#include <iostream>
```

### Namespaces

- namespace [GiNaC](#)

### Functions

- [GiNaC::GINAC\\_IMPLEMENT\\_REGISTERED\\_CLASS\\_OPT](#) ([wildcard](#), [basic](#), [print\\_func< print\\_context >\(&wildcard::do\\_print\)](#), [print\\_func< print\\_tree >\(&wildcard::do\\_print\\_tree\)](#), [print\\_func< print\\_python\\_repr >\(&wildcard::do\\_print\\_python\\_repr\)](#)) [wildcard](#)
- [GiNaC::GINAC\\_BIND\\_UNARCHIVER](#) ([wildcard](#))
- [bool GiNaC::haswild](#) (const [ex](#) &[x](#))

*Check whether [x](#) has a wildcard anywhere as a subexpression.*

### 7.92.1 Detailed Description

Implementation of [GiNaC](#)'s wildcard objects.

## 7.93 wildcard.h File Reference

Interface to [GiNaC](#)'s wildcard objects.

```
#include "ex.h"
#include "archive.h"
```

### Classes

- class [GiNaC::wildcard](#)

*This class acts as a wildcard for [subs\(\)](#), [match\(\)](#), [has\(\)](#) and [find\(\)](#).*

## Namespaces

- namespace [GiNaC](#)

## Functions

- [GiNaC::GINAC\\_DECLARE\\_UNARCHIVER](#) ([wildcard](#))
- [ex GiNaC::wild](#) (unsigned label=0)  
*Create a wildcard object with the specified label.*
- [bool GiNaC::haswild](#) (const [ex](#) &[x](#))  
*Check whether  $x$  has a wildcard anywhere as a subexpression.*

### 7.93.1 Detailed Description

Interface to [GiNaC](#)'s wildcard objects.

# Index

[\\_ex0](#)  
    [GiNaC, 264](#)

[\\_ex1](#)  
    [GiNaC, 265](#)

[\\_ex10](#)  
    [GiNaC, 268](#)

[\\_ex11](#)  
    [GiNaC, 269](#)

[\\_ex12](#)  
    [GiNaC, 269](#)

[\\_ex120](#)  
    [GiNaC, 271](#)

[\\_ex15](#)  
    [GiNaC, 269](#)

[\\_ex18](#)  
    [GiNaC, 270](#)

[\\_ex1\\_2](#)  
    [GiNaC, 265](#)

[\\_ex1\\_3](#)  
    [GiNaC, 265](#)

[\\_ex1\\_4](#)  
    [GiNaC, 265](#)

[\\_ex2](#)  
    [GiNaC, 266](#)

[\\_ex20](#)  
    [GiNaC, 270](#)

[\\_ex24](#)  
    [GiNaC, 270](#)

[\\_ex25](#)  
    [GiNaC, 270](#)

[\\_ex3](#)  
    [GiNaC, 266](#)

[\\_ex30](#)  
    [GiNaC, 271](#)

[\\_ex4](#)  
    [GiNaC, 267](#)

[\\_ex48](#)  
    [GiNaC, 271](#)

[\\_ex5](#)  
    [GiNaC, 267](#)

[\\_ex6](#)  
    [GiNaC, 267](#)

[\\_ex60](#)  
    [GiNaC, 271](#)

[\\_ex7](#)  
    [GiNaC, 268](#)

[\\_ex8](#)  
    [GiNaC, 268](#)

[\\_ex9](#)  
    [GiNaC, 268](#)

[\\_ex\\_1](#)  
    [GiNaC, 263](#)

[\\_ex\\_10](#)  
    [GiNaC, 260](#)

[\\_ex\\_11](#)  
    [GiNaC, 260](#)

[\\_ex\\_12](#)  
    [GiNaC, 260](#)

[\\_ex\\_120](#)  
    [GiNaC, 258](#)

[\\_ex\\_15](#)  
    [GiNaC, 260](#)

[\\_ex\\_18](#)  
    [GiNaC, 259](#)

[\\_ex\\_1\\_2](#)  
    [GiNaC, 263](#)

[\\_ex\\_1\\_3](#)  
    [GiNaC, 263](#)

[\\_ex\\_1\\_4](#)  
    [GiNaC, 264](#)

[\\_ex\\_2](#)  
    [GiNaC, 262](#)

[\\_ex\\_20](#)  
    [GiNaC, 259](#)

[\\_ex\\_24](#)  
    [GiNaC, 259](#)

[\\_ex\\_25](#)  
    [GiNaC, 259](#)

[\\_ex\\_3](#)  
    [GiNaC, 262](#)

[\\_ex\\_30](#)  
    [GiNaC, 258](#)

[\\_ex\\_4](#)  
    [GiNaC, 262](#)

[\\_ex\\_48](#)  
    [GiNaC, 258](#)

[\\_ex\\_5](#)  
    [GiNaC, 262](#)

[\\_ex\\_6](#)  
    [GiNaC, 261](#)

[\\_ex\\_60](#)  
    [GiNaC, 258](#)

[\\_ex\\_7](#)  
    [GiNaC, 261](#)

[\\_ex\\_8](#)  
    [GiNaC, 261](#)

[\\_ex\\_9](#)  
    [GiNaC, 261](#)

\_iter\_rep  
     GiNaC::internal::\_iter\_rep, 274  
 \_num0\_bp  
     GiNaC, 256  
 \_num0\_p  
     GiNaC, 264  
 \_num10\_p  
     GiNaC, 268  
 \_num11\_p  
     GiNaC, 269  
 \_num120\_p  
     GiNaC, 271  
 \_num12\_p  
     GiNaC, 269  
 \_num15\_p  
     GiNaC, 269  
 \_num18\_p  
     GiNaC, 269  
 \_num1\_2\_p  
     GiNaC, 265  
 \_num1\_3\_p  
     GiNaC, 265  
 \_num1\_4\_p  
     GiNaC, 264  
 \_num1\_p  
     GiNaC, 265  
 \_num20\_p  
     GiNaC, 270  
 \_num24\_p  
     GiNaC, 270  
 \_num25\_p  
     GiNaC, 270  
 \_num2\_p  
     GiNaC, 266  
 \_num30\_p  
     GiNaC, 270  
 \_num3\_p  
     GiNaC, 266  
 \_num48\_p  
     GiNaC, 271  
 \_num4\_p  
     GiNaC, 267  
 \_num5\_p  
     GiNaC, 267  
 \_num60\_p  
     GiNaC, 271  
 \_num6\_p  
     GiNaC, 267  
 \_num7\_p  
     GiNaC, 267  
 \_num8\_p  
     GiNaC, 268  
 \_num9\_p  
     GiNaC, 268  
 \_num\_10\_p  
     GiNaC, 260  
 \_num\_11\_p  
     GiNaC, 260  
 \_num\_120\_p  
     GiNaC, 257  
 \_num\_12\_p  
     GiNaC, 260  
 \_num\_15\_p  
     GiNaC, 259  
 \_num\_18\_p  
     GiNaC, 259  
 \_num\_1\_2\_p  
     GiNaC, 263  
 \_num\_1\_3\_p  
     GiNaC, 263  
 \_num\_1\_4\_p  
     GiNaC, 264  
 \_num\_1\_p  
     GiNaC, 263  
 \_num\_20\_p  
     GiNaC, 259  
 \_num\_24\_p  
     GiNaC, 259  
 \_num\_25\_p  
     GiNaC, 258  
 \_num\_2\_p  
     GiNaC, 262  
 \_num\_30\_p  
     GiNaC, 258  
 \_num\_3\_p  
     GiNaC, 262  
 \_num\_48\_p  
     GiNaC, 258  
 \_num\_4\_p  
     GiNaC, 262  
 \_num\_5\_p  
     GiNaC, 261  
 \_num\_60\_p  
     GiNaC, 258  
 \_num\_6\_p  
     GiNaC, 261  
 \_num\_7\_p  
     GiNaC, 261  
 \_num\_8\_p  
     GiNaC, 261  
 \_num\_9\_p  
     GiNaC, 260  
 \_numeric\_digits  
     GiNaC::\_numeric\_digits, 276  
 ~archive  
     GiNaC::archive, 296  
 ~basic  
     GiNaC::basic, 320  
 ~basic\_multi\_iterator  
     GiNaC::basic\_multi\_iterator< T >, 348  
 ~compare\_all\_equal  
     GiNaC::compare\_all\_equal< T >, 389  
 ~compare\_bitwise  
     GiNaC::compare\_bitwise< T >, 390  
 ~compare\_std\_less  
     GiNaC::compare\_std\_less< T >, 391

- ~container\_storage
  - GiNaC::container\_storage< C >, 433
- ~coolmulti
  - GiNaC::composition\_generator::coolmulti, 435
- ~element
  - GiNaC::composition\_generator::coolmulti::element, 495
- ~function\_options
  - GiNaC::function\_options, 616
- ~library\_init
  - GiNaC::library\_init, 730
- ~map\_function
  - GiNaC::map\_function, 735
- ~print\_context
  - GiNaC::print\_context, 924
- ~print\_functor\_impl
  - GiNaC::print\_functor\_impl, 938
- ~ptr
  - GiNaC::ptr< T >, 973
- ~unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 1135
- ~visitor
  - GiNaC::visitor, 1153
- a
  - GiNaC::archive\_node, 311
  - GiNaC::Eisenstein\_kernel, 494
  - GiNaC::integral, 699
- abs
  - GiNaC, 214
- abs\_conjugate
  - GiNaC, 129
- abs\_eval
  - GiNaC, 128
- abs\_evalf
  - GiNaC, 128
- abs\_expand
  - GiNaC, 128
- abs\_expl\_derivative
  - GiNaC, 128
- abs\_imag\_part
  - GiNaC, 129
- abs\_info
  - GiNaC, 129
- abs\_power
  - GiNaC, 129
- abs\_print\_csrc\_float
  - GiNaC, 128
- abs\_print\_latex
  - GiNaC, 128
- abs\_real\_part
  - GiNaC, 129
- accept
  - GiNaC::basic, 327
  - GiNaC::ex, 525
- access\_counter
  - GiNaC::remember\_table\_entry, 1007
- acos
  - GiNaC, 206
- acos\_conjugate
  - GiNaC, 165
- acos\_deriv
  - GiNaC, 165
- acos\_eval
  - GiNaC, 164
- acos\_evalf
  - GiNaC, 164
- acosh
  - GiNaC, 208
- acosh\_conjugate
  - GiNaC, 172
- acosh\_deriv
  - GiNaC, 172
- acosh\_eval
  - GiNaC, 172
- acosh\_evalf
  - GiNaC, 172
- adaptivesimpson
  - GiNaC, 174
- add
  - GiNaC::add, 284, 285
  - GiNaC::matrix, 746
  - GiNaC::mul, 793
  - GiNaC::numeric, 860
  - GiNaC::scalar\_products, 1013
  - GiNaC::symmetry, 1101
- add.cpp, 1163
- add.h, 1164
- add\_bool
  - GiNaC::archive\_node, 306
- add\_callback
  - GiNaC::\_numeric\_digits, 276
- add\_child
  - GiNaC::class\_info< OPT >::tree\_node, 1134
- add\_dyn
  - GiNaC::numeric, 861
- add\_entry
  - GiNaC::remember\_table, 1003
  - GiNaC::remember\_table\_list, 1009
- add\_ex
  - GiNaC::archive\_node, 307
- add\_indexed
  - GiNaC::basic, 331
  - GiNaC::matrix, 744
  - GiNaC::structure< T, ComparisonPolicy >, 1050
- add\_node
  - GiNaC::archive, 299
- add\_reference
  - GiNaC::refcounted, 985
- add\_series
  - GiNaC::pseries, 966
- add\_string
  - GiNaC::archive\_node, 306
- add\_symbol
  - GiNaC, 187
- add\_unsigned
  - GiNaC::archive\_node, 306

- add\_vectors
  - GiNaC::scalar\_products, [1013](#)
- after\_i
  - GiNaC::composition\_generator::coolmulti, [436](#)
- algebraic
  - GiNaC::has\_options, [655](#)
  - GiNaC::subs\_options, [1076](#)
- algebraic\_match\_mul\_with\_mul
  - GiNaC, [186](#)
- algebraic\_subs\_mul
  - GiNaC::mul, [791](#)
- all
  - GiNaC::factor\_options, [571](#)
- all\_index\_values\_are
  - GiNaC::indexed, [684](#)
- antisymmetric
  - GiNaC::symmetry, [1099](#)
- antisymmetric2
  - GiNaC, [239](#)
- antisymmetric3
  - GiNaC, [239](#)
- antisymmetric4
  - GiNaC, [239](#)
- antisymmetrize
  - GiNaC, [108](#), [240](#), [243](#)
  - GiNaC::ex, [537](#), [538](#)
- append
  - GiNaC::container< C >, [428](#)
- append\_factors
  - GiNaC::ncmul, [843](#)
- archive
  - GiNaC::archive, [296](#)
  - GiNaC::basic, [335](#)
  - GiNaC::clifford, [366](#)
  - GiNaC::color, [386](#)
  - GiNaC::constant, [412](#)
  - GiNaC::container< C >, [425](#)
  - GiNaC::expairseq, [559](#)
  - GiNaC::fderivative, [586](#)
  - GiNaC::function, [604](#)
  - GiNaC::idx, [663](#)
  - GiNaC::indexed, [682](#)
  - GiNaC::integral, [697](#)
  - GiNaC::matrix, [745](#)
  - GiNaC::minkmetric, [763](#)
  - GiNaC::numeric, [858](#)
  - GiNaC::power, [918](#)
  - GiNaC::pseries, [964](#)
  - GiNaC::relational, [996](#)
  - GiNaC::spinidx, [1024](#)
  - GiNaC::symbol, [1088](#)
  - GiNaC::symmetry, [1100](#)
  - GiNaC::tensepsilon, [1118](#)
  - GiNaC::varidx, [1150](#)
  - GiNaC::wildcard, [1158](#)
- archive.cpp, [1164](#)
- archive.h, [1165](#)
  - GINAC\_BIND\_UNARCHIVER, [1167](#)
  - GINAC\_DECLARE\_UNARCHIVER, [1166](#)
- archive\_atom
  - GiNaC, [58](#)
- archive\_ex
  - GiNaC::archive, [296](#)
- archive\_node
  - GiNaC::archive\_node, [306](#)
  - GiNaC::ex, [541](#)
- archive\_node\_cit
  - GiNaC::archive\_node, [305](#)
- archive\_node\_id
  - GiNaC, [58](#)
- archived\_ex
  - GiNaC::archive::archived\_ex, [314](#)
- are\_ex\_trivially\_equal
  - GiNaC, [101](#)
  - GiNaC::ex, [541](#)
- arg1
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [882](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [885](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [887](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [893](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [895](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [898](#)
- arg2
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [885](#)
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [887](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [895](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [899](#)
- arg3
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [887](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [899](#)
- argument\_type
  - GiNaC::map\_function, [735](#)
- asin
  - GiNaC, [206](#)
- asin\_conjugate
  - GiNaC, [164](#)
- asin\_deriv
  - GiNaC, [164](#)
- asin\_eval
  - GiNaC, [163](#)
- asin\_evalf
  - GiNaC, [163](#)
- asin\_info
  - GiNaC, [164](#)

- asinh
  - GiNaC, [208](#)
- asinh\_conjugate
  - GiNaC, [171](#)
- asinh\_deriv
  - GiNaC, [171](#)
- asinh\_eval
  - GiNaC, [171](#)
- asinh\_evalf
  - GiNaC, [171](#)
- assertion.h, [1167](#)
  - GINAC\_ASSERT, [1168](#)
- atan
  - GiNaC, [206](#), [207](#)
- atan2\_deriv
  - GiNaC, [167](#)
- atan2\_eval
  - GiNaC, [166](#)
- atan2\_evalf
  - GiNaC, [166](#)
- atan2\_info
  - GiNaC, [167](#)
- atan\_conjugate
  - GiNaC, [166](#)
- atan\_deriv
  - GiNaC, [165](#)
- atan\_eval
  - GiNaC, [165](#)
- atan\_evalf
  - GiNaC, [165](#)
- atan\_info
  - GiNaC, [166](#)
- atan\_series
  - GiNaC, [166](#)
- atanh
  - GiNaC, [209](#)
- atanh\_conjugate
  - GiNaC, [173](#)
- atanh\_deriv
  - GiNaC, [173](#)
- atanh\_eval
  - GiNaC, [172](#)
- atanh\_evalf
  - GiNaC, [172](#)
- atanh\_series
  - GiNaC, [173](#)
- atend
  - GiNaC::composition\_generator, [393](#)
- atomize
  - GiNaC::archive, [300](#)
- atoms
  - GiNaC::archive, [301](#)
- attribute\_deprecated
  - compiler.h, [1179](#)
- automatic
  - GiNaC::determinant\_algo, [439](#)
  - GiNaC::solve\_algo, [1016](#)
- B
  - GiNaC::basic\_multi\_iterator< T >, [351](#)
- b
  - GiNaC::Eisenstein\_kernel, [494](#)
  - GiNaC::integral, [699](#)
- bareiss
  - GiNaC::determinant\_algo, [439](#)
  - GiNaC::solve\_algo, [1016](#)
- base\_and\_index
  - GiNaC, [86](#)
- basic
  - GiNaC::basic, [320](#)
- basic.cpp, [1169](#)
- basic.h, [1170](#)
- basic\_multi\_iterator
  - GiNaC::basic\_multi\_iterator< T >, [348](#)
- basic\_partition\_generator
  - GiNaC::basic\_partition\_generator, [353](#)
- basis
  - GiNaC::power, [923](#)
- begin
  - GiNaC::archive\_node::archive\_node\_cit\_range, [313](#)
  - GiNaC::container< C >, [429](#)
  - GiNaC::ex, [517](#)
- bernoulli
  - GiNaC, [213](#)
- Bernoulli\_polynomial
  - GiNaC, [176](#)
- beta\_deriv
  - GiNaC, [148](#)
- beta\_eval
  - GiNaC, [148](#)
- beta\_evalf
  - GiNaC, [148](#)
- beta\_series
  - GiNaC, [148](#)
- binomial
  - GiNaC, [213](#)
- binomial\_conjugate
  - GiNaC, [137](#)
- binomial\_eval
  - GiNaC, [137](#)
- binomial\_evalf
  - GiNaC, [136](#)
- binomial\_imag\_part
  - GiNaC, [137](#)
- binomial\_real\_part
  - GiNaC, [137](#)
- binomial\_sym
  - GiNaC, [136](#)
- bp
  - GiNaC::ex, [542](#)
- c
  - factor.cpp, [1192](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, [890](#)
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [893](#)

- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 895
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 898
- C\_norm
  - GiNaC::Eisenstein\_h\_kernel, 484
  - GiNaC::Eisenstein\_kernel, 494
  - GiNaC::Kronecker\_dtau\_kernel, 719
  - GiNaC::Kronecker\_dz\_kernel, 727
  - GiNaC::modular\_form\_kernel, 773
- cache
  - factor.cpp, 1194
- cache\_step\_size
  - GiNaC::integration\_kernel, 707
- cache\_vec
  - integration\_kernel.cpp, 1236
- calc\_lanczos\_A
  - GiNaC::lanczos\_coefs, 728
- calchash
  - GiNaC::basic, 334
  - GiNaC::constant, 414
  - GiNaC::expairseq, 560
  - GiNaC::function, 602
  - GiNaC::idx, 664
  - GiNaC::numeric, 859
  - GiNaC::relational, 998
  - GiNaC::structure< T, ComparisonPolicy >, 1053
  - GiNaC::symbol, 1090
  - GiNaC::symmetry, 1100
  - GiNaC::wildcard, 1158
- callback\_registered
  - GiNaC, 82
- callbacklist
  - GiNaC::\_numeric\_digits, 277
- can\_be\_further\_expanded
  - GiNaC::mul, 792
- can\_make\_flat
  - GiNaC::expairseq, 563
  - GiNaC::mul, 790
- canonical
  - GiNaC::relational, 997
- canonicalize
  - GiNaC, 239
  - GiNaC::expairseq, 565
  - GiNaC::symmetry, 1102
- canonicalize\_clifford
  - GiNaC, 90
- Catalan
  - GiNaC, 255
- CatalanEvalf
  - GiNaC, 218
- charpoly
  - GiNaC, 184
  - GiNaC::matrix, 750
- children
  - GiNaC::class\_info< OPT >::tree\_node, 1134
  - GiNaC::symmetry, 1103
- cinteger
  - GiNaC::info\_flags, 688
  - cinteger\_polynomial
    - GiNaC::info\_flags, 688
  - class\_info
    - GiNaC::class\_info< OPT >, 354
  - class\_info.h, 1171
  - clear
    - GiNaC::archive, 299
    - GiNaC::scalar\_products, 1013
  - clear\_all\_entries
    - GiNaC::remember\_table, 1003
  - clearflag
    - GiNaC::basic, 338
  - clifford
    - GiNaC::clifford, 365
  - clifford.cpp, 1172
  - clifford.h, 1174
  - clifford\_bar
    - GiNaC, 95
  - clifford\_inverse
    - GiNaC, 92
  - clifford\_max\_label
    - GiNaC, 91
  - clifford\_moebius\_map
    - GiNaC, 93, 94
  - clifford\_norm
    - GiNaC, 92
  - clifford\_prime
    - GiNaC, 91
  - clifford\_star
    - GiNaC, 96
  - clifford\_star\_bar
    - GiNaC, 90
  - clifford\_to\_lst
    - GiNaC, 93
  - clifford\_unit
    - GiNaC, 87
  - cmgen
    - GiNaC::composition\_generator, 393
  - CMPINDICES
    - color.cpp, 1177
  - coeff
    - GiNaC, 104
    - GiNaC::add, 286
    - GiNaC::basic, 327
    - GiNaC::ex, 527
    - GiNaC::expair, 548
    - GiNaC::mul, 783
    - GiNaC::ncmul, 840
    - GiNaC::numeric, 855
    - GiNaC::power, 914
    - GiNaC::pseries, 961
    - GiNaC::structure< T, ComparisonPolicy >, 1046
    - GiNaC::symminfo, 1105
  - coefficient\_a0
    - GiNaC::Eisenstein\_h\_kernel, 483
  - coefficient\_an
    - GiNaC::Eisenstein\_h\_kernel, 483

- coeffop
  - GiNaC::pseries, [966](#)
- coeffs
  - GiNaC::lanczos\_coeffs, [728](#)
- coerce
  - GiNaC, [203](#)
- coerce< int, cln::cl\_I >
  - GiNaC, [203](#)
- coerce< unsigned int, cln::cl\_I >
  - GiNaC, [203](#)
- col
  - GiNaC::matrix, [756](#)
- collect
  - GiNaC, [106](#)
  - GiNaC::basic, [328](#)
  - GiNaC::ex, [528](#)
  - GiNaC::pseries, [961](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1046](#)
- collect\_common\_factors
  - GiNaC, [200](#)
- collect\_symbols
  - GiNaC, [188](#)
- color
  - GiNaC::color, [385](#)
- color.cpp, [1176](#)
  - CMPINDICES, [1177](#)
  - TEST\_PERMUTATION, [1177](#)
- color.h, [1178](#)
- color\_d
  - GiNaC, [98](#)
- color\_f
  - GiNaC, [98](#)
- color\_h
  - GiNaC, [99](#)
- color\_ONE
  - GiNaC, [97](#)
- color\_T
  - GiNaC, [97](#)
- color\_trace
  - GiNaC, [99](#), [100](#)
- cols
  - GiNaC, [183](#)
  - GiNaC::matrix, [746](#)
- combine\_ex\_with\_coeff\_to\_pair
  - GiNaC::add, [291](#)
  - GiNaC::expairseq, [562](#)
  - GiNaC::mul, [789](#)
- combine\_indices
  - GiNaC::make\_flat\_inserter, [732](#)
- combine\_overall\_coeff
  - GiNaC::expairseq, [563](#)
  - GiNaC::mul, [790](#)
- combine\_pair\_with\_coeff\_to\_pair
  - GiNaC::add, [292](#)
  - GiNaC::expairseq, [562](#)
  - GiNaC::mul, [789](#)
- combine\_same\_terms\_sorted\_seq
  - GiNaC::expairseq, [566](#)
- commutative
  - GiNaC::return\_types, [1012](#)
- commutator\_sign
  - GiNaC::clifford, [370](#)
- compare
  - GiNaC::basic, [337](#)
  - GiNaC::ex, [536](#)
  - GiNaC::expair, [547](#)
  - GiNaC::numeric, [864](#)
- compare\_pointers
  - GiNaC, [249](#)
- compare\_same\_type
  - GiNaC::basic, [334](#)
- compile\_ex
  - GiNaC, [111](#), [112](#)
- compiler.h, [1179](#)
  - attribute\_deprecated, [1179](#)
  - likely, [1179](#)
  - unlikely, [1179](#)
- complex
  - GiNaC::domain, [466](#)
- composition
  - GiNaC::composition\_generator, [393](#)
- composition\_generator
  - GiNaC::composition\_generator, [393](#)
- conjugate
  - GiNaC, [103](#)
  - GiNaC::add, [289](#)
  - GiNaC::basic, [333](#)
  - GiNaC::constant, [412](#)
  - GiNaC::container< C >, [426](#)
  - GiNaC::diracgamma5, [450](#)
  - GiNaC::diracgammaL, [455](#)
  - GiNaC::diracgammaR, [460](#)
  - GiNaC::ex, [522](#)
  - GiNaC::expair, [547](#)
  - GiNaC::expairseq, [558](#)
  - GiNaC::function, [603](#)
  - GiNaC::integral, [696](#)
  - GiNaC::matrix, [744](#)
  - GiNaC::mul, [787](#)
  - GiNaC::ncmul, [841](#)
  - GiNaC::numeric, [858](#)
  - GiNaC::power, [918](#)
  - GiNaC::pseries, [963](#)
  - GiNaC::realsymbol, [982](#)
  - GiNaC::spinidx, [1024](#)
  - GiNaC::symbol, [1088](#)
- conjugate\_conjugate
  - GiNaC, [124](#)
- conjugate\_eval
  - GiNaC, [124](#)
- conjugate\_evalf
  - GiNaC, [124](#)
- conjugate\_expl\_derivative
  - GiNaC, [124](#)
- conjugate\_f
  - GiNaC::function\_options, [646](#)

- conjugate\_func
  - GiNaC::function\_options, 620–622, 639
- conjugate\_funcp
  - GiNaC, 60
- conjugate\_funcp\_1
  - GiNaC, 61
- conjugate\_funcp\_10
  - GiNaC, 72
- conjugate\_funcp\_11
  - GiNaC, 74
- conjugate\_funcp\_12
  - GiNaC, 75
- conjugate\_funcp\_13
  - GiNaC, 77
- conjugate\_funcp\_14
  - GiNaC, 78
- conjugate\_funcp\_2
  - GiNaC, 62
- conjugate\_funcp\_3
  - GiNaC, 63
- conjugate\_funcp\_4
  - GiNaC, 65
- conjugate\_funcp\_5
  - GiNaC, 66
- conjugate\_funcp\_6
  - GiNaC, 67
- conjugate\_funcp\_7
  - GiNaC, 68
- conjugate\_funcp\_8
  - GiNaC, 70
- conjugate\_funcp\_9
  - GiNaC, 71
- conjugate\_funcp\_exvector
  - GiNaC, 80
- conjugate\_imag\_part
  - GiNaC, 125
- conjugate\_info
  - GiNaC, 125
- conjugate\_print\_latex
  - GiNaC, 124
- conjugate\_real\_part
  - GiNaC, 124
- conjugate\_use\_exvector\_args
  - GiNaC::function\_options, 649
- conjugateepvector
  - GiNaC, 114
- const\_iterator
  - GiNaC::const\_iterator, 396
  - GiNaC::container< C >, 422
- const\_postorder\_iterator
  - GiNaC::const\_iterator, 399
  - GiNaC::const\_postorder\_iterator, 401
- const\_preorder\_iterator
  - GiNaC::const\_iterator, 399
  - GiNaC::const\_preorder\_iterator, 404
- const\_reverse\_iterator
  - GiNaC::container< C >, 422
- constant
  - GiNaC::constant, 411
- constant.cpp, 1180
- constant.h, 1181
- construct\_from\_2\_ex
  - GiNaC::expairseq, 564
- construct\_from\_2\_expairseq
  - GiNaC::expairseq, 564
- construct\_from\_basic
  - GiNaC::ex, 539
- construct\_from\_double
  - GiNaC::ex, 540
- construct\_from\_epvector
  - GiNaC::expairseq, 565
- construct\_from\_expairseq\_ex
  - GiNaC::expairseq, 564
- construct\_from\_exvector
  - GiNaC::expairseq, 564
- construct\_from\_int
  - GiNaC::ex, 539
- construct\_from\_long
  - GiNaC::ex, 540
- construct\_from\_longlong
  - GiNaC::ex, 540
- construct\_from\_string\_and\_lst
  - GiNaC::ex, 540
- construct\_from\_uint
  - GiNaC::ex, 539
- construct\_from\_ulong
  - GiNaC::ex, 540
- construct\_from\_ulonglong
  - GiNaC::ex, 540
- cont
  - factor.cpp, 1197
- container
  - GiNaC::container< C >, 422, 423
- container.h, 1181
- container\_storage
  - GiNaC::container\_storage< C >, 433
- content
  - GiNaC::ex, 532
- contract\_with
  - GiNaC::basic, 332
  - GiNaC::cliffordunit, 376
  - GiNaC::diracgamma, 445
  - GiNaC::matrix, 744
  - GiNaC::spinmetric, 1032
  - GiNaC::structure< T, ComparisonPolicy >, 1051
  - GiNaC::su3d, 1059
  - GiNaC::su3f, 1064
  - GiNaC::su3t, 1075
  - GiNaC::tensdelta, 1111
  - GiNaC::tensepsilon, 1118
  - GiNaC::tensmetric, 1125
- convert\_H\_to\_Li
  - GiNaC, 142
- convert\_to\_poly
  - GiNaC::pseries, 965
- coolmulti

- GiNaC::composition\_generator::coolmulti, [435](#)
- cos
  - GiNaC, [205](#)
- cos\_conjugate
  - GiNaC, [162](#)
- cos\_deriv
  - GiNaC, [161](#)
- cos\_eval
  - GiNaC, [161](#)
- cos\_evalf
  - GiNaC, [161](#)
- cos\_imag\_part
  - GiNaC, [161](#)
- cos\_real\_part
  - GiNaC, [161](#)
- cosh
  - GiNaC, [208](#)
- cosh\_conjugate
  - GiNaC, [169](#)
- cosh\_deriv
  - GiNaC, [169](#)
- cosh\_eval
  - GiNaC, [168](#)
- cosh\_evalf
  - GiNaC, [168](#)
- cosh\_imag\_part
  - GiNaC, [169](#)
- cosh\_real\_part
  - GiNaC, [169](#)
- count
  - GiNaC::archive\_node::property\_info, [953](#)
  - GiNaC::library\_init, [731](#)
- count\_dummy\_indices
  - GiNaC, [119](#)
- count\_factors
  - GiNaC::ncmul, [843](#)
- count\_free\_indices
  - GiNaC, [119](#)
- covariant
  - GiNaC::varidx, [1152](#)
- crational
  - GiNaC::info\_flags, [688](#)
- crational\_polynomial
  - GiNaC::info\_flags, [688](#)
- crc32
  - GiNaC, [101](#)
- crc32.h, [1182](#)
- crctab
  - GiNaC, [255](#)
- csgn
  - GiNaC, [219](#)
  - GiNaC::numeric, [864](#)
- csgn\_conjugate
  - GiNaC, [131](#)
- csgn\_eval
  - GiNaC, [131](#)
- csgn\_evalf
  - GiNaC, [131](#)
- csgn\_imag\_part
  - GiNaC, [132](#)
- csgn\_power
  - GiNaC, [132](#)
- csgn\_real\_part
  - GiNaC, [131](#)
- csgn\_series
  - GiNaC, [131](#)
- csrc
  - GiNaC, [231](#)
- csrc\_cl\_N
  - GiNaC, [232](#)
- csrc\_double
  - GiNaC, [232](#)
- csrc\_float
  - GiNaC, [232](#)
- current\_serial
  - GiNaC::function, [608](#)
- current\_updated
  - GiNaC::composition\_generator, [394](#)
  - GiNaC::partition\_generator, [875](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [878](#)
- current\_vector
  - GiNaC::lanczos\_coeffs, [728](#)
- cyclic
  - GiNaC::symmetry, [1099](#)
- cyclic\_permutation
  - GiNaC, [250](#)
- dbgprint
  - GiNaC::basic, [322](#)
  - GiNaC::ex, [519](#)
- dbgprinttree
  - GiNaC::basic, [323](#)
  - GiNaC::ex, [520](#)
- DCOUT
  - factor.cpp, [1192](#)
- DCOUT2
  - factor.cpp, [1192](#)
- DCOUTVAR
  - factor.cpp, [1192](#)
- debugprint
  - GiNaC::scalar\_products, [1014](#)
  - GiNaC::spmapkey, [1035](#)
- DECLARE\_FUNCTION\_10P
  - function.h, [1212](#)
- DECLARE\_FUNCTION\_11P
  - function.h, [1212](#)
- DECLARE\_FUNCTION\_12P
  - function.h, [1212](#)
- DECLARE\_FUNCTION\_13P
  - function.h, [1213](#)
- DECLARE\_FUNCTION\_14P
  - function.h, [1213](#)
- DECLARE\_FUNCTION\_1P
  - function.h, [1210](#)
- DECLARE\_FUNCTION\_2P
  - function.h, [1210](#)
- DECLARE\_FUNCTION\_3P

- function.h, [1210](#)
- DECLARE\_FUNCTION\_4P
  - function.h, [1210](#)
- DECLARE\_FUNCTION\_5P
  - function.h, [1211](#)
- DECLARE\_FUNCTION\_6P
  - function.h, [1211](#)
- DECLARE\_FUNCTION\_7P
  - function.h, [1211](#)
- DECLARE\_FUNCTION\_8P
  - function.h, [1211](#)
- DECLARE\_FUNCTION\_9P
  - function.h, [1212](#)
- decomp\_rational
  - GiNaC, [190](#)
- DEFAULT\_COMPARE
  - utils.h, [1281](#)
- DEFAULT\_CTOR
  - utils.h, [1281](#)
- default\_overall\_coeff
  - GiNaC::expairseq, [563](#)
  - GiNaC::mul, [790](#)
- DEFAULT\_PRINT
  - utils.h, [1281](#)
- DEFAULT\_PRINT\_LATEX
  - utils.h, [1281](#)
- deg
  - GiNaC::pole\_error, [901](#)
- deg\_a
  - GiNaC::sym\_desc, [1080](#)
- deg\_b
  - GiNaC::sym\_desc, [1080](#)
- degree
  - GiNaC, [104](#)
  - GiNaC::add, [286](#)
  - GiNaC::basic, [327](#)
  - GiNaC::ex, [526](#)
  - GiNaC::integral, [694](#)
  - GiNaC::mul, [783](#)
  - GiNaC::ncmul, [839](#)
  - GiNaC::numeric, [854](#)
  - GiNaC::pole\_error, [900](#)
  - GiNaC::power, [914](#)
  - GiNaC::pseries, [960](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1046](#)
- delete\_cyclic
  - GiNaC::remember\_strategies, [1001](#)
- delete\_lfu
  - GiNaC::remember\_strategies, [1001](#)
- delete\_lru
  - GiNaC::remember\_strategies, [1001](#)
- delete\_never
  - GiNaC::remember\_strategies, [1001](#)
- delta\_indent
  - GiNaC::print\_tree, [951](#)
- delta\_tensor
  - GiNaC, [245](#)
- denom
  - GiNaC, [105](#), [223](#)
  - GiNaC::ex, [530](#)
  - GiNaC::numeric, [870](#)
- derivative
  - GiNaC::add, [290](#)
  - GiNaC::basic, [328](#)
  - GiNaC::constant, [413](#)
  - GiNaC::fderivative, [587](#)
  - GiNaC::function, [605](#)
  - GiNaC::idx, [663](#)
  - GiNaC::indexed, [682](#)
  - GiNaC::integral, [697](#)
  - GiNaC::mul, [787](#)
  - GiNaC::ncmul, [842](#)
  - GiNaC::numeric, [859](#)
  - GiNaC::power, [919](#)
  - GiNaC::pseries, [964](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1048](#)
  - GiNaC::symbol, [1089](#)
- derivative\_f
  - GiNaC::function\_options, [646](#)
- derivative\_func
  - GiNaC::function\_options, [628–630](#), [639](#)
- derivative\_funcp
  - GiNaC, [60](#)
- derivative\_funcp\_1
  - GiNaC, [61](#)
- derivative\_funcp\_10
  - GiNaC, [73](#)
- derivative\_funcp\_11
  - GiNaC, [74](#)
- derivative\_funcp\_12
  - GiNaC, [76](#)
- derivative\_funcp\_13
  - GiNaC, [77](#)
- derivative\_funcp\_14
  - GiNaC, [79](#)
- derivative\_funcp\_2
  - GiNaC, [63](#)
- derivative\_funcp\_3
  - GiNaC, [64](#)
- derivative\_funcp\_4
  - GiNaC, [65](#)
- derivative\_funcp\_5
  - GiNaC, [66](#)
- derivative\_funcp\_6
  - GiNaC, [68](#)
- derivative\_funcp\_7
  - GiNaC, [69](#)
- derivative\_funcp\_8
  - GiNaC, [70](#)
- derivative\_funcp\_9
  - GiNaC, [72](#)
- derivative\_funcp\_exvector
  - GiNaC, [80](#)
- derivative\_map\_function
  - GiNaC::derivative\_map\_function, [438](#)
- derivative\_use\_exvector\_args

- GiNaC::function\_options, [649](#)
- derivatives
  - GiNaC::fderivative, [588](#)
- descend
  - GiNaC::const\_postorder\_iterator, [402](#)
- determinant
  - GiNaC, [184](#)
  - GiNaC::matrix, [749](#)
- determinant\_minor
  - GiNaC::matrix, [753](#)
- dflt
  - GiNaC, [231](#)
- diag\_matrix
  - GiNaC, [181](#)
- diff
  - GiNaC, [107](#)
  - GiNaC::basic, [336](#)
  - GiNaC::ex, [528](#)
- difference\_type
  - GiNaC::const\_iterator, [396](#)
  - GiNaC::const\_postorder\_iterator, [400](#)
  - GiNaC::const\_preorder\_iterator, [403](#)
- Digits
  - GiNaC, [257](#)
- digits
  - GiNaC::\_numeric\_digits, [277](#)
- digits\_changed\_callback
  - GiNaC, [81](#)
- dim
  - GiNaC::idx, [667](#)
  - GiNaC::spmapkey, [1035](#)
- dirac\_gamma
  - GiNaC, [87](#)
- dirac\_gamma5
  - GiNaC, [88](#)
- dirac\_gammaL
  - GiNaC, [88](#)
- dirac\_gammaR
  - GiNaC, [88](#)
- dirac\_ONE
  - GiNaC, [86](#)
- dirac\_slash
  - GiNaC, [89](#)
- dirac\_trace
  - GiNaC, [89](#), [90](#)
- dirichlet\_character
  - GiNaC, [175](#)
- div
  - GiNaC::numeric, [860](#)
- div\_dyn
  - GiNaC::numeric, [862](#)
- divfree
  - GiNaC::determinant\_algo, [439](#)
  - GiNaC::solve\_algo, [1016](#)
- divide
  - GiNaC, [192](#)
- divide\_in\_z
  - GiNaC, [192](#)
- division\_free\_elimination
  - GiNaC::matrix, [754](#)
- do\_not\_evalf\_params
  - GiNaC::function\_options, [643](#)
- do\_print
  - GiNaC::add, [293](#)
  - GiNaC::basic, [339](#)
  - GiNaC::basic\_log\_kernel, [346](#)
  - GiNaC::cliffordunit, [376](#)
  - GiNaC::constant, [414](#)
  - GiNaC::container< C >, [430](#)
  - GiNaC::diracgamma, [445](#)
  - GiNaC::diracgamma5, [450](#)
  - GiNaC::diracgammaL, [455](#)
  - GiNaC::diracgammaR, [460](#)
  - GiNaC::diracone, [465](#)
  - GiNaC::Ebar\_kernel, [474](#)
  - GiNaC::Eisenstein\_h\_kernel, [483](#)
  - GiNaC::Eisenstein\_kernel, [493](#)
  - GiNaC::ELi\_kernel, [503](#)
  - GiNaC::expairseq, [563](#)
  - GiNaC::fail, [575](#)
  - GiNaC::fderivative, [588](#)
  - GiNaC::idx, [666](#)
  - GiNaC::indexed, [685](#)
  - GiNaC::integral, [698](#)
  - GiNaC::integration\_kernel, [707](#)
  - GiNaC::Kronecker\_dtau\_kernel, [718](#)
  - GiNaC::Kronecker\_dz\_kernel, [726](#)
  - GiNaC::matrix, [756](#)
  - GiNaC::minkmetric, [764](#)
  - GiNaC::modular\_form\_kernel, [773](#)
  - GiNaC::mul, [791](#)
  - GiNaC::multiple\_polylog\_kernel, [830](#)
  - GiNaC::ncmul, [842](#)
  - GiNaC::numeric, [871](#)
  - GiNaC::pseries, [968](#)
  - GiNaC::relational, [998](#)
  - GiNaC::spinidx, [1026](#)
  - GiNaC::spinmetric, [1033](#)
  - GiNaC::su3d, [1059](#)
  - GiNaC::su3f, [1065](#)
  - GiNaC::su3one, [1069](#)
  - GiNaC::su3t, [1075](#)
  - GiNaC::symbol, [1091](#)
  - GiNaC::symmetry, [1102](#)
  - GiNaC::tensdelta, [1112](#)
  - GiNaC::tensepsilon, [1119](#)
  - GiNaC::tensmetric, [1125](#)
  - GiNaC::user\_defined\_kernel, [1143](#)
  - GiNaC::varidx, [1151](#)
  - GiNaC::wildcard, [1159](#)
- do\_print\_csrc
  - GiNaC::add, [293](#)
  - GiNaC::fderivative, [588](#)
  - GiNaC::idx, [667](#)
  - GiNaC::mul, [792](#)
  - GiNaC::ncmul, [842](#)

- GiNaC::numeric, [871](#)
  - GiNaC::power, [920](#)
- do\_print\_csrc\_cl\_N
  - GiNaC::numeric, [872](#)
  - GiNaC::power, [921](#)
- do\_print\_dflt
  - GiNaC::clifford, [369](#)
  - GiNaC::power, [920](#)
- do\_print\_latex
  - GiNaC::add, [293](#)
  - GiNaC::clifford, [370](#)
  - GiNaC::cliffordunit, [376](#)
  - GiNaC::constant, [414](#)
  - GiNaC::diracgamma, [445](#)
  - GiNaC::diracgamma5, [450](#)
  - GiNaC::diracgammaL, [455](#)
  - GiNaC::diracgammaR, [460](#)
  - GiNaC::diracone, [465](#)
  - GiNaC::fderivative, [588](#)
  - GiNaC::idx, [667](#)
  - GiNaC::indexed, [685](#)
  - GiNaC::integral, [698](#)
  - GiNaC::matrix, [756](#)
  - GiNaC::minkmetric, [764](#)
  - GiNaC::mul, [792](#)
  - GiNaC::numeric, [871](#)
  - GiNaC::power, [920](#)
  - GiNaC::pseries, [968](#)
  - GiNaC::spinidx, [1026](#)
  - GiNaC::spinmetric, [1033](#)
  - GiNaC::su3d, [1059](#)
  - GiNaC::su3f, [1065](#)
  - GiNaC::su3one, [1069](#)
  - GiNaC::su3t, [1075](#)
  - GiNaC::symbol, [1091](#)
  - GiNaC::tensdelta, [1112](#)
  - GiNaC::tensepsilon, [1119](#)
- do\_print\_python
  - GiNaC::container< C >, [431](#)
  - GiNaC::power, [921](#)
  - GiNaC::pseries, [969](#)
- do\_print\_python\_repr
  - GiNaC::add, [293](#)
  - GiNaC::basic, [339](#)
  - GiNaC::constant, [414](#)
  - GiNaC::container< C >, [431](#)
  - GiNaC::matrix, [756](#)
  - GiNaC::mul, [792](#)
  - GiNaC::numeric, [872](#)
  - GiNaC::power, [921](#)
  - GiNaC::pseries, [969](#)
  - GiNaC::relational, [998](#)
  - GiNaC::symbol, [1091](#)
  - GiNaC::wildcard, [1159](#)
- do\_print\_tree
  - GiNaC::basic, [339](#)
  - GiNaC::clifford, [370](#)
  - GiNaC::constant, [414](#)
- GiNaC::container< C >, [430](#)
  - GiNaC::expairseq, [564](#)
  - GiNaC::fderivative, [589](#)
  - GiNaC::idx, [667](#)
  - GiNaC::indexed, [686](#)
  - GiNaC::numeric, [872](#)
  - GiNaC::pseries, [969](#)
  - GiNaC::spinidx, [1026](#)
  - GiNaC::symbol, [1091](#)
  - GiNaC::symmetry, [1102](#)
  - GiNaC::varidx, [1152](#)
  - GiNaC::wildcard, [1159](#)
- do\_renaming
  - GiNaC::make\_flat\_inserter, [733](#)
- domain
  - GiNaC::constant, [416](#)
- dotted
  - GiNaC::spinidx, [1026](#)
- doublefactorial
  - GiNaC, [212](#)
- dummy
  - GiNaC::function\_options, [616](#)
- dump\_hierarchy
  - GiNaC::class\_info< OPT >, [355](#)
- dump\_tree
  - GiNaC::class\_info< OPT >, [355](#)
- duplicate
  - GiNaC::basic, [320](#)
  - GiNaC::possymbol, [906](#)
  - GiNaC::print\_functor\_impl, [938](#)
  - GiNaC::print\_memfun\_handler< T, C >, [942](#)
  - GiNaC::print\_ptrfun\_handler< T, C >, [945](#)
  - GiNaC::realsymbol, [983](#)
- dynallocate
  - GiNaC, [84](#), [85](#)
- dynallocated
  - GiNaC::status\_flags, [1036](#)
- e
  - GiNaC::archive\_node, [312](#)
  - GiNaC::const\_iterator, [399](#)
  - GiNaC::internal::\_iter\_rep, [274](#)
- Ebar\_kernel
  - GiNaC::Ebar\_kernel, [473](#)
- echelon\_form
  - GiNaC::matrix, [753](#)
- ef
  - GiNaC::constant, [415](#)
- Eisenstein\_h\_kernel
  - GiNaC::Eisenstein\_h\_kernel, [481](#)
- Eisenstein\_kernel
  - GiNaC::Eisenstein\_kernel, [491](#)
- element
  - GiNaC::composition\_generator::coolmulti::element, [495](#)
- ELi\_kernel
  - GiNaC::ELi\_kernel, [502](#)
- EllipticE\_deriv
  - GiNaC, [144](#)

- EllipticE\_eval
  - GiNaC, [144](#)
- EllipticE\_evalf
  - GiNaC, [143](#)
- EllipticE\_print\_latex
  - GiNaC, [144](#)
- EllipticE\_series
  - GiNaC, [144](#)
- EllipticK\_deriv
  - GiNaC, [143](#)
- EllipticK\_eval
  - GiNaC, [143](#)
- EllipticK\_evalf
  - GiNaC, [142](#)
- EllipticK\_print\_latex
  - GiNaC, [143](#)
- EllipticK\_series
  - GiNaC, [143](#)
- end
  - GiNaC::archive\_node::archive\_node\_cit\_range, [313](#)
  - GiNaC::container< C >, [430](#)
  - GiNaC::ex, [517](#)
- ensure\_if\_modifiable
  - GiNaC::basic, [339](#)
- epp
  - GiNaC, [59](#)
- epsilon\_tensor
  - GiNaC, [246](#), [247](#)
- epvector
  - GiNaC, [59](#)
- equal
  - GiNaC::relational, [994](#)
- error
  - GiNaC::error\_and\_integral, [506](#)
- error\_and\_integral
  - GiNaC::error\_and\_integral, [505](#)
- eta\_conjugate
  - GiNaC, [133](#)
- eta\_eval
  - GiNaC, [132](#)
- eta\_evalf
  - GiNaC, [132](#)
- eta\_imag\_part
  - GiNaC, [133](#)
- eta\_real\_part
  - GiNaC, [133](#)
- eta\_series
  - GiNaC, [132](#)
- Euler
  - GiNaC, [255](#)
- EulerEvalf
  - GiNaC, [218](#)
- eval
  - GiNaC, [106](#)
  - GiNaC::add, [287](#)
  - GiNaC::basic, [321](#)
  - GiNaC::ex, [518](#)
  - GiNaC::expairseq, [557](#)
  - GiNaC::fderivative, [585](#)
  - GiNaC::function, [602](#)
  - GiNaC::indexed, [681](#)
  - GiNaC::integral, [694](#)
  - GiNaC::mul, [784](#)
  - GiNaC::ncmul, [840](#)
  - GiNaC::numeric, [855](#)
  - GiNaC::power, [914](#)
  - GiNaC::pseries, [962](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1041](#)
  - GiNaC::symbol, [1086](#)
- eval\_f
  - GiNaC::function\_options, [645](#)
- eval\_func
  - GiNaC::function\_options, [616–618](#), [638](#)
- eval\_funcp
  - GiNaC, [60](#)
- eval\_funcp\_1
  - GiNaC, [61](#)
- eval\_funcp\_10
  - GiNaC, [72](#)
- eval\_funcp\_11
  - GiNaC, [74](#)
- eval\_funcp\_12
  - GiNaC, [75](#)
- eval\_funcp\_13
  - GiNaC, [76](#)
- eval\_funcp\_14
  - GiNaC, [78](#)
- eval\_funcp\_2
  - GiNaC, [62](#)
- eval\_funcp\_3
  - GiNaC, [63](#)
- eval\_funcp\_4
  - GiNaC, [64](#)
- eval\_funcp\_5
  - GiNaC, [66](#)
- eval\_funcp\_6
  - GiNaC, [67](#)
- eval\_funcp\_7
  - GiNaC, [68](#)
- eval\_funcp\_8
  - GiNaC, [70](#)
- eval\_funcp\_9
  - GiNaC, [71](#)
- eval\_funcp\_exvector
  - GiNaC, [79](#)
- eval\_indexed
  - GiNaC::basic, [322](#)
  - GiNaC::matrix, [743](#)
  - GiNaC::minkmetric, [763](#)
  - GiNaC::spinmetric, [1032](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1042](#)
  - GiNaC::su3d, [1059](#)
  - GiNaC::su3f, [1064](#)
  - GiNaC::tensdelta, [1111](#)
  - GiNaC::tensepsilon, [1118](#)

- GiNaC::tensmetric, 1125
- eval\_integ
  - GiNaC, 106
  - GiNaC::basic, 321
  - GiNaC::ex, 519
  - GiNaC::integral, 697
  - GiNaC::pseries, 964
- eval\_ncmul
  - GiNaC::add, 290
  - GiNaC::basic, 322
  - GiNaC::clifford, 366
  - GiNaC::color, 386
  - GiNaC::ex, 518
  - GiNaC::function, 602
  - GiNaC::integral, 695
  - GiNaC::mul, 787
  - GiNaC::power, 919
  - GiNaC::relational, 997
  - GiNaC::structure< T, ComparisonPolicy >, 1042
- eval\_use\_exvector\_args
  - GiNaC::function\_options, 648
- evalchildren
  - GiNaC::expairseq, 566
- evalf
  - GiNaC, 106, 183
  - GiNaC::basic, 321
  - GiNaC::constant, 411
  - GiNaC::ex, 518
  - GiNaC::function, 602
  - GiNaC::idx, 662
  - GiNaC::integral, 694
  - GiNaC::mul, 784
  - GiNaC::numeric, 856
  - GiNaC::power, 915
  - GiNaC::pseries, 962
  - GiNaC::symbol, 1086
- evalf\_f
  - GiNaC::function\_options, 646
- evalf\_func
  - GiNaC::function\_options, 618–620, 639
- evalf\_funcp
  - GiNaC, 60
- evalf\_funcp\_1
  - GiNaC, 61
- evalf\_funcp\_10
  - GiNaC, 72
- evalf\_funcp\_11
  - GiNaC, 74
- evalf\_funcp\_12
  - GiNaC, 75
- evalf\_funcp\_13
  - GiNaC, 77
- evalf\_funcp\_14
  - GiNaC, 78
- evalf\_funcp\_2
  - GiNaC, 62
- evalf\_funcp\_3
  - GiNaC, 63
- evalf\_funcp\_4
  - GiNaC, 64
- evalf\_funcp\_5
  - GiNaC, 66
- evalf\_funcp\_6
  - GiNaC, 67
- evalf\_funcp\_7
  - GiNaC, 68
- evalf\_funcp\_8
  - GiNaC, 70
- evalf\_funcp\_9
  - GiNaC, 71
- evalf\_funcp\_exvector
  - GiNaC, 80
- evalf\_params\_first
  - GiNaC::function\_options, 647
- evalf\_use\_exvector\_args
  - GiNaC::function\_options, 649
- evalffunctype
  - GiNaC, 59
- evalm
  - GiNaC, 106
  - GiNaC::add, 287
  - GiNaC::basic, 321
  - GiNaC::ex, 518
  - GiNaC::matrix, 743
  - GiNaC::mul, 785
  - GiNaC::ncmul, 840
  - GiNaC::power, 915
  - GiNaC::pseries, 964
  - GiNaC::structure< T, ComparisonPolicy >, 1042
- evalpoint
  - factor.cpp, 1197
- evaluate
  - GiNaC::scalar\_products, 1014
- evaluated
  - GiNaC::status\_flags, 1036
- even
  - GiNaC::info\_flags, 688
- ex
  - GiNaC::basic, 340
  - GiNaC::const\_iterator, 399
  - GiNaC::ex, 515, 516
- ex.cpp, 1183
- ex.h, 1183
- ex\_to
  - GiNaC, 111
  - GiNaC::ex, 541
- exadd
  - GiNaC, 223
- excompiler.cpp, 1185
- excompiler.h, 1186
- exhashmap
  - GiNaC, 81
- exmap
  - GiNaC, 58
- exminus
  - GiNaC, 223

- exmul
  - GiNaC, 223
- exp
  - GiNaC, 204
- exp\_conjugate
  - GiNaC, 157
- exp\_deriv
  - GiNaC, 156
- exp\_eval
  - GiNaC, 156
- exp\_evalf
  - GiNaC, 156
- exp\_expand
  - GiNaC, 156
- exp\_imag\_part
  - GiNaC, 157
- exp\_info
  - GiNaC, 157
- exp\_power
  - GiNaC, 157
- exp\_real\_part
  - GiNaC, 157
- expair
  - GiNaC::expair, 546
- expair.cpp, 1187
- expair.h, 1188
- expair\_needs\_further\_processing
  - GiNaC::expairseq, 562
  - GiNaC::mul, 789
- expairseq
  - GiNaC::expairseq, 555, 556
- expairseq.cpp, 1188
- expairseq.h, 1189
- expand
  - GiNaC, 103, 183
  - GiNaC::add, 292
  - GiNaC::basic, 328
  - GiNaC::ex, 527
  - GiNaC::expairseq, 560
  - GiNaC::function, 601
  - GiNaC::indexed, 683
  - GiNaC::integral, 696
  - GiNaC::mul, 790
  - GiNaC::ncmul, 840
  - GiNaC::power, 919
  - GiNaC::pseries, 963
  - GiNaC::structure< T, ComparisonPolicy >, 1046
- expand\_add
  - GiNaC::power, 921
- expand\_add\_2
  - GiNaC::power, 921
- expand\_dummy\_sum
  - GiNaC, 123
- expand\_f
  - GiNaC::function\_options, 646
- expand\_func
  - GiNaC::function\_options, 626–628, 639
- expand\_funcp
  - GiNaC, 60
- expand\_funcp\_1
  - GiNaC, 61
- expand\_funcp\_10
  - GiNaC, 73
- expand\_funcp\_11
  - GiNaC, 74
- expand\_funcp\_12
  - GiNaC, 76
- expand\_funcp\_13
  - GiNaC, 77
- expand\_funcp\_14
  - GiNaC, 79
- expand\_funcp\_2
  - GiNaC, 62
- expand\_funcp\_3
  - GiNaC, 64
- expand\_funcp\_4
  - GiNaC, 65
- expand\_funcp\_5
  - GiNaC, 66
- expand\_funcp\_6
  - GiNaC, 67
- expand\_funcp\_7
  - GiNaC, 69
- expand\_funcp\_8
  - GiNaC, 70
- expand\_funcp\_9
  - GiNaC, 71
- expand\_funcp\_exvector
  - GiNaC, 80
- expand\_function\_args
  - GiNaC::expand\_options, 570
- expand\_indexed
  - GiNaC::expand\_options, 570
- expand\_map\_function
  - GiNaC::expand\_map\_function, 569
- expand\_mul
  - GiNaC::power, 922
- expand\_rename\_idx
  - GiNaC::expand\_options, 570
- expand\_transcendental
  - GiNaC::expand\_options, 570
- expand\_use\_exvector\_args
  - GiNaC::function\_options, 649
- expandchildren
  - GiNaC::expairseq, 566
  - GiNaC::mul, 792
  - GiNaC::ncmul, 843
- expanded
  - GiNaC::info\_flags, 688
  - GiNaC::status\_flags, 1036
- expl\_derivative
  - GiNaC::function, 606
- expl\_derivative\_f
  - GiNaC::function\_options, 647
- expl\_derivative\_func
  - GiNaC::function\_options, 630–632, 639

- expl\_derivative\_funcp
    - GiNaC, [60](#)
  - expl\_derivative\_funcp\_1
    - GiNaC, [61](#)
  - expl\_derivative\_funcp\_10
    - GiNaC, [73](#)
  - expl\_derivative\_funcp\_11
    - GiNaC, [74](#)
  - expl\_derivative\_funcp\_12
    - GiNaC, [76](#)
  - expl\_derivative\_funcp\_13
    - GiNaC, [77](#)
  - expl\_derivative\_funcp\_14
    - GiNaC, [79](#)
  - expl\_derivative\_funcp\_2
    - GiNaC, [63](#)
  - expl\_derivative\_funcp\_3
    - GiNaC, [64](#)
  - expl\_derivative\_funcp\_4
    - GiNaC, [65](#)
  - expl\_derivative\_funcp\_5
    - GiNaC, [66](#)
  - expl\_derivative\_funcp\_6
    - GiNaC, [68](#)
  - expl\_derivative\_funcp\_7
    - GiNaC, [69](#)
  - expl\_derivative\_funcp\_8
    - GiNaC, [70](#)
  - expl\_derivative\_funcp\_9
    - GiNaC, [72](#)
  - expl\_derivative\_funcp\_exvector
    - GiNaC, [80](#)
  - expl\_derivative\_use\_exvector\_args
    - GiNaC::function\_options, [649](#)
  - exponent
    - GiNaC::power, [923](#)
  - exponop
    - GiNaC::pseries, [966](#)
  - exprs
    - GiNaC::archive, [301](#)
  - exprseq
    - GiNaC, [59](#)
    - GiNaC::info\_flags, [688](#)
  - exprseq.cpp, [1190](#)
  - exprseq.h, [1190](#)
  - exprtable
    - GiNaC::archive, [302](#)
  - exset
    - GiNaC, [58](#)
  - exvector
    - GiNaC, [58](#)
  - exvectorvector
    - GiNaC, [81](#)
  - F
    - GiNaC::print\_memfun\_handler< T, C >, [942](#)
    - GiNaC::print\_ptrfun\_handler< T, C >, [945](#)
  - f
    - GiNaC::integral, [699](#)
    - GiNaC::print\_memfun\_handler< T, C >, [943](#)
    - GiNaC::print\_ptrfun\_handler< T, C >, [946](#)
    - GiNaC::user\_defined\_kernel, [1143](#)
  - factor
    - GiNaC, [114](#)
  - factor.cpp, [1191](#)
  - c, [1192](#)
  - cache, [1194](#)
  - cont, [1197](#)
  - DCOUT, [1192](#)
  - DCOUT2, [1192](#)
  - DCOUTVAR, [1192](#)
  - evalpoint, [1197](#)
  - factors, [1194](#)
  - k, [1195](#)
  - last, [1195](#)
  - len, [1195](#)
  - lr, [1194](#)
  - m, [1193](#)
  - modulus, [1198](#)
  - n, [1194](#)
  - one, [1194](#)
  - options, [1198](#)
  - poly, [1196](#)
  - pp, [1197](#)
  - R, [1197](#)
  - r, [1192](#)
  - syms, [1198](#)
  - syms\_wox, [1197](#)
  - unit, [1197](#)
  - USE\_SAME\_DEGREE\_FACTOR, [1192](#)
  - value, [1192](#)
  - vn, [1198](#)
  - vnlst, [1198](#)
  - x, [1196](#)
- factor.h, [1199](#)
- factorial
  - GiNaC, [212](#)
- factorial\_conjugate
  - GiNaC, [136](#)
- factorial\_eval
  - GiNaC, [135](#)
- factorial\_evalf
  - GiNaC, [135](#)
- factorial\_imag\_part
  - GiNaC, [136](#)
- factorial\_print\_dfft\_latex
  - GiNaC, [135](#)
- factorial\_real\_part
  - GiNaC, [136](#)
- factors
  - factor.cpp, [1194](#)
- fail.cpp, [1199](#)
- fail.h, [1200](#)
- FAST\_COMPARE
  - normal.cpp, [1246](#)
- fderivative
  - GiNaC::fderivative, [584](#), [585](#)

- GiNaC::function\_options, 645
- fderivative.cpp, 1200
- fderivative.h, 1201
- fibonacci
  - GiNaC, 213
- find
  - GiNaC, 104
  - GiNaC::class\_info< OPT >, 355
  - GiNaC::ex, 523
  - GiNaC::unarchive\_table\_t, 1135
- find\_bool
  - GiNaC::archive\_node, 307
- find\_common\_factor
  - GiNaC, 200
- find\_dummy\_indices
  - GiNaC, 119
- find\_ex
  - GiNaC::archive\_node, 308
- find\_ex\_by\_loc
  - GiNaC::archive\_node, 309
- find\_ex\_node
  - GiNaC::archive\_node, 309
- find\_factory\_fcn
  - GiNaC, 84
- find\_first
  - GiNaC::archive\_node, 308
- find\_free\_and\_dummy
  - GiNaC, 117, 118
- find\_function
  - GiNaC::function, 607
- find\_last
  - GiNaC::archive\_node, 308
- find\_property\_range
  - GiNaC::archive\_node, 308
- find\_real\_imag
  - GiNaC::mul, 791
- find\_string
  - GiNaC::archive\_node, 307
- find\_unsigned
  - GiNaC::archive\_node, 307
- find\_variant\_indices
  - GiNaC, 120
- finished
  - GiNaC::composition\_generator::coolmulti, 436
- first
  - GiNaC::class\_info< OPT >, 356
- flag\_overflow
  - GiNaC::basic\_multi\_iterator< T >, 351
- flags
  - GiNaC::basic, 340
- flags.h, 1201
- force\_include\_tgamma
  - GiNaC, 256
- force\_include\_zeta1
  - GiNaC, 256
- forget
  - GiNaC::archive, 300
  - GiNaC::archive\_node, 310
- format\_index\_value
  - GiNaC, 251
- frac\_cancel
  - GiNaC, 200
- fraction\_free\_elimination
  - GiNaC::matrix, 754
- fsolve
  - GiNaC, 139
- func\_arg\_info
  - GiNaC, 125
- FUNCP\_1P
  - GiNaC, 59
- FUNCP\_2P
  - GiNaC, 59
- FUNCP\_CUBA
  - GiNaC, 59
- function
  - GiNaC::function, 597–601
  - GiNaC::function\_options, 645
- function.cpp, 1202
- function.h, 1203
  - DECLARE\_FUNCTION\_10P, 1212
  - DECLARE\_FUNCTION\_11P, 1212
  - DECLARE\_FUNCTION\_12P, 1212
  - DECLARE\_FUNCTION\_13P, 1213
  - DECLARE\_FUNCTION\_14P, 1213
  - DECLARE\_FUNCTION\_1P, 1210
  - DECLARE\_FUNCTION\_2P, 1210
  - DECLARE\_FUNCTION\_3P, 1210
  - DECLARE\_FUNCTION\_4P, 1210
  - DECLARE\_FUNCTION\_5P, 1211
  - DECLARE\_FUNCTION\_6P, 1211
  - DECLARE\_FUNCTION\_7P, 1211
  - DECLARE\_FUNCTION\_8P, 1211
  - DECLARE\_FUNCTION\_9P, 1212
  - is\_ex\_the\_function, 1213
  - REGISTER\_FUNCTION, 1213
- function\_options
  - GiNaC::function\_options, 615
- functions\_with\_same\_name
  - GiNaC::function\_options, 650
- G
  - GiNaC, 140
- G2\_eval
  - GiNaC, 150
- G2\_evalf
  - GiNaC, 150
- G3\_eval
  - GiNaC, 151
- G3\_evalf
  - GiNaC, 151
- gauss
  - GiNaC::determinant\_algo, 439
  - GiNaC::solve\_algo, 1016
- gauss\_elimination
  - GiNaC::matrix, 753
- gcd
  - GiNaC, 196, 217

- gcd\_pf\_mul
  - GiNaC, 195
- gcd\_pf\_pow
  - GiNaC, 195
- gcd\_pf\_pow\_pow
  - GiNaC, 196
- generalised\_Bernoulli\_number
  - GiNaC, 176
- get
  - GiNaC::composition\_generator, 393
  - GiNaC::partition\_generator, 875
  - GiNaC::partition\_with\_zero\_parts\_generator, 877
- get\_all\_dummy\_indices
  - GiNaC, 122
- get\_all\_dummy\_indices\_safely
  - GiNaC, 122
- get\_cache\_size
  - GiNaC::integration\_kernel, 706
- get\_class\_name
  - GiNaC::structure< T, ComparisonPolicy >, 1041
- get\_clifford\_comp
  - GiNaC, 93
- get\_close\_delim
  - GiNaC::container< C >, 423
- get\_commutator\_sign
  - GiNaC::clifford, 368
- get\_default\_flags
  - GiNaC::container< C >, 423
- get\_default\_TeX\_name
  - GiNaC, 236
- get\_dim
  - GiNaC::idx, 665
- get\_dim\_uint
  - GiNaC, 87
- get\_domain
  - GiNaC::possymbol, 906
  - GiNaC::realsymbol, 982
  - GiNaC::symbol, 1090
- get\_dummy\_indices
  - GiNaC::indexed, 684
- get\_ex
  - GiNaC::archive\_node, 310
- get\_factors
  - GiNaC::ncmul, 843
- get\_first\_symbol
  - GiNaC, 187
- get\_free\_indices
  - GiNaC::add, 290
  - GiNaC::basic, 331
  - GiNaC::ex, 535
  - GiNaC::indexed, 682
  - GiNaC::integral, 696
  - GiNaC::mul, 787
  - GiNaC::ncmul, 841
  - GiNaC::structure< T, ComparisonPolicy >, 1050
- get\_id
  - GiNaC::print\_context\_options, 926
  - GiNaC::registered\_class\_options, 987
- get\_indices
  - GiNaC::indexed, 684
- get\_label
  - GiNaC::wildcard, 1158
- get\_last\_access
  - GiNaC::remember\_table\_entry, 1006
- get\_metric
  - GiNaC::clifford, 368
- get\_name
  - GiNaC::function, 608
  - GiNaC::function\_options, 643
  - GiNaC::print\_context\_options, 925
  - GiNaC::registered\_class\_options, 987
  - GiNaC::symbol, 1091
- get\_node
  - GiNaC::archive, 299
- get\_nparams
  - GiNaC::function\_options, 644
- get\_numerical\_value
  - GiNaC::Ebar\_kernel, 473
  - GiNaC::Eisenstein\_h\_kernel, 482
  - GiNaC::Eisenstein\_kernel, 492
  - GiNaC::ELi\_kernel, 502
  - GiNaC::integration\_kernel, 705
  - GiNaC::Kronecker\_dtau\_kernel, 717
  - GiNaC::Kronecker\_dz\_kernel, 725
  - GiNaC::modular\_form\_kernel, 772
- get\_numerical\_value\_impl
  - GiNaC::integration\_kernel, 707
- get\_open\_delim
  - GiNaC::container< C >, 423
- get\_order
  - GiNaC::lanczos\_coeffs, 728
- get\_parent
  - GiNaC::class\_info< OPT >, 355
- get\_parent\_name
  - GiNaC::print\_context\_options, 925
  - GiNaC::registered\_class\_options, 987
- get\_point
  - GiNaC::pseries, 965
- get\_pointer
  - GiNaC::ptr< T >, 975
- get\_print\_context
  - GiNaC, 230
- get\_print\_dispatch\_table
  - GiNaC::registered\_class\_options, 987
- get\_print\_options
  - GiNaC, 230
- get\_properties
  - GiNaC::archive\_node, 309
- get\_refcount
  - GiNaC::refcounted, 985
- get\_registered\_functions
  - GiNaC::function, 608
- get\_representation\_label
  - GiNaC, 89, 99
  - GiNaC::clifford, 368
  - GiNaC::color, 387

get\_result  
     GiNaC::remember\_table\_entry, 1006  
 get\_serial  
     GiNaC::function, 608  
 get\_series\_coeff  
     GiNaC::integration\_kernel, 706  
 get\_sign  
     GiNaC::multi\_iterator\_permutation< T >, 815  
 get\_struct  
     GiNaC::structure< T, ComparisonPolicy >, 1053  
 get\_successful\_hits  
     GiNaC::remember\_table\_entry, 1006  
 get\_symbol\_stats  
     GiNaC, 188  
 get\_symmetry  
     GiNaC::indexed, 685  
 get\_TeX\_name  
     GiNaC::symbol, 1091  
 get\_top\_node  
     GiNaC::archive, 299  
 get\_type  
     GiNaC::symmetry, 1100  
 get\_value  
     GiNaC::idx, 665  
 get\_var  
     GiNaC::pseries, 965  
 get\_vector  
     GiNaC::basic\_multi\_iterator< T >, 349  
 gethash  
     GiNaC::basic, 338  
     GiNaC::ex, 539  
 GiNaC, 19  
     \_ex0, 264  
     \_ex1, 265  
     \_ex10, 268  
     \_ex11, 269  
     \_ex12, 269  
     \_ex120, 271  
     \_ex15, 269  
     \_ex18, 270  
     \_ex1\_2, 265  
     \_ex1\_3, 265  
     \_ex1\_4, 265  
     \_ex2, 266  
     \_ex20, 270  
     \_ex24, 270  
     \_ex25, 270  
     \_ex3, 266  
     \_ex30, 271  
     \_ex4, 267  
     \_ex48, 271  
     \_ex5, 267  
     \_ex6, 267  
     \_ex60, 271  
     \_ex7, 268  
     \_ex8, 268  
     \_ex9, 268  
     \_ex\_1, 263  
     \_ex\_10, 260  
     \_ex\_11, 260  
     \_ex\_12, 260  
     \_ex\_120, 258  
     \_ex\_15, 260  
     \_ex\_18, 259  
     \_ex\_1\_2, 263  
     \_ex\_1\_3, 263  
     \_ex\_1\_4, 264  
     \_ex\_2, 262  
     \_ex\_20, 259  
     \_ex\_24, 259  
     \_ex\_25, 259  
     \_ex\_3, 262  
     \_ex\_30, 258  
     \_ex\_4, 262  
     \_ex\_48, 258  
     \_ex\_5, 262  
     \_ex\_6, 261  
     \_ex\_60, 258  
     \_ex\_7, 261  
     \_ex\_8, 261  
     \_ex\_9, 261  
     \_num0\_bp, 256  
     \_num0\_p, 264  
     \_num10\_p, 268  
     \_num11\_p, 269  
     \_num120\_p, 271  
     \_num12\_p, 269  
     \_num15\_p, 269  
     \_num18\_p, 269  
     \_num1\_2\_p, 265  
     \_num1\_3\_p, 265  
     \_num1\_4\_p, 264  
     \_num1\_p, 265  
     \_num20\_p, 270  
     \_num24\_p, 270  
     \_num25\_p, 270  
     \_num2\_p, 266  
     \_num30\_p, 270  
     \_num3\_p, 266  
     \_num48\_p, 271  
     \_num4\_p, 267  
     \_num5\_p, 267  
     \_num60\_p, 271  
     \_num6\_p, 267  
     \_num7\_p, 267  
     \_num8\_p, 268  
     \_num9\_p, 268  
     \_num\_10\_p, 260  
     \_num\_11\_p, 260  
     \_num\_120\_p, 257  
     \_num\_12\_p, 260  
     \_num\_15\_p, 259  
     \_num\_18\_p, 259  
     \_num\_1\_2\_p, 263  
     \_num\_1\_3\_p, 263  
     \_num\_1\_4\_p, 264

\_num\_1\_p, 263  
 \_num\_20\_p, 259  
 \_num\_24\_p, 259  
 \_num\_25\_p, 258  
 \_num\_2\_p, 262  
 \_num\_30\_p, 258  
 \_num\_3\_p, 262  
 \_num\_48\_p, 258  
 \_num\_4\_p, 262  
 \_num\_5\_p, 261  
 \_num\_60\_p, 258  
 \_num\_6\_p, 261  
 \_num\_7\_p, 261  
 \_num\_8\_p, 261  
 \_num\_9\_p, 260  
 abs, 214  
 abs\_conjugate, 129  
 abs\_eval, 128  
 abs\_evalf, 128  
 abs\_expand, 128  
 abs\_expl\_derivative, 128  
 abs\_imag\_part, 129  
 abs\_info, 129  
 abs\_power, 129  
 abs\_print\_csrc\_float, 128  
 abs\_print\_latex, 128  
 abs\_real\_part, 129  
 acos, 206  
 acos\_conjugate, 165  
 acos\_deriv, 165  
 acos\_eval, 164  
 acos\_evalf, 164  
 acosh, 208  
 acosh\_conjugate, 172  
 acosh\_deriv, 172  
 acosh\_eval, 172  
 acosh\_evalf, 172  
 adaptivesimpson, 174  
 add\_symbol, 187  
 algebraic\_match\_mul\_with\_mul, 186  
 antisymmetric2, 239  
 antisymmetric3, 239  
 antisymmetric4, 239  
 antisymmetrize, 108, 240, 243  
 archive\_atom, 58  
 archive\_node\_id, 58  
 are\_ex\_trivially\_equal, 101  
 asin, 206  
 asin\_conjugate, 164  
 asin\_deriv, 164  
 asin\_eval, 163  
 asin\_evalf, 163  
 asin\_info, 164  
 asinh, 208  
 asinh\_conjugate, 171  
 asinh\_deriv, 171  
 asinh\_eval, 171  
 asinh\_evalf, 171  
 atan, 206, 207  
 atan2\_deriv, 167  
 atan2\_eval, 166  
 atan2\_evalf, 166  
 atan2\_info, 167  
 atan\_conjugate, 166  
 atan\_deriv, 165  
 atan\_eval, 165  
 atan\_evalf, 165  
 atan\_info, 166  
 atan\_series, 166  
 atanh, 209  
 atanh\_conjugate, 173  
 atanh\_deriv, 173  
 atanh\_eval, 172  
 atanh\_evalf, 172  
 atanh\_series, 173  
 base\_and\_index, 86  
 bernoulli, 213  
 Bernoulli\_polynomial, 176  
 beta\_deriv, 148  
 beta\_eval, 148  
 beta\_evalf, 148  
 beta\_series, 148  
 binomial, 213  
 binomial\_conjugate, 137  
 binomial\_eval, 137  
 binomial\_evalf, 136  
 binomial\_imag\_part, 137  
 binomial\_real\_part, 137  
 binomial\_sym, 136  
 callback\_registered, 82  
 canonicalize, 239  
 canonicalize\_clifford, 90  
 Catalan, 255  
 CatalanEvalf, 218  
 charpoly, 184  
 clifford\_bar, 95  
 clifford\_inverse, 92  
 clifford\_max\_label, 91  
 clifford\_moebius\_map, 93, 94  
 clifford\_norm, 92  
 clifford\_prime, 91  
 clifford\_star, 96  
 clifford\_star\_bar, 90  
 clifford\_to\_lst, 93  
 clifford\_unit, 87  
 coeff, 104  
 coerce, 203  
 coerce< int, cln::cl\_I >, 203  
 coerce< unsigned int, cln::cl\_I >, 203  
 collect, 106  
 collect\_common\_factors, 200  
 collect\_symbols, 188  
 color\_d, 98  
 color\_f, 98  
 color\_h, 99  
 color\_ONE, 97

color\_T, 97  
color\_trace, 99, 100  
cols, 183  
compare\_pointers, 249  
compile\_ex, 111, 112  
conjugate, 103  
conjugate\_conjugate, 124  
conjugate\_eval, 124  
conjugate\_evalf, 124  
conjugate\_expl\_derivative, 124  
conjugate\_funcp, 60  
conjugate\_funcp\_1, 61  
conjugate\_funcp\_10, 72  
conjugate\_funcp\_11, 74  
conjugate\_funcp\_12, 75  
conjugate\_funcp\_13, 77  
conjugate\_funcp\_14, 78  
conjugate\_funcp\_2, 62  
conjugate\_funcp\_3, 63  
conjugate\_funcp\_4, 65  
conjugate\_funcp\_5, 66  
conjugate\_funcp\_6, 67  
conjugate\_funcp\_7, 68  
conjugate\_funcp\_8, 70  
conjugate\_funcp\_9, 71  
conjugate\_funcp\_exvector, 80  
conjugate\_imag\_part, 125  
conjugate\_info, 125  
conjugate\_print\_latex, 124  
conjugate\_real\_part, 124  
conjugateepvector, 114  
convert\_H\_to\_Li, 142  
cos, 205  
cos\_conjugate, 162  
cos\_deriv, 161  
cos\_eval, 161  
cos\_evalf, 161  
cos\_imag\_part, 161  
cos\_real\_part, 161  
cosh, 208  
cosh\_conjugate, 169  
cosh\_deriv, 169  
cosh\_eval, 168  
cosh\_evalf, 168  
cosh\_imag\_part, 169  
cosh\_real\_part, 169  
count\_dummy\_indices, 119  
count\_free\_indices, 119  
crc32, 101  
crctab, 255  
csgn, 219  
csgn\_conjugate, 131  
csgn\_eval, 131  
csgn\_evalf, 131  
csgn\_imag\_part, 132  
csgn\_power, 132  
csgn\_real\_part, 131  
csgn\_series, 131  
csrc, 231  
csrc\_cl\_N, 232  
csrc\_double, 232  
csrc\_float, 232  
cyclic\_permutation, 250  
decomp\_rational, 190  
degree, 104  
delta\_tensor, 245  
denom, 105, 223  
derivative\_funcp, 60  
derivative\_funcp\_1, 61  
derivative\_funcp\_10, 73  
derivative\_funcp\_11, 74  
derivative\_funcp\_12, 76  
derivative\_funcp\_13, 77  
derivative\_funcp\_14, 79  
derivative\_funcp\_2, 63  
derivative\_funcp\_3, 64  
derivative\_funcp\_4, 65  
derivative\_funcp\_5, 66  
derivative\_funcp\_6, 68  
derivative\_funcp\_7, 69  
derivative\_funcp\_8, 70  
derivative\_funcp\_9, 72  
derivative\_funcp\_exvector, 80  
determinant, 184  
dfft, 231  
diag\_matrix, 181  
diff, 107  
Digits, 257  
digits\_changed\_callback, 81  
dirac\_gamma, 87  
dirac\_gamma5, 88  
dirac\_gammaL, 88  
dirac\_gammaR, 88  
dirac\_ONE, 86  
dirac\_slash, 89  
dirac\_trace, 89, 90  
dirichlet\_character, 175  
divide, 192  
divide\_in\_z, 192  
doublefactorial, 212  
dynallocate, 84, 85  
EllipticE\_deriv, 144  
EllipticE\_eval, 144  
EllipticE\_evalf, 143  
EllipticE\_print\_latex, 144  
EllipticE\_series, 144  
EllipticK\_deriv, 143  
EllipticK\_eval, 143  
EllipticK\_evalf, 142  
EllipticK\_print\_latex, 143  
EllipticK\_series, 143  
epp, 59  
epsilon\_tensor, 246, 247  
epvector, 59  
eta\_conjugate, 133  
eta\_eval, 132

eta\_evalf, 132  
eta\_imag\_part, 133  
eta\_real\_part, 133  
eta\_series, 132  
Euler, 255  
EulerEvalf, 218  
eval, 106  
eval\_funcp, 60  
eval\_funcp\_1, 61  
eval\_funcp\_10, 72  
eval\_funcp\_11, 74  
eval\_funcp\_12, 75  
eval\_funcp\_13, 76  
eval\_funcp\_14, 78  
eval\_funcp\_2, 62  
eval\_funcp\_3, 63  
eval\_funcp\_4, 64  
eval\_funcp\_5, 66  
eval\_funcp\_6, 67  
eval\_funcp\_7, 68  
eval\_funcp\_8, 70  
eval\_funcp\_9, 71  
eval\_funcp\_exvector, 79  
eval\_integ, 106  
evalf, 106, 183  
evalf\_funcp, 60  
evalf\_funcp\_1, 61  
evalf\_funcp\_10, 72  
evalf\_funcp\_11, 74  
evalf\_funcp\_12, 75  
evalf\_funcp\_13, 77  
evalf\_funcp\_14, 78  
evalf\_funcp\_2, 62  
evalf\_funcp\_3, 63  
evalf\_funcp\_4, 64  
evalf\_funcp\_5, 66  
evalf\_funcp\_6, 67  
evalf\_funcp\_7, 68  
evalf\_funcp\_8, 70  
evalf\_funcp\_9, 71  
evalf\_funcp\_exvector, 80  
evalffunctype, 59  
evalm, 106  
ex\_to, 111  
exadd, 223  
exhashmap, 81  
exmap, 58  
exminus, 223  
exmul, 223  
exp, 204  
exp\_conjugate, 157  
exp\_deriv, 156  
exp\_eval, 156  
exp\_evalf, 156  
exp\_expand, 156  
exp\_imag\_part, 157  
exp\_info, 157  
exp\_power, 157  
exp\_real\_part, 157  
expand, 103, 183  
expand\_dummy\_sum, 123  
expand\_funcp, 60  
expand\_funcp\_1, 61  
expand\_funcp\_10, 73  
expand\_funcp\_11, 74  
expand\_funcp\_12, 76  
expand\_funcp\_13, 77  
expand\_funcp\_14, 79  
expand\_funcp\_2, 62  
expand\_funcp\_3, 64  
expand\_funcp\_4, 65  
expand\_funcp\_5, 66  
expand\_funcp\_6, 67  
expand\_funcp\_7, 69  
expand\_funcp\_8, 70  
expand\_funcp\_9, 71  
expand\_funcp\_exvector, 80  
expl\_derivative\_funcp, 60  
expl\_derivative\_funcp\_1, 61  
expl\_derivative\_funcp\_10, 73  
expl\_derivative\_funcp\_11, 74  
expl\_derivative\_funcp\_12, 76  
expl\_derivative\_funcp\_13, 77  
expl\_derivative\_funcp\_14, 79  
expl\_derivative\_funcp\_2, 63  
expl\_derivative\_funcp\_3, 64  
expl\_derivative\_funcp\_4, 65  
expl\_derivative\_funcp\_5, 66  
expl\_derivative\_funcp\_6, 68  
expl\_derivative\_funcp\_7, 69  
expl\_derivative\_funcp\_8, 70  
expl\_derivative\_funcp\_9, 72  
expl\_derivative\_funcp\_exvector, 80  
exprseq, 59  
exset, 58  
exvector, 58  
exvectorvector, 81  
factor, 114  
factorial, 212  
factorial\_conjugate, 136  
factorial\_eval, 135  
factorial\_evalf, 135  
factorial\_imag\_part, 136  
factorial\_print\_dflt\_latex, 135  
factorial\_real\_part, 136  
fibonacci, 213  
find, 104  
find\_common\_factor, 200  
find\_dummy\_indices, 119  
find\_factory\_fcn, 84  
find\_free\_and\_dummy, 117, 118  
find\_variant\_indices, 120  
force\_include\_tgamma, 256  
force\_include\_zeta1, 256  
format\_index\_value, 251  
frac\_cancel, 200

- fsolve, 139
- func\_arg\_info, 125
- FUNCP\_1P, 59
- FUNCP\_2P, 59
- FUNCP\_CUBA, 59
- G, 140
- G2\_eval, 150
- G2\_evalf, 150
- G3\_eval, 151
- G3\_evalf, 151
- gcd, 196, 217
- gcd\_pf\_mul, 195
- gcd\_pf\_pow, 195
- gcd\_pf\_pow\_pow, 196
- generalised\_Bernoulli\_number, 176
- get\_all\_dummy\_indices, 122
- get\_all\_dummy\_indices\_safely, 122
- get\_clifford\_comp, 93
- get\_default\_TeX\_name, 236
- get\_dim\_uint, 87
- get\_first\_symbol, 187
- get\_print\_context, 230
- get\_print\_options, 230
- get\_representation\_label, 89, 99
- get\_symbol\_stats, 188
- GINAC\_BIND\_UNARCHIVER, 82, 85, 86, 96, 101, 115–117, 119, 174, 176–179, 181, 186, 187, 202, 233, 234, 236, 237, 244, 254, 256
- GINAC\_DECLARE\_UNARCHIVER, 82, 94, 95, 100, 101, 115, 116, 118, 124, 174, 179, 180, 182, 186, 187, 218, 233, 234, 236, 237, 240, 248, 254
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, 82, 84, 85, 96, 101, 114–116, 119, 173, 176–179, 181, 185, 186, 201, 232, 234–237, 244, 253
- golden\_ratio\_hash, 249
- guess\_precision, 210
- H\_deriv, 154
- H\_eval, 154
- H\_evalf, 153
- H\_print\_latex, 154
- H\_series, 154
- has, 103
- hasindex, 122
- haswild, 254
- heur\_gcd, 195
- heur\_gcd\_z, 194
- hold\_ncmul, 187
- I, 256
- idx, 256
- idx\_symmetrization, 121
- ifactor, 174
- imag, 222
- imag\_part, 103
- imag\_part\_conjugate, 127
- imag\_part\_eval, 127
- imag\_part\_evalf, 126
- imag\_part\_expl\_derivative, 127
- imag\_part\_funcp, 60
- imag\_part\_funcp\_1, 61
- imag\_part\_funcp\_10, 73
- imag\_part\_funcp\_11, 74
- imag\_part\_funcp\_12, 75
- imag\_part\_funcp\_13, 77
- imag\_part\_funcp\_14, 78
- imag\_part\_funcp\_2, 62
- imag\_part\_funcp\_3, 64
- imag\_part\_funcp\_4, 65
- imag\_part\_funcp\_5, 66
- imag\_part\_funcp\_6, 67
- imag\_part\_funcp\_7, 69
- imag\_part\_funcp\_8, 70
- imag\_part\_funcp\_9, 71
- imag\_part\_funcp\_exvector, 80
- imag\_part\_imag\_part, 127
- imag\_part\_print\_latex, 127
- imag\_part\_real\_part, 127
- index0, 237
- index1, 238
- index2, 238
- index3, 238
- index\_dimensions, 232
- indices\_consistent, 120
- info\_funcp, 61
- info\_funcp\_1, 62
- info\_funcp\_10, 73
- info\_funcp\_11, 75
- info\_funcp\_12, 76
- info\_funcp\_13, 78
- info\_funcp\_14, 79
- info\_funcp\_2, 63
- info\_funcp\_3, 64
- info\_funcp\_4, 65
- info\_funcp\_5, 67
- info\_funcp\_6, 68
- info\_funcp\_7, 69
- info\_funcp\_8, 71
- info\_funcp\_9, 72
- info\_funcp\_exvector, 81
- interpolate, 193
- inverse, 184, 219
- iquo, 216
- irem, 215
- is\_a, 84, 110, 234
- is\_cinteger, 221
- is\_clifford\_tinfo, 95
- is\_color\_tinfo, 99
- is\_crational, 221
- is\_dirac\_slash, 86
- is\_discriminant\_of\_quadratic\_number\_field, 174
- is\_dummy\_pair, 117
- is\_even, 220
- is\_exactly\_a, 84, 110
- is\_integer, 220
- is\_negative, 220

[is\\_nonneg\\_integer](#), 220  
[is\\_odd](#), 221  
[is\\_order\\_function](#), 142  
[is\\_polynomial](#), 104  
[is\\_pos\\_integer](#), 220  
[is\\_positive](#), 220  
[is\\_prime](#), 221  
[is\\_rational](#), 221  
[is\\_real](#), 221  
[is\\_terminating](#), 235  
[is\\_the\\_function](#), 116  
[is\\_the\\_function< G\\_SERIAL >](#), 141  
[is\\_the\\_function< iterated\\_integral\\_SERIAL >](#), 142  
[is\\_the\\_function< psi\\_SERIAL >](#), 141  
[is\\_the\\_function< zeta\\_SERIAL >](#), 140  
[is\\_zero](#), 109, 219  
[isqrt](#), 218  
[iterated\\_integral](#), 141, 142  
[iterated\\_integral2\\_eval](#), 145  
[iterated\\_integral2\\_evalf](#), 145  
[iterated\\_integral3\\_eval](#), 145  
[iterated\\_integral3\\_evalf](#), 145  
[iterated\\_integral\\_evalf\\_impl](#), 144  
[kronecker\\_symbol](#), 175  
[latex](#), 231  
[lcm](#), 197, 217  
[lcm\\_of\\_coefficients\\_denominators](#), 188  
[lcmcoeff](#), 188  
[ldegree](#), 104  
[lgamma](#), 210, 211  
[lgamma\\_conjugate](#), 146  
[lgamma\\_deriv](#), 146  
[lgamma\\_eval](#), 146  
[lgamma\\_evalf](#), 145  
[lgamma\\_series](#), 146  
[lhs](#), 109  
[Li2](#), 210  
[Li2\\_](#), 209  
[Li2\\_conjugate](#), 134  
[Li2\\_deriv](#), 134  
[Li2\\_eval](#), 133  
[Li2\\_evalf](#), 133  
[Li2\\_projection](#), 209  
[Li2\\_series](#), 134, 209  
[Li3\\_eval](#), 134  
[Li\\_deriv](#), 152  
[Li\\_eval](#), 151  
[Li\\_evalf](#), 151  
[Li\\_print\\_latex](#), 152  
[Li\\_series](#), 151  
[library\\_initializer](#), 256  
[link\\_ex](#), 112, 113  
[log](#), 204  
[log2](#), 248  
[log\\_conjugate](#), 159  
[log\\_deriv](#), 158  
[log\\_eval](#), 158  
[log\\_evalf](#), 158  
[log\\_expand](#), 159  
[log\\_imag\\_part](#), 159  
[log\\_info](#), 159  
[log\\_real\\_part](#), 158  
[log\\_series](#), 158  
[lookup\\_map](#), 81  
[lorentz\\_eps](#), 247  
[lorentz\\_g](#), 246  
[lsolve](#), 139  
[lst](#), 81  
[lst\\_to\\_clifford](#), 92  
[lst\\_to\\_matrix](#), 181  
[make\\_hash\\_seed](#), 116  
[make\\_real\\_float](#), 201  
[make\\_return\\_type\\_t](#), 235  
[map\\_eval\\_integ](#), 254  
[map\\_evalm](#), 254  
[match](#), 107  
[metric\\_tensor](#), 245  
[minimal\\_dim](#), 118  
[mod](#), 214  
[multinomial\\_coefficient](#), 249  
[multiply\\_lcm](#), 189  
[my\\_ios\\_callback](#), 229  
[my\\_ios\\_index](#), 229  
[next\\_print\\_context\\_id](#), 257  
[no\\_index\\_dimensions](#), 232  
[nops](#), 102, 183  
[normal](#), 105  
[not\\_symmetric](#), 238  
[number\\_of\\_type](#), 120  
[numer](#), 105, 222  
[numer\\_denom](#), 105  
[op](#), 109  
[operator!=](#), 228  
[operator<](#), 229  
[operator<<](#), 83, 102, 218, 230, 251–253  
[operator<=](#), 229  
[operator>](#), 229  
[operator>>](#), 83, 231  
[operator>=](#), 229  
[operator+](#), 223, 224, 226  
[operator++](#), 227, 228  
[operator+=](#), 225  
[operator-](#), 224, 226  
[operator--](#), 227, 228  
[operator-=](#), 225, 226  
[operator/](#), 224, 225  
[operator/=](#), 225, 226  
[operator==](#), 228  
[operator\\*](#), 224  
[operator\\*=](#), 225, 226  
[Order\\_conjugate](#), 138  
[Order\\_eval](#), 137  
[Order\\_expl\\_derivative](#), 138  
[Order\\_imag\\_part](#), 138  
[Order\\_power](#), 138  
[Order\\_real\\_part](#), 138

- Order\_series, 138
- paramset, 59
- permutation\_sign, 250
- permute\_free\_index\_to\_front, 97
- Pi, 255
- PiEvalf, 218
- pow, 219, 233
- power\_funcp, 60
- power\_funcp\_1, 62
- power\_funcp\_10, 73
- power\_funcp\_11, 74
- power\_funcp\_12, 76
- power\_funcp\_13, 77
- power\_funcp\_14, 79
- power\_funcp\_2, 63
- power\_funcp\_3, 64
- power\_funcp\_4, 65
- power\_funcp\_5, 66
- power\_funcp\_6, 68
- power\_funcp\_7, 69
- power\_funcp\_8, 70
- power\_funcp\_9, 72
- power\_funcp\_exvector, 80
- prem, 191
- primitive\_dirichlet\_character, 175
- print\_context\_class\_info, 82
- print\_func< print\_context >, 116
- print\_func< print\_dflt >, 85, 96, 244
- print\_funcp, 61
- print\_funcp\_1, 62
- print\_funcp\_10, 73
- print\_funcp\_11, 75
- print\_funcp\_12, 76
- print\_funcp\_13, 78
- print\_funcp\_14, 79
- print\_funcp\_2, 63
- print\_funcp\_3, 64
- print\_funcp\_4, 65
- print\_funcp\_5, 67
- print\_funcp\_6, 68
- print\_funcp\_7, 69
- print\_funcp\_8, 71
- print\_funcp\_9, 72
- print\_funcp\_exvector, 80
- print\_integer\_csrc, 202
- print\_operator, 236
- print\_real\_cl\_N, 204
- print\_real\_csrc, 203
- print\_real\_number, 202
- print\_sym\_pow, 232
- product\_to\_exvector, 121
- psi, 141, 211
- psi1\_deriv, 149
- psi1\_eval, 149
- psi1\_evalf, 149
- psi1\_series, 149
- psi2\_deriv, 150
- psi2\_eval, 150
- psi2\_evalf, 149
- psi2\_series, 150
- python, 231
- python\_repr, 231
- quo, 189
- rank, 184, 185
- read\_real\_float, 201
- read\_unsigned, 83
- real, 222
- real\_part, 103
- real\_part\_conjugate, 126
- real\_part\_eval, 125
- real\_part\_evalf, 125
- real\_part\_expl\_derivative, 126
- real\_part\_funcp, 60
- real\_part\_funcp\_1, 61
- real\_part\_funcp\_10, 73
- real\_part\_funcp\_11, 74
- real\_part\_funcp\_12, 75
- real\_part\_funcp\_13, 77
- real\_part\_funcp\_14, 78
- real\_part\_funcp\_2, 62
- real\_part\_funcp\_3, 63
- real\_part\_funcp\_4, 65
- real\_part\_funcp\_5, 66
- real\_part\_funcp\_6, 67
- real\_part\_funcp\_7, 69
- real\_part\_funcp\_8, 70
- real\_part\_funcp\_9, 71
- real\_part\_funcp\_exvector, 80
- real\_part\_imag\_part, 126
- real\_part\_print\_latex, 126
- real\_part\_real\_part, 126
- reduced\_matrix, 182
- reeval\_ncmul, 186
- REGISTER\_FUNCTION, 125–127, 129, 131–137, 139, 143, 144, 146, 148, 152–154, 157, 159, 161–169, 171–173
- registered\_class\_info, 82
- rem, 190
- remove\_dirac\_ONE, 91
- rename\_dummy\_indices, 120
- rename\_dummy\_indices\_uniquely, 122, 123
- replace\_with\_symbol, 199
- reposition\_dummy\_indices, 120
- resultant, 201
- rhs, 109
- rotate\_left, 249
- rows, 183
- S\_deriv, 153
- S\_eval, 152
- S\_evalf, 152
- S\_print\_latex, 153
- S\_series, 153
- series, 107
- series\_funcp, 60
- series\_funcp\_1, 62
- series\_funcp\_10, 73

series\_funcp\_11, 75  
 series\_funcp\_12, 76  
 series\_funcp\_13, 78  
 series\_funcp\_14, 79  
 series\_funcp\_2, 63  
 series\_funcp\_3, 64  
 series\_funcp\_4, 65  
 series\_funcp\_5, 67  
 series\_funcp\_6, 68  
 series\_funcp\_7, 69  
 series\_funcp\_8, 71  
 series\_funcp\_9, 72  
 series\_funcp\_exvector, 80  
 series\_to\_poly, 234  
 set\_print\_context, 230  
 set\_print\_func, 235  
 set\_print\_options, 230  
 shaker\_sort, 250  
 simplify\_indexed, 107, 121  
 simplify\_indexed\_product, 121  
 sin, 205  
 sin\_conjugate, 160  
 sin\_deriv, 160  
 sin\_eval, 160  
 sin\_evalf, 159  
 sin\_imag\_part, 160  
 sin\_real\_part, 160  
 sinh, 207  
 sinh\_conjugate, 168  
 sinh\_deriv, 167  
 sinh\_eval, 167  
 sinh\_evalf, 167  
 sinh\_imag\_part, 168  
 sinh\_real\_part, 168  
 smod, 214  
 spinor\_metric, 246  
 spmap, 81  
 sprem, 191  
 sqrfree, 198  
 sqrfree\_parfrac, 198  
 sqrfree\_yun, 197  
 sqrt, 217, 233  
 sr\_gcd, 193  
 step, 219  
 step\_conjugate, 130  
 step\_eval, 130  
 step\_evalf, 130  
 step\_imag\_part, 130  
 step\_real\_part, 130  
 step\_series, 130  
 sub\_matrix, 182  
 subs, 110  
 subsvalue, 173  
 swap, 109, 114  
 sy\_anti, 242  
 sy\_cycl, 242, 243  
 sy\_none, 240, 241  
 sy\_symm, 241, 242  
 sym\_desc\_vec, 81  
 symbolic\_matrix, 182, 185  
 symm, 239  
 symmetric2, 238  
 symmetric3, 238  
 symmetric4, 238  
 symmetrize, 108, 240, 243  
 symmetrize\_cyclic, 108, 240, 243  
 synthesize\_func, 58  
 tan, 205  
 tan\_conjugate, 163  
 tan\_deriv, 162  
 tan\_eval, 162  
 tan\_evalf, 162  
 tan\_imag\_part, 163  
 tan\_real\_part, 162  
 tan\_series, 163  
 tanh, 208  
 tanh\_conjugate, 170  
 tanh\_deriv, 170  
 tanh\_eval, 170  
 tanh\_evalf, 169  
 tanh\_imag\_part, 170  
 tanh\_real\_part, 170  
 tanh\_series, 170  
 tensor, 255  
 tgamma, 211  
 tgamma\_conjugate, 147  
 tgamma\_deriv, 147  
 tgamma\_eval, 147  
 tgamma\_evalf, 147  
 tgamma\_series, 147  
 to\_double, 222  
 to\_int, 222  
 to\_long, 222  
 to\_polynomial, 106  
 to\_rational, 105  
 trace, 184  
 trace\_string, 89  
 transpose, 183  
 tree, 231  
 trig\_info, 160  
 tryfactsubs, 185  
 uintvector, 81  
 unarch\_table\_instance, 254  
 unarchive\_map\_t, 58  
 unit\_matrix, 181, 185  
 unlink\_ex, 113  
 unsignedvector, 81  
 version\_major, 257  
 version\_micro, 257  
 version\_minor, 257  
 wild, 254  
 write\_real\_float, 202  
 write\_unsigned, 83  
 zeta, 140, 210  
 zeta1\_deriv, 155  
 zeta1\_eval, 155

- zeta1\_evalf, 155
- zeta1\_print\_latex, 155
- zeta2\_deriv, 156
- zeta2\_eval, 155
- zeta2\_evalf, 155
- zeta2\_print\_latex, 156
- zetaderiv\_deriv, 135
- zetaderiv\_eval, 135
- ginac.h, 1214
- GiNaC::numeric\_digits, 275
  - \_numeric\_digits, 276
  - add\_callback, 276
  - callbacklist, 277
  - digits, 277
  - operator long, 276
  - operator=, 276
  - print, 276
  - too\_late, 277
- GiNaC::add, 278
  - add, 284, 285
  - coeff, 286
  - combine\_ex\_with\_coeff\_to\_pair, 291
  - combine\_pair\_with\_coeff\_to\_pair, 292
  - conjugate, 289
  - degree, 286
  - derivative, 290
  - do\_print, 293
  - do\_print\_csrc, 293
  - do\_print\_latex, 293
  - do\_print\_python\_repr, 293
  - eval, 287
  - eval\_ncmul, 290
  - evalm, 287
  - expand, 292
  - get\_free\_indices, 290
  - imag\_part, 289
  - info, 285
  - integer\_content, 288
  - is\_polynomial, 286
  - ldegree, 286
  - max\_coefficient, 289
  - mul, 294
  - normal, 288
  - power, 294
  - precedence, 285
  - print\_add, 292
  - real\_part, 289
  - recombine\_pair\_to\_ex, 292
  - return\_type, 290
  - return\_type\_tinfo, 290
  - series, 287
  - smod, 288
  - split\_ex\_to\_pair, 291
  - thisexpairseq, 291
- GiNaC::archive, 294
  - ~archive, 296
  - add\_node, 299
  - archive, 296
  - archive\_ex, 296
  - atomize, 300
  - atoms, 301
  - clear, 299
  - exprs, 301
  - exprtable, 302
  - forget, 300
  - get\_node, 299
  - get\_top\_node, 299
  - inv\_at\_cit, 296
  - inverse\_atoms, 302
  - nodes, 301
  - num\_expressions, 299
  - operator<<, 301
  - operator>>, 301
  - printraw, 300
  - unarchive\_ex, 297
  - unatomize, 300
- GiNaC::archive::archived\_ex, 314
  - archived\_ex, 314
  - name, 315
  - root, 315
- GiNaC::archive\_node, 302
  - a, 311
  - add\_bool, 306
  - add\_ex, 307
  - add\_string, 306
  - add\_unsigned, 306
  - archive\_node, 306
  - archive\_node\_cit, 305
  - e, 312
  - find\_bool, 307
  - find\_ex, 308
  - find\_ex\_by\_loc, 309
  - find\_ex\_node, 309
  - find\_first, 308
  - find\_last, 308
  - find\_property\_range, 308
  - find\_string, 307
  - find\_unsigned, 307
  - forget, 310
  - get\_ex, 310
  - get\_properties, 309
  - has\_ex, 310
  - has\_expression, 311
  - has\_same\_ex\_as, 310
  - operator<<, 311
  - operator>>, 311
  - operator=, 306
  - printraw, 310
  - property\_type, 305
  - propinfovector, 305
  - props, 311
  - PTYPE\_BOOL, 305
  - PTYPE\_NODE, 305
  - PTYPE\_STRING, 305
  - PTYPE\_UNSIGNED, 305
  - unarchive, 309

- GiNaC::archive\_node::archive\_node\_cit\_range, 312
  - begin, 313
  - end, 313
- GiNaC::archive\_node::property, 951
  - name, 952
  - property, 951
  - type, 952
  - value, 952
- GiNaC::archive\_node::property\_info, 952
  - count, 953
  - name, 953
  - property\_info, 953
  - type, 953
- GiNaC::basic, 315
  - ~basic, 320
  - accept, 327
  - add\_indexed, 331
  - archive, 335
  - basic, 320
  - calchash, 334
  - clearflag, 338
  - coeff, 327
  - collect, 328
  - compare, 337
  - compare\_same\_type, 334
  - conjugate, 333
  - contract\_with, 332
  - dbgprint, 322
  - dbgprinttree, 323
  - degree, 327
  - derivative, 328
  - diff, 336
  - do\_print, 339
  - do\_print\_python\_repr, 339
  - do\_print\_tree, 339
  - duplicate, 320
  - ensure\_if\_modifiable, 339
  - eval, 321
  - eval\_indexed, 322
  - eval\_integ, 321
  - eval\_ncmul, 322
  - evalf, 321
  - evalm, 321
  - ex, 340
  - expand, 328
  - flags, 340
  - get\_free\_indices, 331
  - gethash, 338
  - has, 325
  - hashvalue, 340
  - hold, 337
  - imag\_part, 333
  - info, 323
  - integer\_content, 330
  - is\_equal, 337
  - is\_equal\_same\_type, 334
  - is\_polynomial, 327
  - ldegree, 327
  - let\_op, 324
  - map, 326
  - match, 325
  - match\_same\_type, 326
  - max\_coefficient, 331
  - nops, 323
  - normal, 329
  - op, 324
  - operator=, 320
  - operator[], 324, 325
  - precedence, 323
  - print, 322
  - print\_dispatch, 334, 335
  - read\_archive, 335
  - real\_part, 333
  - return\_type, 333
  - return\_type\_tinfo, 333
  - scalar\_mul\_indexed, 332
  - series, 329
  - setflag, 338
  - smod, 330
  - subs, 326
  - subs\_one\_level, 336
  - to\_polynomial, 330
  - to\_rational, 329
- GiNaC::basic\_log\_kernel, 341
  - do\_print, 346
  - series\_coeff\_impl, 346
- GiNaC::basic\_multi\_iterator< T >, 346
  - ~basic\_multi\_iterator, 348
  - B, 351
  - basic\_multi\_iterator, 348
  - flag\_overflow, 351
  - get\_vector, 349
  - init, 350
  - N, 351
  - operator<<, 351
  - operator(), 350
  - operator++, 350
  - operator[], 349
  - overflow, 349
  - size, 349
  - v, 351
- GiNaC::basic\_partition\_generator, 352
  - basic\_partition\_generator, 353
  - mpgen, 353
- GiNaC::basic\_partition\_generator::mpartition2, 773
  - m, 774
  - mpartition2, 774
  - n, 774
  - next\_partition, 774
  - x, 774
- GiNaC::class\_info< OPT >, 353
  - class\_info, 354
  - dump\_hierarchy, 355
  - dump\_tree, 355
  - find, 355
  - first, 356

- get\_parent, 355
  - identify\_parents, 355
  - next, 356
  - options, 356
  - parent, 356
  - parents\_identified, 356
- GiNaC::class\_info< OPT >::tree\_node, 1133
  - add\_child, 1134
  - children, 1134
  - info, 1134
  - tree\_node, 1134
- GiNaC::clifford, 357
  - archive, 366
  - clifford, 365
  - commutator\_sign, 370
  - do\_print\_dflt, 369
  - do\_print\_latex, 370
  - do\_print\_tree, 370
  - eval\_ncmul, 366
  - get\_commutator\_sign, 368
  - get\_metric, 368
  - get\_representation\_label, 368
  - let\_op, 369
  - match\_same\_type, 367
  - metric, 370
  - nops, 368
  - op, 369
  - precedence, 366
  - read\_archive, 366
  - representation\_label, 370
  - return\_type, 367
  - return\_type\_tinfo, 367
  - same\_metric, 368
  - subs, 369
  - thiscontainer, 367
- GiNaC::cliffordunit, 371
  - contract\_with, 376
  - do\_print, 376
  - do\_print\_latex, 376
- GiNaC::color, 376
  - archive, 386
  - color, 385
  - eval\_ncmul, 386
  - get\_representation\_label, 387
  - match\_same\_type, 386
  - read\_archive, 386
  - representation\_label, 388
  - return\_type, 387
  - return\_type\_tinfo, 387
  - thiscontainer, 387
- GiNaC::compare\_all\_equal< T >, 388
  - ~compare\_all\_equal, 389
  - struct\_compare, 389
  - struct\_is\_equal, 389
- GiNaC::compare\_bitwise< T >, 389
  - ~compare\_bitwise, 390
  - struct\_compare, 390
  - struct\_is\_equal, 390
- GiNaC::compare\_std\_less< T >, 390
  - ~compare\_std\_less, 391
  - struct\_compare, 391
  - struct\_is\_equal, 391
- GiNaC::composition\_generator, 391
  - atend, 393
  - cmgen, 393
  - composition, 393
  - composition\_generator, 393
  - current\_updated, 394
  - get, 393
  - next, 393
  - trivial, 393
- GiNaC::composition\_generator::coolmulti, 435
  - ~coolmulti, 435
  - after\_i, 436
  - coolmulti, 435
  - finished, 436
  - head, 436
  - i, 436
  - next\_permutation, 436
- GiNaC::composition\_generator::coolmulti::element, 494
  - ~element, 495
  - element, 495
  - next, 495
  - value, 495
- GiNaC::const\_iterator, 394
  - const\_iterator, 396
  - const\_postorder\_iterator, 399
  - const\_preorder\_iterator, 399
  - difference\_type, 396
  - e, 399
  - ex, 399
  - i, 399
  - iterator\_category, 395
  - operator!=, 398
  - operator<, 398
  - operator<=, 398
  - operator>, 398
  - operator>=, 398
  - operator+, 397, 399
  - operator++, 397
  - operator+=", 397
  - operator-, 398, 399
  - operator->, 396
  - operator--, 397
  - operator-=, 397
  - operator==, 398
  - operator[], 396
  - operator\*, 396
  - pointer, 396
  - reference, 396
  - value\_type, 395
- GiNaC::const\_postorder\_iterator, 400
  - const\_postorder\_iterator, 401
  - descend, 402
  - difference\_type, 400
  - increment, 402

- iterator\_category, 400
  - operator!=, 402
  - operator++, 401
  - operator->, 401
  - operator==, 402
  - operator\*, 401
  - pointer, 400
  - reference, 401
  - s, 402
  - value\_type, 400
- GiNaC::const\_preorder\_iterator, 403
  - const\_preorder\_iterator, 404
  - difference\_type, 403
  - increment, 405
  - iterator\_category, 403
  - operator!=, 405
  - operator++, 404
  - operator->, 404
  - operator==, 405
  - operator\*, 404
  - pointer, 403
  - reference, 404
  - s, 405
  - value\_type, 403
- GiNaC::constant, 406
  - archive, 412
  - calchash, 414
  - conjugate, 412
  - constant, 411
  - derivative, 413
  - do\_print, 414
  - do\_print\_latex, 414
  - do\_print\_python\_repr, 414
  - do\_print\_tree, 414
  - domain, 416
  - ef, 415
  - evalf, 411
  - imag\_part, 412
  - info, 411
  - is\_equal\_same\_type, 413
  - is\_polynomial, 412
  - name, 415
  - next\_serial, 415
  - number, 415
  - read\_archive, 413
  - real\_part, 412
  - serial, 415
  - TeX\_name, 415
- GiNaC::container< C >, 416
  - append, 428
  - archive, 425
  - begin, 429
  - conjugate, 426
  - const\_iterator, 422
  - const\_reverse\_iterator, 422
  - container, 422, 423
  - do\_print, 430
  - do\_print\_python, 431
  - do\_print\_python\_repr, 431
  - do\_print\_tree, 430
  - end, 430
  - get\_close\_delim, 423
  - get\_default\_flags, 423
  - get\_open\_delim, 423
  - imag\_part, 426
  - info, 423
  - is\_equal\_same\_type, 426
  - let\_op, 424
  - nops, 424
  - op, 424
  - precedence, 423
  - prepend, 428
  - printseq, 427
  - rbegin, 430
  - read\_archive, 425
  - real\_part, 426
  - remove\_all, 429
  - remove\_first, 429
  - remove\_last, 429
  - rend, 430
  - sort, 429
  - sort\_, 428
  - STLT, 422
  - subs, 425
  - subchildren, 431
  - thiscontainer, 427
  - unique, 429
  - unique\_, 428, 431
- GiNaC::container\_storage< C >, 432
  - ~container\_storage, 433
  - container\_storage, 433
  - reserve, 434
  - seq, 434
  - STLT, 433
- GiNaC::derivative\_map\_function, 437
  - derivative\_map\_function, 438
  - operator(), 438
  - s, 438
- GiNaC::determinant\_algo, 439
  - automatic, 439
  - bareiss, 439
  - divfree, 439
  - gauss, 439
  - laplace, 439
- GiNaC::diracgamma, 440
  - contract\_with, 445
  - do\_print, 445
  - do\_print\_latex, 445
- GiNaC::diracgamma5, 445
  - conjugate, 450
  - do\_print, 450
  - do\_print\_latex, 450
- GiNaC::diracgammaL, 451
  - conjugate, 455
  - do\_print, 455
  - do\_print\_latex, 455

- GiNaC::diracgammaR, 456
  - conjugate, 460
  - do\_print, 460
  - do\_print\_latex, 460
- GiNaC::diracone, 461
  - do\_print, 465
  - do\_print\_latex, 465
- GiNaC::do\_taylor, 466
- GiNaC::domain, 466
  - complex, 466
  - positive, 466
  - real, 466
- GiNaC::dunno, 467
- GiNaC::Ebar\_kernel, 467
  - do\_print, 474
  - Ebar\_kernel, 473
  - get\_numerical\_value, 473
  - is\_numeric, 473
  - let\_op, 473
  - m, 474
  - n, 474
  - nops, 473
  - op, 473
  - series\_coeff\_impl, 474
  - x, 474
  - y, 475
- GiNaC::Eisenstein\_h\_kernel, 475
  - C\_norm, 484
  - coefficient\_a0, 483
  - coefficient\_an, 483
  - do\_print, 483
  - Eisenstein\_h\_kernel, 481
  - get\_numerical\_value, 482
  - is\_numeric, 482
  - k, 484
  - Laurent\_series, 482
  - let\_op, 481
  - N, 484
  - nops, 481
  - op, 481
  - q\_expansion\_modular\_form, 483
  - r, 484
  - s, 484
  - series, 481
  - uses\_Laurent\_series, 482
- GiNaC::Eisenstein\_kernel, 485
  - a, 494
  - b, 494
  - C\_norm, 494
  - do\_print, 493
  - Eisenstein\_kernel, 491
  - get\_numerical\_value, 492
  - is\_numeric, 492
  - K, 494
  - k, 493
  - Laurent\_series, 492
  - let\_op, 492
  - N, 493
  - nops, 491
  - op, 492
  - q\_expansion\_modular\_form, 493
  - series, 491
  - uses\_Laurent\_series, 493
- GiNaC::ELi\_kernel, 496
  - do\_print, 503
  - ELi\_kernel, 502
  - get\_numerical\_value, 502
  - is\_numeric, 502
  - let\_op, 502
  - m, 503
  - n, 503
  - nops, 502
  - op, 502
  - series\_coeff\_impl, 503
  - x, 503
  - y, 504
- GiNaC::error\_and\_integral, 505
  - error, 506
  - error\_and\_integral, 505
  - integral, 506
- GiNaC::error\_and\_integral\_is\_less, 506
  - operator(), 506
- GiNaC::eval\_integ\_map\_function, 507
  - operator(), 508
- GiNaC::evalf\_map\_function, 508
  - operator(), 509
- GiNaC::evalm\_map\_function, 510
  - operator(), 511
- GiNaC::ex, 511
  - accept, 525
  - antisymmetrize, 537, 538
  - archive\_node, 541
  - are\_ex\_trivially\_equal, 541
  - begin, 517
  - bp, 542
  - coeff, 527
  - collect, 528
  - compare, 536
  - conjugate, 522
  - construct\_from\_basic, 539
  - construct\_from\_double, 540
  - construct\_from\_int, 539
  - construct\_from\_long, 540
  - construct\_from\_longlong, 540
  - construct\_from\_string\_and\_lst, 540
  - construct\_from\_uint, 539
  - construct\_from\_ulong, 540
  - construct\_from\_ulonglong, 540
  - content, 532
  - dbgprint, 519
  - dbgprinttree, 520
  - degree, 526
  - denom, 530
  - diff, 528
  - end, 517
  - eval, 518

- eval\_integ, 519
- eval\_ncmul, 518
- evalf, 518
- evalm, 518
- ex, 515, 516
- ex\_to, 541
- expand, 527
- find, 523
- get\_free\_indices, 535
- gethash, 539
- has, 523
- imag\_part, 523
- info, 520
- integer\_content, 532
- is\_a, 542
- is\_equal, 536
- is\_exactly\_a, 542
- is\_polynomial, 526
- is\_zero, 536
- is\_zero\_matrix, 537
- lcoeff, 527
- ldegree, 526
- let\_op, 522
- lhs, 522
- makewritable, 541
- map, 525
- match, 524
- max\_coefficient, 534
- nops, 520
- normal, 529
- numer, 530
- numer\_denom, 531
- op, 521
- operator[], 521, 522
- postorder\_begin, 518
- postorder\_end, 518
- preorder\_begin, 517
- preorder\_end, 517
- primpart, 533
- print, 519
- real\_part, 523
- return\_type, 538
- return\_type\_tinfo, 538
- rhs, 522
- series, 528
- share, 541
- simplify\_indexed, 535
- smod, 534
- subs, 524, 525
- swap, 517
- symmetrize, 537
- symmetrize\_cyclic, 538
- tcoeff, 527
- to\_polynomial, 530
- to\_rational, 529
- traverse, 526
- traverse\_postorder, 526
- traverse\_preorder, 525
- unit, 531
- unitcontprim, 534
- GiNaC::ex\_base\_is\_less, 543
  - operator(), 543
- GiNaC::ex\_is\_equal, 543
  - operator(), 543
- GiNaC::ex\_is\_less, 543
  - operator(), 544
- GiNaC::ex\_swap, 544
  - operator(), 544
- GiNaC::expair, 545
  - coeff, 548
  - compare, 547
  - conjugate, 547
  - expair, 546
  - is\_canonical\_numeric, 547
  - is\_equal, 546
  - is\_less, 546
  - print, 547
  - rest, 548
  - swap, 547
- GiNaC::expair\_is\_less, 548
  - operator(), 549
- GiNaC::expair\_rest\_is\_less, 549
  - operator(), 549
- GiNaC::expair\_swap, 550
  - operator(), 550
- GiNaC::expairseq, 550
  - archive, 559
  - calchash, 560
  - can\_make\_flat, 563
  - canonicalize, 565
  - combine\_ex\_with\_coeff\_to\_pair, 562
  - combine\_overall\_coeff, 563
  - combine\_pair\_with\_coeff\_to\_pair, 562
  - combine\_same\_terms\_sorted\_seq, 566
  - conjugate, 558
  - construct\_from\_2\_ex, 564
  - construct\_from\_2\_expairseq, 564
  - construct\_from\_epvector, 565
  - construct\_from\_expairseq\_ex, 564
  - construct\_from\_exvector, 564
  - default\_overall\_coeff, 563
  - do\_print, 563
  - do\_print\_tree, 564
  - eval, 557
  - evalchildren, 566
  - expair\_needs\_further\_processing, 562
  - expairseq, 555, 556
  - expand, 560
  - expandchildren, 566
  - info, 556
  - is\_canonical, 566
  - is\_equal\_same\_type, 559
  - make\_flat, 565
  - map, 557
  - match, 558
  - nops, 556

- op, [557](#)
- overall\_coeff, [567](#)
- precedence, [556](#)
- printpair, [561](#)
- printseq, [561](#)
- read\_archive, [559](#)
- recombine\_pair\_to\_ex, [562](#)
- return\_type, [560](#)
- seq, [567](#)
- split\_ex\_to\_pair, [561](#)
- subs, [558](#)
- subchildren, [567](#)
- thisexpairseq, [560](#), [561](#)
- to\_polynomial, [558](#)
- to\_rational, [557](#)
- GiNaC::expand\_map\_function, [568](#)
- expand\_map\_function, [569](#)
- operator(), [570](#)
- options, [570](#)
- GiNaC::expand\_options, [570](#)
- expand\_function\_args, [570](#)
- expand\_indexed, [570](#)
- expand\_rename\_idx, [570](#)
- expand\_transcendental, [570](#)
- GiNaC::factor\_options, [571](#)
- all, [571](#)
- polynomial, [571](#)
- GiNaC::fail, [571](#)
- do\_print, [575](#)
- return\_type, [575](#)
- GiNaC::fderivative, [576](#)
- archive, [586](#)
- derivative, [587](#)
- derivatives, [588](#)
- do\_print, [588](#)
- do\_print\_csrc, [588](#)
- do\_print\_latex, [588](#)
- do\_print\_tree, [589](#)
- eval, [585](#)
- fderivative, [584](#), [585](#)
- is\_equal\_same\_type, [587](#)
- match\_same\_type, [587](#)
- parameter\_set, [589](#)
- print, [585](#)
- read\_archive, [586](#)
- series, [585](#)
- thiscontainer, [586](#)
- GiNaC::function, [589](#)
- archive, [604](#)
- calchash, [602](#)
- conjugate, [603](#)
- current\_serial, [608](#)
- derivative, [605](#)
- eval, [602](#)
- eval\_ncmul, [602](#)
- evalf, [602](#)
- expand, [601](#)
- expl\_derivative, [606](#)
- find\_function, [607](#)
- function, [597–601](#)
- get\_name, [608](#)
- get\_registered\_functions, [608](#)
- get\_serial, [608](#)
- imag\_part, [604](#)
- info, [605](#)
- is\_equal\_same\_type, [605](#)
- lookup\_remember\_table, [607](#)
- match\_same\_type, [605](#)
- pderivative, [606](#)
- power, [607](#)
- precedence, [601](#)
- print, [601](#)
- read\_archive, [604](#)
- real\_part, [604](#)
- register\_new, [607](#)
- registered\_functions, [607](#)
- remember\_table\_entry, [608](#)
- return\_type, [606](#)
- return\_type\_tinfo, [606](#)
- serial, [608](#)
- series, [603](#)
- store\_remember\_table, [607](#)
- thiscontainer, [603](#)
- GiNaC::function\_options, [609](#)
- ~function\_options, [616](#)
- conjugate\_f, [646](#)
- conjugate\_func, [620–622](#), [639](#)
- conjugate\_use\_exvector\_args, [649](#)
- derivative\_f, [646](#)
- derivative\_func, [628–630](#), [639](#)
- derivative\_use\_exvector\_args, [649](#)
- do\_not\_evalf\_params, [643](#)
- dummy, [616](#)
- eval\_f, [645](#)
- eval\_func, [616–618](#), [638](#)
- eval\_use\_exvector\_args, [648](#)
- evalf\_f, [646](#)
- evalf\_func, [618–620](#), [639](#)
- evalf\_params\_first, [647](#)
- evalf\_use\_exvector\_args, [649](#)
- expand\_f, [646](#)
- expand\_func, [626–628](#), [639](#)
- expand\_use\_exvector\_args, [649](#)
- expl\_derivative\_f, [647](#)
- expl\_derivative\_func, [630–632](#), [639](#)
- expl\_derivative\_use\_exvector\_args, [649](#)
- fderivative, [645](#)
- function, [645](#)
- function\_options, [615](#)
- functions\_with\_same\_name, [650](#)
- get\_name, [643](#)
- get\_nparams, [644](#)
- has\_derivative, [644](#)
- has\_power, [644](#)
- imag\_part\_f, [646](#)
- imag\_part\_func, [624–626](#), [639](#)

- imag\_part\_use\_exvector\_args, 649
- info\_f, 647
- info\_func, 636–638, 640
- info\_use\_exvector\_args, 650
- initialize, 616
- latex\_name, 616
- name, 645
- nparams, 645
- overloaded, 643
- power\_f, 647
- power\_func, 632–634, 640
- power\_use\_exvector\_args, 649
- print\_dispatch\_table, 647
- print\_func, 640–642
- print\_use\_exvector\_args, 650
- real\_part\_f, 646
- real\_part\_func, 622–624, 639
- real\_part\_use\_exvector\_args, 649
- remember, 643
- remember\_assoc\_size, 648
- remember\_size, 648
- remember\_strategy, 648
- return\_type, 648
- return\_type\_tinfo, 648
- series\_f, 647
- series\_func, 634–636, 640
- series\_use\_exvector\_args, 650
- set\_name, 616
- set\_print\_func, 644
- set\_return\_type, 643
- set\_symmetry, 643
- symtree, 650
- test\_and\_set\_nparams, 644
- TeX\_name, 645
- use\_remember, 648
- use\_return\_type, 648
- GiNaC::G2\_SERIAL, 650
  - serial, 651
- GiNaC::G3\_SERIAL, 651
  - serial, 652
- GiNaC::gcd\_options, 652
  - no\_heur\_gcd, 652
  - no\_part\_factored, 652
  - use\_sr\_gcd, 652
- GiNaC::gcdheu\_failed, 653
- GiNaC::has\_distance < T >, 653
  - no\_type, 654
  - test, 654
  - value, 654
  - yes\_type, 654
- GiNaC::has\_options, 655
  - algebraic, 655
- GiNaC::idx, 656
  - archive, 663
  - calchash, 664
  - derivative, 663
  - dim, 667
  - do\_print, 666
  - do\_print\_csrc, 667
  - do\_print\_latex, 667
  - do\_print\_tree, 667
  - evalf, 662
  - get\_dim, 665
  - get\_value, 665
  - idx, 661
  - info, 661
  - is\_dim\_numeric, 665
  - is\_dim\_symbolic, 666
  - is\_dummy\_pair\_same\_type, 664
  - is\_numeric, 665
  - is\_symbolic, 665
  - map, 662
  - match\_same\_type, 664
  - minimal\_dim, 666
  - nops, 662
  - op, 662
  - print\_index, 666
  - read\_archive, 663
  - replace\_dim, 666
  - subs, 662
  - value, 667
- GiNaC::idx\_is\_equal\_ignore\_dim, 668
  - operator(), 668
- GiNaC::indexed, 669
  - all\_index\_values\_are, 684
  - archive, 682
  - derivative, 682
  - do\_print, 685
  - do\_print\_latex, 685
  - do\_print\_tree, 686
  - eval, 681
  - expand, 683
  - get\_dummy\_indices, 684
  - get\_free\_indices, 682
  - get\_indices, 684
  - get\_symmetry, 685
  - has\_dummy\_index\_for, 684
  - imag\_part, 681
  - indexed, 676–680
  - info, 681
  - precedence, 681
  - print\_indexed, 685
  - printindices, 685
  - read\_archive, 682
  - real\_part, 681
  - reposition\_dummy\_indices, 686
  - return\_type, 683
  - return\_type\_tinfo, 683
  - simplify\_indexed, 686
  - simplify\_indexed\_product, 686
  - symtree, 687
  - thiscontainer, 683
  - validate, 686
- GiNaC::info\_flags, 687
  - cinteger, 688
  - cinteger\_polynomial, 688

- crational, 688
- crational\_polynomial, 688
- even, 688
- expanded, 688
- exprseq, 688
- has\_indices, 688
- idx, 688
- indefinite, 688
- indexed, 688
- integer, 688
- integer\_polynomial, 688
- list, 688
- negative, 688
- negint, 688
- nonnegative, 688
- nonnegint, 688
- numeric, 688
- odd, 688
- polynomial, 688
- posint, 688
- positive, 688
- prime, 688
- rational, 688
- rational\_function, 688
- rational\_polynomial, 688
- real, 688
- relation, 688
- relation\_equal, 688
- relation\_greater, 688
- relation\_greater\_or\_equal, 688
- relation\_less, 688
- relation\_less\_or\_equal, 688
- relation\_not\_equal, 688
- symbol, 688
- GiNaC::integral, 689
  - a, 699
  - archive, 697
  - b, 699
  - conjugate, 696
  - degree, 694
  - derivative, 697
  - do\_print, 698
  - do\_print\_latex, 698
  - eval, 694
  - eval\_integ, 697
  - eval\_ncmul, 695
  - evalf, 694
  - expand, 696
  - f, 699
  - get\_free\_indices, 696
  - integral, 694
  - ldegree, 695
  - let\_op, 695
  - max\_integration\_level, 699
  - nops, 695
  - op, 695
  - precedence, 694
  - read\_archive, 697
  - relative\_integration\_error, 699
  - return\_type, 696
  - return\_type\_tinfo, 696
  - series, 698
  - x, 699
- GiNaC::integration\_kernel, 700
  - cache\_step\_size, 707
  - do\_print, 707
  - get\_cache\_size, 706
  - get\_numerical\_value, 705
  - get\_numerical\_value\_impl, 707
  - get\_series\_coeff, 706
  - has\_trailing\_zero, 705
  - is\_numeric, 705
  - Laurent\_series, 705
  - series, 705
  - series\_coeff, 707
  - series\_coeff\_impl, 706
  - series\_vec, 707
  - set\_cache\_step, 706
  - uses\_Laurent\_series, 706
- GiNaC::internal, 272
- GiNaC::internal::\_iter\_rep, 273
  - \_iter\_rep, 274
  - e, 274
  - i, 274
  - i\_end, 275
  - operator!=, 274
  - operator==, 274
- GiNaC::is\_not\_a\_clifford, 708
  - operator(), 708
- GiNaC::is\_summation\_idx, 708
  - operator(), 709
- GiNaC::iterated\_integral2\_SERIAL, 709
  - serial, 709
- GiNaC::iterated\_integral3\_SERIAL, 710
  - serial, 710
- GiNaC::Kronecker\_dtau\_kernel, 710
  - C\_norm, 719
  - do\_print, 718
  - get\_numerical\_value, 717
  - is\_numeric, 717
  - K, 718
  - Kronecker\_dtau\_kernel, 717
  - let\_op, 717
  - n, 718
  - nops, 717
  - op, 717
  - series\_coeff\_impl, 718
  - z, 718
- GiNaC::Kronecker\_dz\_kernel, 719
  - C\_norm, 727
  - do\_print, 726
  - get\_numerical\_value, 725
  - is\_numeric, 725
  - K, 727
  - Kronecker\_dz\_kernel, 725
  - let\_op, 725

- n, 726
- nops, 725
- op, 725
- series\_coeff\_impl, 726
- tau, 726
- z\_j, 726
- GiNaC::lanczos\_coeffs, 727
  - calc\_lanczos\_A, 728
  - coeffs, 728
  - current\_vector, 728
  - get\_order, 728
  - lanczos\_coeffs, 727
  - sufficiently\_accurate, 728
- GiNaC::library\_init, 729
  - ~library\_init, 730
  - count, 731
  - init\_unarchivers, 731
  - library\_init, 730
- GiNaC::make\_flat\_inserter, 731
  - combine\_indices, 732
  - do\_renaming, 733
  - handle\_factor, 732
  - make\_flat\_inserter, 732
  - used\_indices, 733
- GiNaC::map\_function, 733
  - ~map\_function, 735
  - argument\_type, 735
  - operator(), 735
  - result\_type, 735
- GiNaC::matrix, 736
  - add, 746
  - add\_indexed, 744
  - archive, 745
  - charpoly, 750
  - col, 756
  - cols, 746
  - conjugate, 744
  - contract\_with, 744
  - determinant, 749
  - determinant\_minor, 753
  - division\_free\_elimination, 754
  - do\_print, 756
  - do\_print\_latex, 756
  - do\_print\_python\_repr, 756
  - echelon\_form, 753
  - eval\_indexed, 743
  - evalm, 743
  - fraction\_free\_elimination, 754
  - gauss\_elimination, 753
  - imag\_part, 745
  - inverse, 751
  - is\_zero\_matrix, 753
  - let\_op, 743
  - m, 757
  - markowitz\_elimination, 755
  - match\_same\_type, 745
  - matrix, 741, 742
  - mul, 747
  - mul\_scalar, 747
  - nops, 742
  - op, 742
  - operator(), 748
  - pivot, 755
  - pow, 748
  - print\_elements, 756
  - rank, 752
  - read\_archive, 745
  - real\_part, 745
  - return\_type, 746
  - row, 756
  - rows, 746
  - scalar\_mul\_indexed, 744
  - set, 749
  - solve, 752
  - sub, 747
  - subs, 743
  - trace, 750
  - transpose, 749
- GiNaC::minkmetric, 757
  - archive, 763
  - do\_print, 764
  - do\_print\_latex, 764
  - eval\_indexed, 763
  - info, 763
  - minkmetric, 763
  - pos\_sig, 765
  - read\_archive, 764
  - return\_type, 764
- GiNaC::modular\_form\_kernel, 765
  - C\_norm, 773
  - do\_print, 773
  - get\_numerical\_value, 772
  - is\_numeric, 771
  - k, 773
  - Laurent\_series, 772
  - let\_op, 771
  - modular\_form\_kernel, 770
  - nops, 771
  - op, 771
  - P, 773
  - q\_expansion\_modular\_form, 772
  - series, 771
  - uses\_Laurent\_series, 772
- GiNaC::mul, 775
  - add, 793
  - algebraic\_subs\_mul, 791
  - can\_be\_further\_expanded, 792
  - can\_make\_flat, 790
  - coeff, 783
  - combine\_ex\_with\_coeff\_to\_pair, 789
  - combine\_overall\_coeff, 790
  - combine\_pair\_with\_coeff\_to\_pair, 789
  - conjugate, 787
  - default\_overall\_coeff, 790
  - degree, 783
  - derivative, 787

- do\_print, 791
- do\_print\_csrc, 792
- do\_print\_latex, 792
- do\_print\_python\_repr, 792
- eval, 784
- eval\_ncmul, 787
- evalf, 784
- evalm, 785
- expair\_needs\_further\_processing, 789
- expand, 790
- expandchildren, 792
- find\_real\_imag, 791
- get\_free\_indices, 787
- has, 784
- imag\_part, 785
- info, 782
- integer\_content, 786
- is\_polynomial, 783
- ldegree, 783
- max\_coefficient, 786
- mul, 781, 782
- ncmul, 793
- normal, 785
- power, 793
- precedence, 782
- print\_overall\_coeff, 791
- real\_part, 785
- recombine\_pair\_to\_ex, 789
- return\_type, 788
- return\_type\_tinfo, 788
- series, 785
- smod, 786
- split\_ex\_to\_pair, 788
- thisexpairseq, 788
- GiNaC::multi\_iterator\_counter< T >, 794
  - init, 796
  - multi\_iterator\_counter, 796
  - operator<<, 797
  - operator++, 796
- GiNaC::multi\_iterator\_counter\_indv< T >, 797
  - init, 800
  - multi\_iterator\_counter\_indv, 799
  - Nv, 801
  - operator<<, 800
  - operator++, 800
- GiNaC::multi\_iterator\_ordered< T >, 801
  - init, 804
  - multi\_iterator\_ordered, 803
  - operator<<, 804
  - operator++, 804
- GiNaC::multi\_iterator\_ordered\_eq< T >, 805
  - init, 807
  - multi\_iterator\_ordered\_eq, 807
  - operator<<, 808
  - operator++, 807
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 808
  - init, 811
  - multi\_iterator\_ordered\_eq\_indv, 810
- Nv, 812
- operator<<, 811
- operator++, 811
- GiNaC::multi\_iterator\_permutation< T >, 812
  - get\_sign, 815
  - init, 815
  - multi\_iterator\_permutation, 814
  - operator<<, 816
  - operator++, 815
- GiNaC::multi\_iterator\_shuffle< T >, 816
  - init, 819
  - multi\_iterator\_shuffle, 818
  - N\_internal, 819
  - operator<<, 819
  - operator++, 819
  - v\_internal, 819
  - v\_orig, 820
- GiNaC::multi\_iterator\_shuffle\_prime< T >, 820
  - init, 823
  - multi\_iterator\_shuffle\_prime, 823
  - operator<<, 823
- GiNaC::multiple\_polylog\_kernel, 824
  - do\_print, 830
  - is\_numeric, 830
  - let\_op, 830
  - multiple\_polylog\_kernel, 829
  - nops, 830
  - op, 830
  - series\_coeff\_impl, 830
  - z, 831
- GiNaC::ncmul, 831
  - append\_factors, 843
  - coeff, 840
  - conjugate, 841
  - count\_factors, 843
  - degree, 839
  - derivative, 842
  - do\_print, 842
  - do\_print\_csrc, 842
  - eval, 840
  - evalm, 840
  - expand, 840
  - expandchildren, 843
  - get\_factors, 843
  - get\_free\_indices, 841
  - hold\_ncmul, 844
  - imag\_part, 841
  - info, 839
  - ldegree, 839
  - ncmul, 838, 839
  - power, 843
  - precedence, 839
  - real\_part, 841
  - reeval\_ncmul, 843
  - return\_type, 842
  - return\_type\_tinfo, 842
  - thiscontainer, 841
- GiNaC::normal\_map\_function, 844

- operator(), 845
- GiNaC::numeric, 846
  - add, 860
  - add\_dyn, 861
  - archive, 858
  - calchash, 859
  - coeff, 855
  - compare, 864
  - conjugate, 858
  - csgn, 864
  - degree, 854
  - denom, 870
  - derivative, 859
  - div, 860
  - div\_dyn, 862
  - do\_print, 871
  - do\_print\_csrc, 871
  - do\_print\_csrc\_cl\_N, 872
  - do\_print\_latex, 871
  - do\_print\_python\_repr, 872
  - do\_print\_tree, 872
  - eval, 855
  - evalf, 856
  - has, 855
  - imag, 870
  - imag\_part, 858
  - info, 854
  - int\_length, 870
  - integer\_content, 857
  - inverse, 863
  - is\_cinteger, 867
  - is\_crational, 867
  - is\_equal, 864
  - is\_equal\_same\_type, 859
  - is\_even, 866
  - is\_integer, 865
  - is\_negative, 865
  - is\_nonneg\_integer, 866
  - is\_odd, 866
  - is\_polynomial, 854
  - is\_pos\_integer, 865
  - is\_positive, 865
  - is\_prime, 866
  - is\_rational, 866
  - is\_real, 867
  - is\_zero, 865
  - ldegree, 855
  - max\_coefficient, 858
  - mul, 860
  - mul\_dyn, 861
  - normal, 856
  - numer, 870
  - numeric, 852, 853
  - operator!=, 867
  - operator<, 867
  - operator<=, 868
  - operator>, 868
  - operator>=, 868
  - operator=, 862, 863
  - operator==, 867
  - power, 861
  - power\_dyn, 862
  - precedence, 854
  - print\_numeric, 871
  - read\_archive, 859
  - real, 869
  - real\_part, 858
  - smod, 857
  - step, 863
  - sub, 860
  - sub\_dyn, 861
  - subs, 856
  - to\_cl\_N, 869
  - to\_double, 869
  - to\_int, 868
  - to\_long, 869
  - to\_polynomial, 857
  - to\_rational, 856
  - value, 872
- GiNaC::op0\_is\_equal, 873
- operator(), 873
- GiNaC::partition\_generator, 873
  - current\_updated, 875
  - get, 875
  - next, 875
  - partition, 875
  - partition\_generator, 875
- GiNaC::partition\_with\_zero\_parts\_generator, 876
  - current\_updated, 878
  - get, 877
  - m, 878
  - next, 877
  - partition, 878
  - partition\_with\_zero\_parts\_generator, 877
- GiNaC::pointer\_to\_map\_function, 878
  - operator(), 880
  - pointer\_to\_map\_function, 880
  - ptr, 880
- GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 880
  - arg1, 882
  - operator(), 882
  - pointer\_to\_map\_function\_1arg, 882
  - ptr, 882
- GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 883
  - arg1, 885
  - arg2, 885
  - operator(), 884
  - pointer\_to\_map\_function\_2args, 884
  - ptr, 885
- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 885
  - arg1, 887
  - arg2, 887
  - arg3, 887
  - operator(), 887

- pointer\_to\_map\_function\_3args, 887
- ptr, 887
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, 888
  - c, 890
  - operator(), 890
  - pointer\_to\_member\_to\_map\_function, 890
  - ptr, 890
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, 891
  - arg1, 893
  - c, 893
  - operator(), 892
  - pointer\_to\_member\_to\_map\_function\_1arg, 892
  - ptr, 893
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 893
  - arg1, 895
  - arg2, 895
  - c, 895
  - operator(), 895
  - pointer\_to\_member\_to\_map\_function\_2args, 895
  - ptr, 895
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 896
  - arg1, 898
  - arg2, 899
  - arg3, 899
  - c, 898
  - operator(), 898
  - pointer\_to\_member\_to\_map\_function\_3args, 898
  - ptr, 898
- GiNaC::pole\_error, 899
  - deg, 901
  - degree, 900
  - pole\_error, 900
- GiNaC::possymbol, 901
  - duplicate, 906
  - get\_domain, 906
  - possymbol, 906
- GiNaC::power, 907
  - archive, 918
  - basis, 923
  - coeff, 914
  - conjugate, 918
  - degree, 914
  - derivative, 919
  - do\_print\_csrc, 920
  - do\_print\_csrc\_cl\_N, 921
  - do\_print\_dflt, 920
  - do\_print\_latex, 920
  - do\_print\_python, 921
  - do\_print\_python\_repr, 921
  - eval, 914
  - eval\_ncmul, 919
  - evalf, 915
  - evalm, 915
  - expand, 919
  - expand\_add, 921
  - expand\_add\_2, 921
  - expand\_mul, 922
  - exponent, 923
  - has, 916
  - imag\_part, 918
  - info, 912
  - is\_polynomial, 913
  - ldegree, 914
  - map, 913
  - mul, 922
  - nops, 913
  - normal, 917
  - op, 913
  - power, 912
  - precedence, 912
  - print\_power, 920
  - read\_archive, 918
  - real\_part, 918
  - return\_type, 919
  - return\_type\_tinfo, 919
  - series, 916
  - subs, 916
  - to\_polynomial, 917
  - to\_rational, 917
- GiNaC::print\_context, 923
  - ~print\_context, 924
  - options, 924
  - print\_context, 924
  - s, 924
- GiNaC::print\_context\_options, 925
  - get\_id, 926
  - get\_name, 925
  - get\_parent\_name, 925
  - id, 926
  - name, 926
  - parent\_name, 926
  - print\_context\_options, 925
- GiNaC::print\_csrc, 927
  - print\_csrc, 928
- GiNaC::print\_csrc\_cl\_N, 928
  - print\_csrc\_cl\_N, 930
- GiNaC::print\_csrc\_double, 930
  - print\_csrc\_double, 932
- GiNaC::print\_csrc\_float, 932
  - print\_csrc\_float, 934
- GiNaC::print\_dflt, 934
  - print\_dflt, 935
- GiNaC::print\_functor, 935
  - impl, 937
  - is\_valid, 937
  - operator(), 937
  - operator=, 937
  - print\_functor, 936
- GiNaC::print\_functor\_impl, 938
  - ~print\_functor\_impl, 938
  - duplicate, 938
  - operator(), 938

- GiNaC::print\_latex, 939
  - print\_latex, 940
- GiNaC::print\_memfun\_handler< T, C >, 941
  - duplicate, 942
  - F, 942
  - f, 943
  - operator(), 942
  - print\_memfun\_handler, 942
- GiNaC::print\_options, 943
  - print\_index\_dimensions, 943
- GiNaC::print\_ptrfun\_handler< T, C >, 943
  - duplicate, 945
  - F, 945
  - f, 946
  - operator(), 945
  - print\_ptrfun\_handler, 945
- GiNaC::print\_python, 946
  - print\_python, 947
- GiNaC::print\_python\_repr, 948
  - print\_python\_repr, 949
- GiNaC::print\_tree, 949
  - delta\_indent, 951
  - print\_tree, 950
- GiNaC::pseries, 954
  - add\_series, 966
  - archive, 964
  - coeff, 961
  - coeffop, 966
  - collect, 961
  - conjugate, 963
  - convert\_to\_poly, 965
  - degree, 960
  - derivative, 964
  - do\_print, 968
  - do\_print\_latex, 968
  - do\_print\_python, 969
  - do\_print\_python\_repr, 969
  - do\_print\_tree, 969
  - eval, 962
  - eval\_integ, 964
  - evalf, 962
  - evalm, 964
  - expand, 963
  - exponop, 966
  - get\_point, 965
  - get\_var, 965
  - imag\_part, 963
  - is\_compatible\_to, 965
  - is\_terminating, 966
  - is\_zero, 966
  - ldegree, 961
  - mul\_const, 967
  - mul\_series, 967
  - nops, 960
  - normal, 962
  - op, 960
  - point, 970
  - power\_const, 968
  - precedence, 960
  - print\_series, 968
  - pseries, 959, 960
  - read\_archive, 964
  - real\_part, 963
  - seq, 969
  - series, 962
  - shift\_exponents, 968
  - subs, 962
  - var, 969
- GiNaC::psi1\_SERIAL, 970
  - serial, 971
- GiNaC::psi2\_SERIAL, 971
  - serial, 971
- GiNaC::ptr< T >, 972
  - ~ptr, 973
  - get\_pointer, 975
  - makewritable, 974
  - operator!=, 975, 976
  - operator<<, 976
  - operator->, 974
  - operator=, 974
  - operator==, 974, 975
  - operator\*, 974
  - p, 976
  - ptr, 973
  - std::less< ptr< T > >, 975
  - swap, 974
- GiNaC::realsymbol, 977
  - conjugate, 982
  - duplicate, 983
  - get\_domain, 982
  - imag\_part, 983
  - real\_part, 982
  - realsymbol, 982
- GiNaC::refcounted, 983
  - add\_reference, 985
  - get\_refcount, 985
  - refcount, 986
  - refcounted, 985
  - remove\_reference, 985
  - set\_refcount, 985
- GiNaC::registered\_class\_options, 986
  - get\_id, 987
  - get\_name, 987
  - get\_parent\_name, 987
  - get\_print\_dispatch\_table, 987
  - name, 988
  - parent\_name, 988
  - print\_dispatch\_table, 988
  - print\_func, 987, 988
  - registered\_class\_options, 987
  - set\_print\_func, 988
  - tinfo\_key, 988
- GiNaC::relational, 989
  - archive, 996
  - calchash, 998
  - canonical, 997

- do\_print, 998
- do\_print\_python\_repr, 998
- equal, 994
- eval\_ncmul, 997
- greater, 994
- greater\_or\_equal, 994
- info, 995
- less, 994
- less\_or\_equal, 994
- lh, 1000
- lhs, 998
- make\_safe\_bool, 999
- map, 996
- match\_same\_type, 997
- nops, 995
- not\_equal, 994
- o, 1000
- op, 995
- operator safe\_bool, 999
- operator!, 999
- operators, 994
- precedence, 995
- read\_archive, 996
- relational, 995
- return\_type, 997
- return\_type\_tinfo, 998
- rh, 1000
- rhs, 999
- safe\_bool, 994
- subs, 996
- GiNaC::relational::safe\_bool\_helper, 1012
- nonnull, 1012
- GiNaC::remember\_strategies, 1000
- delete\_cyclic, 1001
- delete\_lfu, 1001
- delete\_lru, 1001
- delete\_never, 1001
- GiNaC::remember\_table, 1001
- add\_entry, 1003
- clear\_all\_entries, 1003
- init\_table, 1003
- lookup\_entry, 1003
- max\_assoc\_size, 1004
- remember\_strategy, 1004
- remember\_table, 1002
- remember\_tables, 1003
- show\_statistics, 1003
- table\_size, 1004
- GiNaC::remember\_table\_entry, 1005
- access\_counter, 1007
- get\_last\_access, 1006
- get\_result, 1006
- get\_successful\_hits, 1006
- hashvalue, 1007
- is\_equal, 1006
- last\_access, 1007
- remember\_table\_entry, 1006
- result, 1007
- seq, 1007
- successful\_hits, 1007
- GiNaC::remember\_table\_list, 1008
- add\_entry, 1009
- lookup\_entry, 1009
- max\_assoc\_size, 1009
- remember\_strategy, 1009
- remember\_table\_list, 1009
- GiNaC::return\_type\_t, 1010
- operator!=, 1010
- operator<, 1010
- operator==, 1010
- rl, 1011
- tinfo, 1011
- GiNaC::return\_types, 1011
- commutative, 1012
- noncommutative, 1012
- noncommutative\_composite, 1012
- GiNaC::scalar\_products, 1012
- add, 1013
- add\_vectors, 1013
- clear, 1013
- debugprint, 1014
- evaluate, 1014
- is\_defined, 1014
- spm, 1014
- GiNaC::series\_options, 1015
- suppress\_branchcut, 1015
- GiNaC::solve\_algo, 1015
- automatic, 1016
- bareiss, 1016
- divfree, 1016
- gauss, 1016
- markowitz, 1016
- GiNaC::spinidx, 1016
- archive, 1024
- conjugate, 1024
- do\_print, 1026
- do\_print\_latex, 1026
- do\_print\_tree, 1026
- dotted, 1026
- is\_dotted, 1025
- is\_dummy\_pair\_same\_type, 1024
- is\_undotted, 1025
- match\_same\_type, 1024
- read\_archive, 1024
- spinidx, 1023
- toggle\_dot, 1025
- toggle\_variance\_dot, 1025
- GiNaC::spinmetric, 1027
- contract\_with, 1032
- do\_print, 1033
- do\_print\_latex, 1033
- eval\_indexed, 1032
- info, 1032
- GiNaC::spmapkey, 1033
- debugprint, 1035
- dim, 1035

- operator<, 1035
- operator==, 1035
- spmapkey, 1034
- v1, 1035
- v2, 1035
- GiNaC::status\_flags, 1036
- dynallocated, 1036
- evaluated, 1036
- expanded, 1036
- has\_indices, 1036
- has\_no\_indices, 1036
- hash\_calculated, 1036
- is\_negative, 1036
- is\_positive, 1036
- not\_shareable, 1036
- purely\_indefinite, 1036
- GiNaC::structure< T, ComparisonPolicy >, 1037
- add\_indexed, 1050
- calchash, 1053
- coeff, 1046
- collect, 1046
- contract\_with, 1051
- degree, 1046
- derivative, 1048
- eval, 1041
- eval\_indexed, 1042
- eval\_ncmul, 1042
- evalm, 1042
- expand, 1046
- get\_class\_name, 1041
- get\_free\_indices, 1050
- get\_struct, 1053
- has, 1044
- info, 1043
- integer\_content, 1049
- is\_equal\_same\_type, 1052
- ldegree, 1046
- let\_op, 1044
- map, 1045
- match, 1044
- match\_same\_type, 1045
- max\_coefficient, 1050
- nops, 1043
- normal, 1048
- obj, 1054
- op, 1043
- operator->, 1053
- operator[], 1043, 1044
- precedence, 1043
- print, 1042
- return\_type, 1052
- return\_type\_tinfo, 1052
- scalar\_mul\_indexed, 1051
- series, 1048
- smod, 1049
- structure, 1041
- subs, 1045
- to\_polynomial, 1049
- to\_rational, 1049
- GiNaC::su3d, 1054
- contract\_with, 1059
- do\_print, 1059
- do\_print\_latex, 1059
- eval\_indexed, 1059
- return\_type, 1059
- GiNaC::su3f, 1060
- contract\_with, 1064
- do\_print, 1065
- do\_print\_latex, 1065
- eval\_indexed, 1064
- return\_type, 1064
- GiNaC::su3one, 1065
- do\_print, 1069
- do\_print\_latex, 1069
- GiNaC::su3t, 1070
- contract\_with, 1075
- do\_print, 1075
- do\_print\_latex, 1075
- GiNaC::subs\_options, 1075
- algebraic, 1076
- no\_index\_renaming, 1076
- no\_pattern, 1076
- pattern\_is\_not\_product, 1076
- pattern\_is\_product, 1076
- really\_subs\_idx, 1076
- subs\_algebraic, 1076
- subs\_no\_pattern, 1076
- GiNaC::sy\_is\_less, 1076
- operator(), 1076
- sy\_is\_less, 1076
- v, 1077
- GiNaC::sy\_swap, 1077
- operator(), 1077
- swapped, 1078
- sy\_swap, 1077
- v, 1078
- GiNaC::sym\_desc, 1078
- deg\_a, 1080
- deg\_b, 1080
- ldeg\_a, 1080
- ldeg\_b, 1080
- max\_deg, 1080
- max\_lcnops, 1081
- operator<, 1080
- sym, 1080
- sym\_desc, 1079
- GiNaC::symbol, 1081
- archive, 1088
- calchash, 1090
- conjugate, 1088
- derivative, 1089
- do\_print, 1091
- do\_print\_latex, 1091
- do\_print\_python\_repr, 1091
- do\_print\_tree, 1091
- eval, 1086

- evalf, 1086
- get\_domain, 1090
- get\_name, 1091
- get\_TeX\_name, 1091
- imag\_part, 1088
- info, 1086
- is\_equal\_same\_type, 1089
- is\_polynomial, 1088
- name, 1092
- next\_serial, 1092
- normal, 1087
- read\_archive, 1089
- real\_part, 1088
- serial, 1092
- series, 1086
- set\_name, 1090
- set\_TeX\_name, 1090
- subs, 1087
- symbol, 1085
- TeX\_name, 1092
- to\_polynomial, 1088
- to\_rational, 1087
- GiNaC::symbolset, 1093
  - has, 1093
  - insert\_symbols, 1093
  - s, 1094
  - symbolset, 1093
- GiNaC::symmetry, 1094
  - add, 1101
  - antisymmetric, 1099
  - archive, 1100
  - calchash, 1100
  - canonicalize, 1102
  - children, 1103
  - cyclic, 1099
  - do\_print, 1102
  - do\_print\_tree, 1102
  - get\_type, 1100
  - has\_cyclic, 1102
  - has\_nonsymmetric, 1101
  - has\_symmetry, 1101
  - indices, 1103
  - none, 1099
  - read\_archive, 1100
  - set\_type, 1101
  - sy\_is\_less, 1102
  - sy\_swap, 1102
  - symmetric, 1099
  - symmetry, 1099
  - symmetry\_type, 1099
  - type, 1103
  - validate, 1101
- GiNaC::symminfo, 1104
  - coeff, 1105
  - num, 1105
  - orig, 1105
  - symminfo, 1105
  - symmterm, 1105
- GiNaC::symminfo\_is\_less\_by\_orig, 1106
  - operator(), 1106
- GiNaC::symminfo\_is\_less\_by\_symmterm, 1106
  - operator(), 1106
- GiNaC::tensdelta, 1107
  - contract\_with, 1111
  - do\_print, 1112
  - do\_print\_latex, 1112
  - eval\_indexed, 1111
  - info, 1111
  - return\_type, 1112
- GiNaC::tensepsilon, 1113
  - archive, 1118
  - contract\_with, 1118
  - do\_print, 1119
  - do\_print\_latex, 1119
  - eval\_indexed, 1118
  - info, 1118
  - minkowski, 1119
  - pos\_sig, 1119
  - read\_archive, 1118
  - return\_type, 1119
  - tensepsilon, 1117
- GiNaC::tensmetric, 1120
  - contract\_with, 1125
  - do\_print, 1125
  - eval\_indexed, 1125
  - info, 1125
  - return\_type, 1125
- GiNaC::tensor, 1126
  - replace\_contr\_index, 1130
  - return\_type, 1130
- GiNaC::terminfo, 1131
  - orig, 1132
  - symm, 1132
  - terminfo, 1132
- GiNaC::terminfo\_is\_less, 1133
  - operator(), 1133
- GiNaC::unarchive\_table\_t, 1134
  - ~unarchive\_table\_t, 1135
  - find, 1135
  - insert, 1135
  - unarch\_map, 1135
  - unarchive\_table\_t, 1135
  - usecount, 1135
- GiNaC::user\_defined\_kernel, 1136
  - do\_print, 1143
  - f, 1143
  - is\_numeric, 1142
  - Laurent\_series, 1142
  - let\_op, 1142
  - nops, 1142
  - op, 1142
  - user\_defined\_kernel, 1142
  - uses\_Laurent\_series, 1143
  - x, 1143
- GiNaC::varidx, 1144
  - archive, 1150

- covariant, [1152](#)
- do\_print, [1151](#)
- do\_print\_tree, [1152](#)
- is\_contravariant, [1151](#)
- is\_covariant, [1151](#)
- is\_dummy\_pair\_same\_type, [1150](#)
- match\_same\_type, [1150](#)
- read\_archive, [1150](#)
- toggle\_variance, [1151](#)
- varidx, [1149](#)
- GiNaC::visitor, [1152](#)
- ~visitor, [1153](#)
- GiNaC::wildcard, [1153](#)
  - archive, [1158](#)
  - calchash, [1158](#)
  - do\_print, [1159](#)
  - do\_print\_python\_repr, [1159](#)
  - do\_print\_tree, [1159](#)
  - get\_label, [1158](#)
  - label, [1159](#)
  - match, [1158](#)
  - read\_archive, [1158](#)
  - wildcard, [1157](#)
- GiNaC::zeta1\_SERIAL, [1160](#)
  - serial, [1160](#)
- GiNaC::zeta2\_SERIAL, [1160](#)
  - serial, [1161](#)
- GINAC\_ASSERT
  - assertion.h, [1168](#)
- GINAC\_BIND\_UNARCHIVER
  - archive.h, [1167](#)
  - GiNaC, [82](#), [85](#), [86](#), [96](#), [101](#), [115–117](#), [119](#), [174](#), [176–179](#), [181](#), [186](#), [187](#), [202](#), [233](#), [234](#), [236](#), [237](#), [244](#), [254](#), [256](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT
  - print.h, [1261](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE
  - print.h, [1261](#)
- GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON
  - print.h, [1261](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS
  - registrar.h, [1266](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON
  - registrar.h, [1266](#)
- GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS
  - registrar.h, [1266](#)
- GINAC\_DECLARE\_UNARCHIVER
  - archive.h, [1166](#)
  - GiNaC, [82](#), [94](#), [95](#), [100](#), [101](#), [115](#), [116](#), [118](#), [124](#), [174](#), [179](#), [180](#), [182](#), [186](#), [187](#), [218](#), [233](#), [234](#), [236](#), [237](#), [240](#), [248](#), [254](#)
- GINAC\_IMPLEMENT\_PRINT\_CONTEXT
  - print.h, [1262](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS
  - registrar.h, [1267](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT
  - GiNaC, [82](#), [84](#), [85](#), [96](#), [101](#), [114–116](#), [119](#), [173](#), [176–179](#), [181](#), [185](#), [186](#), [201](#), [232](#), [234–237](#), [244](#), [253](#)
- registrar.h, [1267](#)
- GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T
  - registrar.h, [1267](#)
- GINAC\_LT\_AGE
  - version.h, [1284](#)
- GINAC\_LT\_CURRENT
  - version.h, [1284](#)
- GINAC\_LT\_REVISION
  - version.h, [1284](#)
- GINACLIB\_ARCHIVE\_AGE
  - version.h, [1284](#)
- GINACLIB\_ARCHIVE\_VERSION
  - version.h, [1284](#)
- GINACLIB\_MAJOR\_VERSION
  - version.h, [1284](#)
- GINACLIB\_MICRO\_VERSION
  - version.h, [1284](#)
- GINACLIB\_MINOR\_VERSION
  - version.h, [1284](#)
- GINACLIB\_STR
  - version.h, [1284](#)
- GINACLIB\_STR\_HELPER
  - version.h, [1284](#)
- GINACLIB\_VERSION
  - version.h, [1284](#)
- golden\_ratio\_hash
  - GiNaC, [249](#)
- greater
  - GiNaC::relational, [994](#)
- greater\_or\_equal
  - GiNaC::relational, [994](#)
- guess\_precision
  - GiNaC, [210](#)
- H\_deriv
  - GiNaC, [154](#)
- H\_eval
  - GiNaC, [154](#)
- H\_evalf
  - GiNaC, [153](#)
- H\_print\_latex
  - GiNaC, [154](#)
- H\_series
  - GiNaC, [154](#)
- handle\_factor
  - GiNaC::make\_flat\_inserter, [732](#)
- has
  - GiNaC, [103](#)
  - GiNaC::basic, [325](#)
  - GiNaC::ex, [523](#)
  - GiNaC::mul, [784](#)
  - GiNaC::numeric, [855](#)
  - GiNaC::power, [916](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1044](#)
  - GiNaC::symbolset, [1093](#)
- has\_cyclic
  - GiNaC::symmetry, [1102](#)
- has\_derivative

- GiNaC::function\_options, 644
- has\_dummy\_index\_for
  - GiNaC::indexed, 684
- has\_ex
  - GiNaC::archive\_node, 310
- has\_expression
  - GiNaC::archive\_node, 311
- has\_indices
  - GiNaC::info\_flags, 688
  - GiNaC::status\_flags, 1036
- has\_no\_indices
  - GiNaC::status\_flags, 1036
- has\_nonsymmetric
  - GiNaC::symmetry, 1101
- has\_power
  - GiNaC::function\_options, 644
- has\_same\_ex\_as
  - GiNaC::archive\_node, 310
- has\_symmetry
  - GiNaC::symmetry, 1101
- has\_trailing\_zero
  - GiNaC::integration\_kernel, 705
- hash\_calculated
  - GiNaC::status\_flags, 1036
- hash\_map.h, 1215
- hash\_seed.h, 1215
- hashvalue
  - GiNaC::basic, 340
  - GiNaC::remember\_table\_entry, 1007
- hasindex
  - GiNaC, 122
- haswild
  - GiNaC, 254
- head
  - GiNaC::composition\_generator::coolmulti, 436
- heur\_gcd
  - GiNaC, 195
- heur\_gcd\_z
  - GiNaC, 194
- hold
  - GiNaC::basic, 337
- hold\_ncmul
  - GiNaC, 187
  - GiNaC::ncmul, 844
- I
  - GiNaC, 256
- i
  - GiNaC::composition\_generator::coolmulti, 436
  - GiNaC::const\_iterator, 399
  - GiNaC::internal::\_iter\_rep, 274
- i\_end
  - GiNaC::internal::\_iter\_rep, 275
- id
  - GiNaC::print\_context\_options, 926
- identify\_parents
  - GiNaC::class\_info< OPT >, 355
- idx
  - GiNaC, 256
- GiNaC::idx, 661
  - GiNaC::info\_flags, 688
- idx.cpp, 1216
- idx.h, 1217
- idx\_symmetrization
  - GiNaC, 121
- ifactor
  - GiNaC, 174
- imag
  - GiNaC, 222
  - GiNaC::numeric, 870
- imag\_part
  - GiNaC, 103
  - GiNaC::add, 289
  - GiNaC::basic, 333
  - GiNaC::constant, 412
  - GiNaC::container< C >, 426
  - GiNaC::ex, 523
  - GiNaC::function, 604
  - GiNaC::indexed, 681
  - GiNaC::matrix, 745
  - GiNaC::mul, 785
  - GiNaC::ncmul, 841
  - GiNaC::numeric, 858
  - GiNaC::power, 918
  - GiNaC::pseries, 963
  - GiNaC::realsymbol, 983
  - GiNaC::symbol, 1088
- imag\_part\_conjugate
  - GiNaC, 127
- imag\_part\_eval
  - GiNaC, 127
- imag\_part\_evalf
  - GiNaC, 126
- imag\_part\_expl\_derivative
  - GiNaC, 127
- imag\_part\_f
  - GiNaC::function\_options, 646
- imag\_part\_func
  - GiNaC::function\_options, 624–626, 639
- imag\_part\_funcp
  - GiNaC, 60
- imag\_part\_funcp\_1
  - GiNaC, 61
- imag\_part\_funcp\_10
  - GiNaC, 73
- imag\_part\_funcp\_11
  - GiNaC, 74
- imag\_part\_funcp\_12
  - GiNaC, 75
- imag\_part\_funcp\_13
  - GiNaC, 77
- imag\_part\_funcp\_14
  - GiNaC, 78
- imag\_part\_funcp\_2
  - GiNaC, 62
- imag\_part\_funcp\_3
  - GiNaC, 64

- imag\_part\_funcp\_4
  - GiNaC, [65](#)
- imag\_part\_funcp\_5
  - GiNaC, [66](#)
- imag\_part\_funcp\_6
  - GiNaC, [67](#)
- imag\_part\_funcp\_7
  - GiNaC, [69](#)
- imag\_part\_funcp\_8
  - GiNaC, [70](#)
- imag\_part\_funcp\_9
  - GiNaC, [71](#)
- imag\_part\_funcp\_exvector
  - GiNaC, [80](#)
- imag\_part\_imag\_part
  - GiNaC, [127](#)
- imag\_part\_print\_latex
  - GiNaC, [127](#)
- imag\_part\_real\_part
  - GiNaC, [127](#)
- imag\_part\_use\_exvector\_args
  - GiNaC::function\_options, [649](#)
- impl
  - GiNaC::print\_functor, [937](#)
- increment
  - GiNaC::const\_postorder\_iterator, [402](#)
  - GiNaC::const\_preorder\_iterator, [405](#)
- indefinite
  - GiNaC::info\_flags, [688](#)
- index0
  - GiNaC, [237](#)
- index1
  - GiNaC, [238](#)
- index2
  - GiNaC, [238](#)
- index3
  - GiNaC, [238](#)
- index\_dimensions
  - GiNaC, [232](#)
- indexed
  - GiNaC::indexed, [676–680](#)
  - GiNaC::info\_flags, [688](#)
- indexed.cpp, [1218](#)
- indexed.h, [1220](#)
- indices
  - GiNaC::symmetry, [1103](#)
- indices\_consistent
  - GiNaC, [120](#)
- info
  - GiNaC::add, [285](#)
  - GiNaC::basic, [323](#)
  - GiNaC::class\_info< OPT >::tree\_node, [1134](#)
  - GiNaC::constant, [411](#)
  - GiNaC::container< C >, [423](#)
  - GiNaC::ex, [520](#)
  - GiNaC::expairseq, [556](#)
  - GiNaC::function, [605](#)
  - GiNaC::idx, [661](#)
  - GiNaC::indexed, [681](#)
  - GiNaC::minkmetric, [763](#)
  - GiNaC::mul, [782](#)
  - GiNaC::ncmul, [839](#)
  - GiNaC::numeric, [854](#)
  - GiNaC::power, [912](#)
  - GiNaC::relational, [995](#)
  - GiNaC::spinmetric, [1032](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1043](#)
  - GiNaC::symbol, [1086](#)
  - GiNaC::tensdelta, [1111](#)
  - GiNaC::tensepsilon, [1118](#)
  - GiNaC::tensmetric, [1125](#)
- info\_f
  - GiNaC::function\_options, [647](#)
- info\_func
  - GiNaC::function\_options, [636–638](#), [640](#)
- info\_funcp
  - GiNaC, [61](#)
- info\_funcp\_1
  - GiNaC, [62](#)
- info\_funcp\_10
  - GiNaC, [73](#)
- info\_funcp\_11
  - GiNaC, [75](#)
- info\_funcp\_12
  - GiNaC, [76](#)
- info\_funcp\_13
  - GiNaC, [78](#)
- info\_funcp\_14
  - GiNaC, [79](#)
- info\_funcp\_2
  - GiNaC, [63](#)
- info\_funcp\_3
  - GiNaC, [64](#)
- info\_funcp\_4
  - GiNaC, [65](#)
- info\_funcp\_5
  - GiNaC, [67](#)
- info\_funcp\_6
  - GiNaC, [68](#)
- info\_funcp\_7
  - GiNaC, [69](#)
- info\_funcp\_8
  - GiNaC, [71](#)
- info\_funcp\_9
  - GiNaC, [72](#)
- info\_funcp\_exvector
  - GiNaC, [81](#)
- info\_use\_exvector\_args
  - GiNaC::function\_options, [650](#)
- inifcns.cpp, [1221](#)
- inifcns.h, [1224](#)
- inifcns\_elliptic.cpp, [1225](#)
- inifcns\_gamma.cpp, [1227](#)
- inifcns\_nstdsums.cpp, [1228](#)
- inifcns\_trans.cpp, [1230](#)
- init

- GiNaC::basic\_multi\_iterator< T >, 350
- GiNaC::multi\_iterator\_counter< T >, 796
- GiNaC::multi\_iterator\_counter\_indv< T >, 800
- GiNaC::multi\_iterator\_ordered< T >, 804
- GiNaC::multi\_iterator\_ordered\_eq< T >, 807
- GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 811
- GiNaC::multi\_iterator\_permutation< T >, 815
- GiNaC::multi\_iterator\_shuffle< T >, 819
- GiNaC::multi\_iterator\_shuffle\_prime< T >, 823
- init\_table
  - GiNaC::remember\_table, 1003
- init\_unarchivers
  - GiNaC::library\_init, 731
- initialize
  - GiNaC::function\_options, 616
- insert
  - GiNaC::unarchive\_table\_t, 1135
- insert\_symbols
  - GiNaC::symbolset, 1093
- int\_length
  - GiNaC::numeric, 870
- integer
  - GiNaC::info\_flags, 688
- integer\_content
  - GiNaC::add, 288
  - GiNaC::basic, 330
  - GiNaC::ex, 532
  - GiNaC::mul, 786
  - GiNaC::numeric, 857
  - GiNaC::structure< T, ComparisonPolicy >, 1049
- integer\_polynomial
  - GiNaC::info\_flags, 688
- integral
  - GiNaC::error\_and\_integral, 506
  - GiNaC::integral, 694
- integral.cpp, 1233
- integral.h, 1234
- integration\_kernel.cpp, 1234
  - cache\_vec, 1236
  - order, 1236
  - qbar, 1236
  - x, 1236
- integration\_kernel.h, 1236
- interpolate
  - GiNaC, 193
- inv\_at\_cit
  - GiNaC::archive, 296
- inverse
  - GiNaC, 184, 219
  - GiNaC::matrix, 751
  - GiNaC::numeric, 863
- inverse\_atoms
  - GiNaC::archive, 302
- iquo
  - GiNaC, 216
- irem
  - GiNaC, 215
- is\_a
  - GiNaC, 84, 110, 234
  - GiNaC::ex, 542
- is\_canonical
  - GiNaC::expairseq, 566
- is\_canonical\_numeric
  - GiNaC::expair, 547
- is\_cinteger
  - GiNaC, 221
  - GiNaC::numeric, 867
- is\_clifford\_tinfo
  - GiNaC, 95
- is\_color\_tinfo
  - GiNaC, 99
- is\_compatible\_to
  - GiNaC::pseries, 965
- is\_contravariant
  - GiNaC::varidx, 1151
- is\_covariant
  - GiNaC::varidx, 1151
- is\_crational
  - GiNaC, 221
  - GiNaC::numeric, 867
- is\_defined
  - GiNaC::scalar\_products, 1014
- is\_dim\_numeric
  - GiNaC::idx, 665
- is\_dim\_symbolic
  - GiNaC::idx, 666
- is\_dirac\_slash
  - GiNaC, 86
- is\_discriminant\_of\_quadratic\_number\_field
  - GiNaC, 174
- is\_dotted
  - GiNaC::spinidx, 1025
- is\_dummy\_pair
  - GiNaC, 117
- is\_dummy\_pair\_same\_type
  - GiNaC::idx, 664
  - GiNaC::spinidx, 1024
  - GiNaC::varidx, 1150
- is\_equal
  - GiNaC::basic, 337
  - GiNaC::ex, 536
  - GiNaC::expair, 546
  - GiNaC::numeric, 864
  - GiNaC::remember\_table\_entry, 1006
- is\_equal\_same\_type
  - GiNaC::basic, 334
  - GiNaC::constant, 413
  - GiNaC::container< C >, 426
  - GiNaC::expairseq, 559
  - GiNaC::fderivative, 587
  - GiNaC::function, 605
  - GiNaC::numeric, 859
  - GiNaC::structure< T, ComparisonPolicy >, 1052
  - GiNaC::symbol, 1089
- is\_even
  - GiNaC, 220

- GiNaC::numeric, 866
- is\_ex\_the\_function
  - function.h, 1213
- is\_exactly\_a
  - GiNaC, 84, 110
  - GiNaC::ex, 542
- is\_integer
  - GiNaC, 220
  - GiNaC::numeric, 865
- is\_less
  - GiNaC::expair, 546
- is\_negative
  - GiNaC, 220
  - GiNaC::numeric, 865
  - GiNaC::status\_flags, 1036
- is\_nonneg\_integer
  - GiNaC, 220
  - GiNaC::numeric, 866
- is\_numeric
  - GiNaC::Ebar\_kernel, 473
  - GiNaC::Eisenstein\_h\_kernel, 482
  - GiNaC::Eisenstein\_kernel, 492
  - GiNaC::ELi\_kernel, 502
  - GiNaC::idx, 665
  - GiNaC::integration\_kernel, 705
  - GiNaC::Kronecker\_dtau\_kernel, 717
  - GiNaC::Kronecker\_dz\_kernel, 725
  - GiNaC::modular\_form\_kernel, 771
  - GiNaC::multiple\_polylog\_kernel, 830
  - GiNaC::user\_defined\_kernel, 1142
- is\_odd
  - GiNaC, 221
  - GiNaC::numeric, 866
- is\_order\_function
  - GiNaC, 142
- is\_polynomial
  - GiNaC, 104
  - GiNaC::add, 286
  - GiNaC::basic, 327
  - GiNaC::constant, 412
  - GiNaC::ex, 526
  - GiNaC::mul, 783
  - GiNaC::numeric, 854
  - GiNaC::power, 913
  - GiNaC::symbol, 1088
- is\_pos\_integer
  - GiNaC, 220
  - GiNaC::numeric, 865
- is\_positive
  - GiNaC, 220
  - GiNaC::numeric, 865
  - GiNaC::status\_flags, 1036
- is\_prime
  - GiNaC, 221
  - GiNaC::numeric, 866
- is\_rational
  - GiNaC, 221
  - GiNaC::numeric, 866
- is\_real
  - GiNaC, 221
  - GiNaC::numeric, 867
- is\_symbolic
  - GiNaC::idx, 665
- is\_terminating
  - GiNaC, 235
  - GiNaC::pseries, 966
- is\_the\_function
  - GiNaC, 116
- is\_the\_function< G\_SERIAL >
  - GiNaC, 141
- is\_the\_function< iterated\_integral\_SERIAL >
  - GiNaC, 142
- is\_the\_function< psi\_SERIAL >
  - GiNaC, 141
- is\_the\_function< zeta\_SERIAL >
  - GiNaC, 140
- is\_undotted
  - GiNaC::spinidx, 1025
- is\_valid
  - GiNaC::print\_functor, 937
- is\_zero
  - GiNaC, 109, 219
  - GiNaC::ex, 536
  - GiNaC::numeric, 865
  - GiNaC::pseries, 966
- is\_zero\_matrix
  - GiNaC::ex, 537
  - GiNaC::matrix, 753
- isqrt
  - GiNaC, 218
- iterated\_integral
  - GiNaC, 141, 142
- iterated\_integral2\_eval
  - GiNaC, 145
- iterated\_integral2\_evalf
  - GiNaC, 145
- iterated\_integral3\_eval
  - GiNaC, 145
- iterated\_integral3\_evalf
  - GiNaC, 145
- iterated\_integral\_evalf\_impl
  - GiNaC, 144
- iterator\_category
  - GiNaC::const\_iterator, 395
  - GiNaC::const\_postorder\_iterator, 400
  - GiNaC::const\_preorder\_iterator, 403
- K
  - GiNaC::Eisenstein\_kernel, 494
  - GiNaC::Kronecker\_dtau\_kernel, 718
  - GiNaC::Kronecker\_dz\_kernel, 727
- k
  - factor.cpp, 1195
  - GiNaC::Eisenstein\_h\_kernel, 484
  - GiNaC::Eisenstein\_kernel, 493
  - GiNaC::modular\_form\_kernel, 773
- Kronecker\_dtau\_kernel

- GiNaC::Kronecker\_dtau\_kernel, 717
- Kronecker\_dz\_kernel
  - GiNaC::Kronecker\_dz\_kernel, 725
- kronecker\_symbol
  - GiNaC, 175
- label
  - GiNaC::wildcard, 1159
- lanczos\_coeffs
  - GiNaC::lanczos\_coeffs, 727
- laplace
  - GiNaC::determinant\_algo, 439
- last
  - factor.cpp, 1195
- last\_access
  - GiNaC::remember\_table\_entry, 1007
- latex
  - GiNaC, 231
- latex\_name
  - GiNaC::function\_options, 616
- Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, 482
  - GiNaC::Eisenstein\_kernel, 492
  - GiNaC::integration\_kernel, 705
  - GiNaC::modular\_form\_kernel, 772
  - GiNaC::user\_defined\_kernel, 1142
- lcm
  - GiNaC, 197, 217
- lcm\_of\_coefficients\_denominators
  - GiNaC, 188
- lcmcoeff
  - GiNaC, 188
- lcoeff
  - GiNaC::ex, 527
- ldeg\_a
  - GiNaC::sym\_desc, 1080
- ldeg\_b
  - GiNaC::sym\_desc, 1080
- ldegree
  - GiNaC, 104
  - GiNaC::add, 286
  - GiNaC::basic, 327
  - GiNaC::ex, 526
  - GiNaC::integral, 695
  - GiNaC::mul, 783
  - GiNaC::ncmul, 839
  - GiNaC::numeric, 855
  - GiNaC::power, 914
  - GiNaC::pseries, 961
  - GiNaC::structure< T, ComparisonPolicy >, 1046
- len
  - factor.cpp, 1195
- less
  - GiNaC::relational, 994
- less\_or\_equal
  - GiNaC::relational, 994
- let\_op
  - GiNaC::basic, 324
  - GiNaC::clifford, 369
- GiNaC::container< C >, 424
  - GiNaC::Ebar\_kernel, 473
  - GiNaC::Eisenstein\_h\_kernel, 481
  - GiNaC::Eisenstein\_kernel, 492
  - GiNaC::ELi\_kernel, 502
  - GiNaC::ex, 522
  - GiNaC::integral, 695
  - GiNaC::Kronecker\_dtau\_kernel, 717
  - GiNaC::Kronecker\_dz\_kernel, 725
  - GiNaC::matrix, 743
  - GiNaC::modular\_form\_kernel, 771
  - GiNaC::multiple\_polylog\_kernel, 830
  - GiNaC::structure< T, ComparisonPolicy >, 1044
  - GiNaC::user\_defined\_kernel, 1142
- lgamma
  - GiNaC, 210, 211
- lgamma\_conjugate
  - GiNaC, 146
- lgamma\_deriv
  - GiNaC, 146
- lgamma\_eval
  - GiNaC, 146
- lgamma\_evalf
  - GiNaC, 145
- lgamma\_series
  - GiNaC, 146
- lh
  - GiNaC::relational, 1000
- lhs
  - GiNaC, 109
  - GiNaC::ex, 522
  - GiNaC::relational, 998
- Li2
  - GiNaC, 210
- Li2\_
  - GiNaC, 209
- Li2\_conjugate
  - GiNaC, 134
- Li2\_deriv
  - GiNaC, 134
- Li2\_eval
  - GiNaC, 133
- Li2\_evalf
  - GiNaC, 133
- Li2\_projection
  - GiNaC, 209
- Li2\_series
  - GiNaC, 134, 209
- Li3\_eval
  - GiNaC, 134
- Li\_deriv
  - GiNaC, 152
- Li\_eval
  - GiNaC, 151
- Li\_evalf
  - GiNaC, 151
- Li\_print\_latex
  - GiNaC, 152

- Li\_series
  - GiNaC, [151](#)
- library\_init
  - GiNaC::library\_init, [730](#)
- library\_initializer
  - GiNaC, [256](#)
- likely
  - compiler.h, [1179](#)
- link\_ex
  - GiNaC, [112](#), [113](#)
- list
  - GiNaC::info\_flags, [688](#)
- log
  - GiNaC, [204](#)
- log2
  - GiNaC, [248](#)
- log\_conjugate
  - GiNaC, [159](#)
- log\_deriv
  - GiNaC, [158](#)
- log\_eval
  - GiNaC, [158](#)
- log\_evalf
  - GiNaC, [158](#)
- log\_expand
  - GiNaC, [159](#)
- log\_imag\_part
  - GiNaC, [159](#)
- log\_info
  - GiNaC, [159](#)
- log\_real\_part
  - GiNaC, [158](#)
- log\_series
  - GiNaC, [158](#)
- lookup\_entry
  - GiNaC::remember\_table, [1003](#)
  - GiNaC::remember\_table\_list, [1009](#)
- lookup\_map
  - GiNaC, [81](#)
- lookup\_remember\_table
  - GiNaC::function, [607](#)
- lorentz\_eps
  - GiNaC, [247](#)
- lorentz\_g
  - GiNaC, [246](#)
- lr
  - factor.cpp, [1194](#)
- lsolve
  - GiNaC, [139](#)
- lst
  - GiNaC, [81](#)
- lst.cpp, [1238](#)
- lst.h, [1238](#)
- lst\_to\_clifford
  - GiNaC, [92](#)
- lst\_to\_matrix
  - GiNaC, [181](#)
- m
  - factor.cpp, [1193](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [774](#)
  - GiNaC::Ebar\_kernel, [474](#)
  - GiNaC::ELi\_kernel, [503](#)
  - GiNaC::matrix, [757](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [878](#)
  - make\_flat
    - GiNaC::expairseq, [565](#)
  - make\_flat\_inserter
    - GiNaC::make\_flat\_inserter, [732](#)
  - make\_hash\_seed
    - GiNaC, [116](#)
  - make\_real\_float
    - GiNaC, [201](#)
  - make\_return\_type\_t
    - GiNaC, [235](#)
  - make\_safe\_bool
    - GiNaC::relational, [999](#)
  - makewritable
    - GiNaC::ptr< T >, [974](#)
  - makewritable
    - GiNaC::ex, [541](#)
  - map
    - GiNaC::basic, [326](#)
    - GiNaC::ex, [525](#)
    - GiNaC::expairseq, [557](#)
    - GiNaC::idx, [662](#)
    - GiNaC::power, [913](#)
    - GiNaC::relational, [996](#)
    - GiNaC::structure< T, ComparisonPolicy >, [1045](#)
  - map\_eval\_integ
    - GiNaC, [254](#)
  - map\_evalm
    - GiNaC, [254](#)
  - markowitz
    - GiNaC::solve\_algo, [1016](#)
  - markowitz\_elimination
    - GiNaC::matrix, [755](#)
  - match
    - GiNaC, [107](#)
    - GiNaC::basic, [325](#)
    - GiNaC::ex, [524](#)
    - GiNaC::expairseq, [558](#)
    - GiNaC::structure< T, ComparisonPolicy >, [1044](#)
    - GiNaC::wildcard, [1158](#)
  - match\_same\_type
    - GiNaC::basic, [326](#)
    - GiNaC::clifford, [367](#)
    - GiNaC::color, [386](#)
    - GiNaC::fderivative, [587](#)
    - GiNaC::function, [605](#)
    - GiNaC::idx, [664](#)
    - GiNaC::matrix, [745](#)
    - GiNaC::relational, [997](#)
    - GiNaC::spinidx, [1024](#)
    - GiNaC::structure< T, ComparisonPolicy >, [1045](#)
    - GiNaC::varidx, [1150](#)

- matrix
  - GiNaC::matrix, [741](#), [742](#)
- matrix.cpp, [1239](#)
- matrix.h, [1240](#)
- max\_assoc\_size
  - GiNaC::remember\_table, [1004](#)
  - GiNaC::remember\_table\_list, [1009](#)
- max\_coefficient
  - GiNaC::add, [289](#)
  - GiNaC::basic, [331](#)
  - GiNaC::ex, [534](#)
  - GiNaC::mul, [786](#)
  - GiNaC::numeric, [858](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1050](#)
- max\_deg
  - GiNaC::sym\_desc, [1080](#)
- max\_integration\_level
  - GiNaC::integral, [699](#)
- max\_lcnops
  - GiNaC::sym\_desc, [1081](#)
- metric
  - GiNaC::clifford, [370](#)
- metric\_tensor
  - GiNaC, [245](#)
- minimal\_dim
  - GiNaC, [118](#)
  - GiNaC::idx, [666](#)
- minkmetric
  - GiNaC::minkmetric, [763](#)
- minkowski
  - GiNaC::tensepsilon, [1119](#)
- mod
  - GiNaC, [214](#)
- modular\_form\_kernel
  - GiNaC::modular\_form\_kernel, [770](#)
- modulus
  - factor.cpp, [1198](#)
- mpartition2
  - GiNaC::basic\_partition\_generator::mpartition2, [774](#)
- mpgen
  - GiNaC::basic\_partition\_generator, [353](#)
- mul
  - GiNaC::add, [294](#)
  - GiNaC::matrix, [747](#)
  - GiNaC::mul, [781](#), [782](#)
  - GiNaC::numeric, [860](#)
  - GiNaC::power, [922](#)
- mul.cpp, [1241](#)
- mul.h, [1242](#)
- mul\_const
  - GiNaC::pseries, [967](#)
- mul\_dyn
  - GiNaC::numeric, [861](#)
- mul\_scalar
  - GiNaC::matrix, [747](#)
- mul\_series
  - GiNaC::pseries, [967](#)
- multi\_iterator\_counter
  - GiNaC::multi\_iterator\_counter< T >, [796](#)
- multi\_iterator\_counter\_indv
  - GiNaC::multi\_iterator\_counter\_indv< T >, [799](#)
- multi\_iterator\_ordered
  - GiNaC::multi\_iterator\_ordered< T >, [803](#)
- multi\_iterator\_ordered\_eq
  - GiNaC::multi\_iterator\_ordered\_eq< T >, [807](#)
- multi\_iterator\_ordered\_eq\_indv
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, [810](#)
- multi\_iterator\_permutation
  - GiNaC::multi\_iterator\_permutation< T >, [814](#)
- multi\_iterator\_shuffle
  - GiNaC::multi\_iterator\_shuffle< T >, [818](#)
- multi\_iterator\_shuffle\_prime
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, [823](#)
- multinomial\_coefficient
  - GiNaC, [249](#)
- multiple\_polylog\_kernel
  - GiNaC::multiple\_polylog\_kernel, [829](#)
- multiply\_lcm
  - GiNaC, [189](#)
- my\_ios\_callback
  - GiNaC, [229](#)
- my\_ios\_index
  - GiNaC, [229](#)
- N
  - GiNaC::basic\_multi\_iterator< T >, [351](#)
  - GiNaC::Eisenstein\_h\_kernel, [484](#)
  - GiNaC::Eisenstein\_kernel, [493](#)
- n
  - factor.cpp, [1194](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [774](#)
  - GiNaC::Ebar\_kernel, [474](#)
  - GiNaC::ELi\_kernel, [503](#)
  - GiNaC::Kronecker\_dtau\_kernel, [718](#)
  - GiNaC::Kronecker\_dz\_kernel, [726](#)
- N\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, [819](#)
- name
  - GiNaC::archive::archived\_ex, [315](#)
  - GiNaC::archive\_node::property, [952](#)
  - GiNaC::archive\_node::property\_info, [953](#)
  - GiNaC::constant, [415](#)
  - GiNaC::function\_options, [645](#)
  - GiNaC::print\_context\_options, [926](#)
  - GiNaC::registered\_class\_options, [988](#)
  - GiNaC::symbol, [1092](#)
- ncmul
  - GiNaC::mul, [793](#)
  - GiNaC::ncmul, [838](#), [839](#)
- ncmul.cpp, [1243](#)
- ncmul.h, [1244](#)
- negative
  - GiNaC::info\_flags, [688](#)
- negint
  - GiNaC::info\_flags, [688](#)

- next
  - GiNaC::class\_info< OPT >, 356
  - GiNaC::composition\_generator, 393
  - GiNaC::composition\_generator::coolmulti::element, 495
  - GiNaC::partition\_generator, 875
  - GiNaC::partition\_with\_zero\_parts\_generator, 877
- next\_partition
  - GiNaC::basic\_partition\_generator::mpartition2, 774
- next\_permutation
  - GiNaC::composition\_generator::coolmulti, 436
- next\_print\_context\_id
  - GiNaC, 257
- next\_serial
  - GiNaC::constant, 415
  - GiNaC::symbol, 1092
- no\_heur\_gcd
  - GiNaC::gcd\_options, 652
- no\_index\_dimensions
  - GiNaC, 232
- no\_index\_renaming
  - GiNaC::subs\_options, 1076
- no\_part\_factored
  - GiNaC::gcd\_options, 652
- no\_pattern
  - GiNaC::subs\_options, 1076
- no\_type
  - GiNaC::has\_distance< T >, 654
- nodes
  - GiNaC::archive, 301
- noncommutative
  - GiNaC::return\_types, 1012
- noncommutative\_composite
  - GiNaC::return\_types, 1012
- none
  - GiNaC::symmetry, 1099
- nonnegative
  - GiNaC::info\_flags, 688
- nonnegint
  - GiNaC::info\_flags, 688
- nonnull
  - GiNaC::relational::safe\_bool\_helper, 1012
- nops
  - GiNaC, 102, 183
  - GiNaC::basic, 323
  - GiNaC::clifford, 368
  - GiNaC::container< C >, 424
  - GiNaC::Ebar\_kernel, 473
  - GiNaC::Eisenstein\_h\_kernel, 481
  - GiNaC::Eisenstein\_kernel, 491
  - GiNaC::ELi\_kernel, 502
  - GiNaC::ex, 520
  - GiNaC::expairseq, 556
  - GiNaC::idx, 662
  - GiNaC::integral, 695
  - GiNaC::Kronecker\_dtau\_kernel, 717
  - GiNaC::Kronecker\_dz\_kernel, 725
  - GiNaC::matrix, 742
  - GiNaC::modular\_form\_kernel, 771
  - GiNaC::multiple\_polylog\_kernel, 830
  - GiNaC::power, 913
  - GiNaC::pseries, 960
  - GiNaC::relational, 995
  - GiNaC::structure< T, ComparisonPolicy >, 1043
  - GiNaC::user\_defined\_kernel, 1142
- normal
  - GiNaC, 105
  - GiNaC::add, 288
  - GiNaC::basic, 329
  - GiNaC::ex, 529
  - GiNaC::mul, 785
  - GiNaC::numeric, 856
  - GiNaC::power, 917
  - GiNaC::pseries, 962
  - GiNaC::structure< T, ComparisonPolicy >, 1048
  - GiNaC::symbol, 1087
- normal.cpp, 1244
  - FAST\_COMPARE, 1246
  - STATISTICS, 1247
  - USE\_REMEMBER, 1246
  - USE\_TRIAL\_DIVISION, 1247
- normal.h, 1247
- not\_equal
  - GiNaC::relational, 994
- not\_shareable
  - GiNaC::status\_flags, 1036
- not\_symmetric
  - GiNaC, 238
- nparams
  - GiNaC::function\_options, 645
- num
  - GiNaC::symminfo, 1105
- num\_expressions
  - GiNaC::archive, 299
- number
  - GiNaC::constant, 415
- number\_of\_type
  - GiNaC, 120
- numer
  - GiNaC, 105, 222
  - GiNaC::ex, 530
  - GiNaC::numeric, 870
- numer\_denom
  - GiNaC, 105
  - GiNaC::ex, 531
- numeric
  - GiNaC::info\_flags, 688
  - GiNaC::numeric, 852, 853
- numeric.cpp, 1248
- numeric.h, 1251
- Nv
  - GiNaC::multi\_iterator\_counter\_indv< T >, 801
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 812
- o
  - GiNaC::relational, 1000

- obj
  - GiNaC::structure< T, ComparisonPolicy >, 1054
- odd
  - GiNaC::info\_flags, 688
- one
  - factor.cpp, 1194
- op
  - GiNaC, 109
  - GiNaC::basic, 324
  - GiNaC::clifford, 369
  - GiNaC::container< C >, 424
  - GiNaC::Ebar\_kernel, 473
  - GiNaC::Eisenstein\_h\_kernel, 481
  - GiNaC::Eisenstein\_kernel, 492
  - GiNaC::ELi\_kernel, 502
  - GiNaC::ex, 521
  - GiNaC::expairseq, 557
  - GiNaC::idx, 662
  - GiNaC::integral, 695
  - GiNaC::Kronecker\_dtau\_kernel, 717
  - GiNaC::Kronecker\_dz\_kernel, 725
  - GiNaC::matrix, 742
  - GiNaC::modular\_form\_kernel, 771
  - GiNaC::multiple\_polylog\_kernel, 830
  - GiNaC::power, 913
  - GiNaC::pseries, 960
  - GiNaC::relational, 995
  - GiNaC::structure< T, ComparisonPolicy >, 1043
  - GiNaC::user\_defined\_kernel, 1142
- operator long
  - GiNaC::\_numeric\_digits, 276
- operator safe\_bool
  - GiNaC::relational, 999
- operator!
  - GiNaC::relational, 999
- operator!=
  - GiNaC, 228
  - GiNaC::const\_iterator, 398
  - GiNaC::const\_postorder\_iterator, 402
  - GiNaC::const\_preorder\_iterator, 405
  - GiNaC::internal::\_iter\_rep, 274
  - GiNaC::numeric, 867
  - GiNaC::ptr< T >, 975, 976
  - GiNaC::return\_type\_t, 1010
- operator<
  - GiNaC, 229
  - GiNaC::const\_iterator, 398
  - GiNaC::numeric, 867
  - GiNaC::return\_type\_t, 1010
  - GiNaC::spmapkey, 1035
  - GiNaC::sym\_desc, 1080
- operator<<
  - GiNaC, 83, 102, 218, 230, 251–253
  - GiNaC::archive, 301
  - GiNaC::archive\_node, 311
  - GiNaC::basic\_multi\_iterator< T >, 351
  - GiNaC::multi\_iterator\_counter< T >, 797
  - GiNaC::multi\_iterator\_counter\_indv< T >, 800
  - GiNaC::multi\_iterator\_ordered< T >, 804
  - GiNaC::multi\_iterator\_ordered\_eq< T >, 808
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 811
  - GiNaC::multi\_iterator\_permutation< T >, 816
  - GiNaC::multi\_iterator\_shuffle< T >, 819
  - GiNaC::multi\_iterator\_shuffle\_prime< T >, 823
  - GiNaC::ptr< T >, 976
- operator<=
  - GiNaC, 229
  - GiNaC::const\_iterator, 398
  - GiNaC::numeric, 868
- operator>
  - GiNaC, 229
  - GiNaC::const\_iterator, 398
  - GiNaC::numeric, 868
- operator>>
  - GiNaC, 83, 231
  - GiNaC::archive, 301
  - GiNaC::archive\_node, 311
- operator>=
  - GiNaC, 229
  - GiNaC::const\_iterator, 398
  - GiNaC::numeric, 868
- operator()
  - GiNaC::basic\_multi\_iterator< T >, 350
  - GiNaC::derivative\_map\_function, 438
  - GiNaC::error\_and\_integral\_is\_less, 506
  - GiNaC::eval\_integ\_map\_function, 508
  - GiNaC::evalf\_map\_function, 509
  - GiNaC::evalm\_map\_function, 511
  - GiNaC::ex\_base\_is\_less, 543
  - GiNaC::ex\_is\_equal, 543
  - GiNaC::ex\_is\_less, 544
  - GiNaC::ex\_swap, 544
  - GiNaC::expair\_is\_less, 549
  - GiNaC::expair\_rest\_is\_less, 549
  - GiNaC::expair\_swap, 550
  - GiNaC::expand\_map\_function, 570
  - GiNaC::idx\_is\_equal\_ignore\_dim, 668
  - GiNaC::is\_not\_a\_clifford, 708
  - GiNaC::is\_summation\_idx, 709
  - GiNaC::map\_function, 735
  - GiNaC::matrix, 748
  - GiNaC::normal\_map\_function, 845
  - GiNaC::op0\_is\_equal, 873
  - GiNaC::pointer\_to\_map\_function, 880
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, 882
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, 884
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, 887
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, 890
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, 892
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, 895

- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, 898
  - GiNaC::print\_functor, 937
  - GiNaC::print\_functor\_impl, 938
  - GiNaC::print\_memfun\_handler< T, C >, 942
  - GiNaC::print\_ptrfun\_handler< T, C >, 945
  - GiNaC::sy\_is\_less, 1076
  - GiNaC::sy\_swap, 1077
  - GiNaC::symminfo\_is\_less\_by\_orig, 1106
  - GiNaC::symminfo\_is\_less\_by\_symmterm, 1106
  - GiNaC::terminfo\_is\_less, 1133
  - std::equal\_to< GiNaC::ex >, 504
  - std::hash< GiNaC::ex >, 656
  - std::less< GiNaC::ptr< T > >, 729
- operator+
  - GiNaC, 223, 224, 226
  - GiNaC::const\_iterator, 397, 399
- operator++
  - GiNaC, 227, 228
  - GiNaC::basic\_multi\_iterator< T >, 350
  - GiNaC::const\_iterator, 397
  - GiNaC::const\_postorder\_iterator, 401
  - GiNaC::const\_preorder\_iterator, 404
  - GiNaC::multi\_iterator\_counter< T >, 796
  - GiNaC::multi\_iterator\_counter\_indv< T >, 800
  - GiNaC::multi\_iterator\_ordered< T >, 804
  - GiNaC::multi\_iterator\_ordered\_eq< T >, 807
  - GiNaC::multi\_iterator\_ordered\_eq\_indv< T >, 811
  - GiNaC::multi\_iterator\_permutation< T >, 815
  - GiNaC::multi\_iterator\_shuffle< T >, 819
- operator+=
  - GiNaC, 225
  - GiNaC::const\_iterator, 397
- operator-
  - GiNaC, 224, 226
  - GiNaC::const\_iterator, 398, 399
- operator->
  - GiNaC::const\_iterator, 396
  - GiNaC::const\_postorder\_iterator, 401
  - GiNaC::const\_preorder\_iterator, 404
  - GiNaC::ptr< T >, 974
  - GiNaC::structure< T, ComparisonPolicy >, 1053
- operator--
  - GiNaC, 227, 228
  - GiNaC::const\_iterator, 397
- operator-=
  - GiNaC, 225, 226
  - GiNaC::const\_iterator, 397
- operator/
  - GiNaC, 224, 225
- operator/=
  - GiNaC, 225, 226
- operator=
  - GiNaC::\_numeric\_digits, 276
  - GiNaC::archive\_node, 306
  - GiNaC::basic, 320
  - GiNaC::numeric, 862, 863
  - GiNaC::print\_functor, 937
  - GiNaC::ptr< T >, 974
  - operator==
    - GiNaC, 228
    - GiNaC::const\_iterator, 398
    - GiNaC::const\_postorder\_iterator, 402
    - GiNaC::const\_preorder\_iterator, 405
    - GiNaC::internal::\_iter\_rep, 274
    - GiNaC::numeric, 867
    - GiNaC::ptr< T >, 974, 975
    - GiNaC::return\_type\_t, 1010
    - GiNaC::spmapkey, 1035
  - operator[]
    - GiNaC::basic, 324, 325
    - GiNaC::basic\_multi\_iterator< T >, 349
    - GiNaC::const\_iterator, 396
    - GiNaC::ex, 521, 522
    - GiNaC::structure< T, ComparisonPolicy >, 1043, 1044
  - operator\*
    - GiNaC, 224
    - GiNaC::const\_iterator, 396
    - GiNaC::const\_postorder\_iterator, 401
    - GiNaC::const\_preorder\_iterator, 404
    - GiNaC::ptr< T >, 974
  - operator\*=
    - GiNaC, 225, 226
  - operators
    - GiNaC::relational, 994
  - operators.cpp, 1254
  - operators.h, 1256
  - options
    - factor.cpp, 1198
    - GiNaC::class\_info< OPT >, 356
    - GiNaC::expand\_map\_function, 570
    - GiNaC::print\_context, 924
  - order
    - integration\_kernel.cpp, 1236
  - Order\_conjugate
    - GiNaC, 138
  - Order\_eval
    - GiNaC, 137
  - Order\_expl\_derivative
    - GiNaC, 138
  - Order\_imag\_part
    - GiNaC, 138
  - Order\_power
    - GiNaC, 138
  - Order\_real\_part
    - GiNaC, 138
  - Order\_series
    - GiNaC, 138
  - orig
    - GiNaC::symminfo, 1105
    - GiNaC::terminfo, 1132
  - overall\_coeff
    - GiNaC::expairseq, 567
  - overflow
    - GiNaC::basic\_multi\_iterator< T >, 349

- overloaded
  - GiNaC::function\_options, [643](#)
- P
  - GiNaC::modular\_form\_kernel, [773](#)
- p
  - GiNaC::ptr< T >, [976](#)
- parameter\_set
  - GiNaC::fderivative, [589](#)
- paramset
  - GiNaC, [59](#)
- parent
  - GiNaC::class\_info< OPT >, [356](#)
- parent\_name
  - GiNaC::print\_context\_options, [926](#)
  - GiNaC::registered\_class\_options, [988](#)
- parents\_identified
  - GiNaC::class\_info< OPT >, [356](#)
- partition
  - GiNaC::partition\_generator, [875](#)
  - GiNaC::partition\_with\_zero\_parts\_generator, [878](#)
- partition\_generator
  - GiNaC::partition\_generator, [875](#)
- partition\_with\_zero\_parts\_generator
  - GiNaC::partition\_with\_zero\_parts\_generator, [877](#)
- pattern\_is\_not\_product
  - GiNaC::subs\_options, [1076](#)
- pattern\_is\_product
  - GiNaC::subs\_options, [1076](#)
- pderivative
  - GiNaC::function, [606](#)
- permutation\_sign
  - GiNaC, [250](#)
- permute\_free\_index\_to\_front
  - GiNaC, [97](#)
- Pi
  - GiNaC, [255](#)
- PiEvalf
  - GiNaC, [218](#)
- pivot
  - GiNaC::matrix, [755](#)
- point
  - GiNaC::pseries, [970](#)
- pointer
  - GiNaC::const\_iterator, [396](#)
  - GiNaC::const\_postorder\_iterator, [400](#)
  - GiNaC::const\_preorder\_iterator, [403](#)
- pointer\_to\_map\_function
  - GiNaC::pointer\_to\_map\_function, [880](#)
- pointer\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [882](#)
- pointer\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [884](#)
- pointer\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [887](#)
- pointer\_to\_member\_to\_map\_function
  - GiNaC::pointer\_to\_member\_to\_map\_function< C >, [890](#)
- pointer\_to\_member\_to\_map\_function\_1arg
  - GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [892](#)
- pointer\_to\_member\_to\_map\_function\_2args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [895](#)
- pointer\_to\_member\_to\_map\_function\_3args
  - GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [898](#)
- pole\_error
  - GiNaC::pole\_error, [900](#)
- poly
  - factor.cpp, [1196](#)
- polynomial
  - GiNaC::factor\_options, [571](#)
  - GiNaC::info\_flags, [688](#)
- pos\_sig
  - GiNaC::minkmetric, [765](#)
  - GiNaC::tensepsilon, [1119](#)
- posint
  - GiNaC::info\_flags, [688](#)
- positive
  - GiNaC::domain, [466](#)
  - GiNaC::info\_flags, [688](#)
- possymbol
  - GiNaC::possymbol, [906](#)
- postorder\_begin
  - GiNaC::ex, [518](#)
- postorder\_end
  - GiNaC::ex, [518](#)
- pow
  - GiNaC, [219](#), [233](#)
  - GiNaC::matrix, [748](#)
- power
  - GiNaC::add, [294](#)
  - GiNaC::function, [607](#)
  - GiNaC::mul, [793](#)
  - GiNaC::ncmul, [843](#)
  - GiNaC::numeric, [861](#)
  - GiNaC::power, [912](#)
- power.cpp, [1258](#)
- power.h, [1258](#)
- power\_const
  - GiNaC::pseries, [968](#)
- power\_dyn
  - GiNaC::numeric, [862](#)
- power\_f
  - GiNaC::function\_options, [647](#)
- power\_func
  - GiNaC::function\_options, [632–634](#), [640](#)
- power\_funcp
  - GiNaC, [60](#)
- power\_funcp\_1
  - GiNaC, [62](#)
- power\_funcp\_10
  - GiNaC, [73](#)

- power\_funcp\_11
  - GiNaC, [74](#)
- power\_funcp\_12
  - GiNaC, [76](#)
- power\_funcp\_13
  - GiNaC, [77](#)
- power\_funcp\_14
  - GiNaC, [79](#)
- power\_funcp\_2
  - GiNaC, [63](#)
- power\_funcp\_3
  - GiNaC, [64](#)
- power\_funcp\_4
  - GiNaC, [65](#)
- power\_funcp\_5
  - GiNaC, [66](#)
- power\_funcp\_6
  - GiNaC, [68](#)
- power\_funcp\_7
  - GiNaC, [69](#)
- power\_funcp\_8
  - GiNaC, [70](#)
- power\_funcp\_9
  - GiNaC, [72](#)
- power\_funcp\_exvector
  - GiNaC, [80](#)
- power\_use\_exvector\_args
  - GiNaC::function\_options, [649](#)
- pp
  - factor.cpp, [1197](#)
- precedence
  - GiNaC::add, [285](#)
  - GiNaC::basic, [323](#)
  - GiNaC::clifford, [366](#)
  - GiNaC::container< C >, [423](#)
  - GiNaC::expairseq, [556](#)
  - GiNaC::function, [601](#)
  - GiNaC::indexed, [681](#)
  - GiNaC::integral, [694](#)
  - GiNaC::mul, [782](#)
  - GiNaC::ncmul, [839](#)
  - GiNaC::numeric, [854](#)
  - GiNaC::power, [912](#)
  - GiNaC::pseries, [960](#)
  - GiNaC::relational, [995](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1043](#)
- prem
  - GiNaC, [191](#)
- preorder\_begin
  - GiNaC::ex, [517](#)
- preorder\_end
  - GiNaC::ex, [517](#)
- prepend
  - GiNaC::container< C >, [428](#)
- prime
  - GiNaC::info\_flags, [688](#)
- primitive\_dirichlet\_character
  - GiNaC, [175](#)
- primpart
  - GiNaC::ex, [533](#)
- print
  - GiNaC::\_numeric\_digits, [276](#)
  - GiNaC::basic, [322](#)
  - GiNaC::ex, [519](#)
  - GiNaC::expair, [547](#)
  - GiNaC::fderivative, [585](#)
  - GiNaC::function, [601](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1042](#)
- print.cpp, [1259](#)
- print.h, [1260](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT, [1261](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_BASE, [1261](#)
  - GINAC\_DECLARE\_PRINT\_CONTEXT\_COMMON, [1261](#)
  - GINAC\_IMPLEMENT\_PRINT\_CONTEXT, [1262](#)
- print\_add
  - GiNaC::add, [292](#)
- print\_context
  - GiNaC::print\_context, [924](#)
- print\_context\_class\_info
  - GiNaC, [82](#)
- print\_context\_options
  - GiNaC::print\_context\_options, [925](#)
- print\_csrc
  - GiNaC::print\_csrc, [928](#)
- print\_csrc\_cl\_N
  - GiNaC::print\_csrc\_cl\_N, [930](#)
- print\_csrc\_double
  - GiNaC::print\_csrc\_double, [932](#)
- print\_csrc\_float
  - GiNaC::print\_csrc\_float, [934](#)
- print\_dflt
  - GiNaC::print\_dflt, [935](#)
- print\_dispatch
  - GiNaC::basic, [334](#), [335](#)
- print\_dispatch\_table
  - GiNaC::function\_options, [647](#)
  - GiNaC::registered\_class\_options, [988](#)
- print\_elements
  - GiNaC::matrix, [756](#)
- print\_func
  - GiNaC::function\_options, [640–642](#)
  - GiNaC::registered\_class\_options, [987](#), [988](#)
- print\_func< print\_context >
  - GiNaC, [116](#)
- print\_func< print\_dflt >
  - GiNaC, [85](#), [96](#), [244](#)
- print\_funcp
  - GiNaC, [61](#)
- print\_funcp\_1
  - GiNaC, [62](#)
- print\_funcp\_10
  - GiNaC, [73](#)
- print\_funcp\_11
  - GiNaC, [75](#)

- print\_funcp\_12
  - GiNaC, [76](#)
- print\_funcp\_13
  - GiNaC, [78](#)
- print\_funcp\_14
  - GiNaC, [79](#)
- print\_funcp\_2
  - GiNaC, [63](#)
- print\_funcp\_3
  - GiNaC, [64](#)
- print\_funcp\_4
  - GiNaC, [65](#)
- print\_funcp\_5
  - GiNaC, [67](#)
- print\_funcp\_6
  - GiNaC, [68](#)
- print\_funcp\_7
  - GiNaC, [69](#)
- print\_funcp\_8
  - GiNaC, [71](#)
- print\_funcp\_9
  - GiNaC, [72](#)
- print\_funcp\_exvector
  - GiNaC, [80](#)
- print\_functor
  - GiNaC::print\_functor, [936](#)
- print\_index
  - GiNaC::idx, [666](#)
- print\_index\_dimensions
  - GiNaC::print\_options, [943](#)
- print\_indexed
  - GiNaC::indexed, [685](#)
- print\_integer\_csrc
  - GiNaC, [202](#)
- print\_latex
  - GiNaC::print\_latex, [940](#)
- print\_memfun\_handler
  - GiNaC::print\_memfun\_handler< T, C >, [942](#)
- print\_numeric
  - GiNaC::numeric, [871](#)
- print\_operator
  - GiNaC, [236](#)
- print\_overall\_coeff
  - GiNaC::mul, [791](#)
- print\_power
  - GiNaC::power, [920](#)
- print\_ptrfun\_handler
  - GiNaC::print\_ptrfun\_handler< T, C >, [945](#)
- print\_python
  - GiNaC::print\_python, [947](#)
- print\_python\_repr
  - GiNaC::print\_python\_repr, [949](#)
- print\_real\_cl\_N
  - GiNaC, [204](#)
- print\_real\_csrc
  - GiNaC, [203](#)
- print\_real\_number
  - GiNaC, [202](#)
- print\_series
  - GiNaC::pseries, [968](#)
- print\_sym\_pow
  - GiNaC, [232](#)
- print\_tree
  - GiNaC::print\_tree, [950](#)
- print\_use\_exvector\_args
  - GiNaC::function\_options, [650](#)
- printindices
  - GiNaC::indexed, [685](#)
- printpair
  - GiNaC::expairseq, [561](#)
- printraw
  - GiNaC::archive, [300](#)
  - GiNaC::archive\_node, [310](#)
- printseq
  - GiNaC::container< C >, [427](#)
  - GiNaC::expairseq, [561](#)
- product\_to\_exvector
  - GiNaC, [121](#)
- property
  - GiNaC::archive\_node::property, [951](#)
- property\_info
  - GiNaC::archive\_node::property\_info, [953](#)
- property\_type
  - GiNaC::archive\_node, [305](#)
- propinfovector
  - GiNaC::archive\_node, [305](#)
- props
  - GiNaC::archive\_node, [311](#)
- pseries
  - GiNaC::pseries, [959](#), [960](#)
- pseries.cpp, [1262](#)
- pseries.h, [1263](#)
- psi
  - GiNaC, [141](#), [211](#)
- psi1\_deriv
  - GiNaC, [149](#)
- psi1\_eval
  - GiNaC, [149](#)
- psi1\_evalf
  - GiNaC, [149](#)
- psi1\_series
  - GiNaC, [149](#)
- psi2\_deriv
  - GiNaC, [150](#)
- psi2\_eval
  - GiNaC, [150](#)
- psi2\_evalf
  - GiNaC, [149](#)
- psi2\_series
  - GiNaC, [150](#)
- ptr
  - GiNaC::pointer\_to\_map\_function, [880](#)
  - GiNaC::pointer\_to\_map\_function\_1arg< T1 >, [882](#)
  - GiNaC::pointer\_to\_map\_function\_2args< T1, T2 >, [885](#)

- GiNaC::pointer\_to\_map\_function\_3args< T1, T2, T3 >, [887](#)
- GiNaC::pointer\_to\_member\_to\_map\_function< C >, [890](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_1arg< C, T1 >, [893](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_2args< C, T1, T2 >, [895](#)
- GiNaC::pointer\_to\_member\_to\_map\_function\_3args< C, T1, T2, T3 >, [898](#)
- GiNaC::ptr< T >, [973](#)
- ptr.h, [1264](#)
- PTYPE\_BOOL
  - GiNaC::archive\_node, [305](#)
- PTYPE\_NODE
  - GiNaC::archive\_node, [305](#)
- PTYPE\_STRING
  - GiNaC::archive\_node, [305](#)
- PTYPE\_UNSIGNED
  - GiNaC::archive\_node, [305](#)
- purely\_indefinite
  - GiNaC::status\_flags, [1036](#)
- python
  - GiNaC, [231](#)
- python\_repr
  - GiNaC, [231](#)
- q\_expansion\_modular\_form
  - GiNaC::Eisenstein\_h\_kernel, [483](#)
  - GiNaC::Eisenstein\_kernel, [493](#)
  - GiNaC::modular\_form\_kernel, [772](#)
- qbar
  - integration\_kernel.cpp, [1236](#)
- quo
  - GiNaC, [189](#)
- R
  - factor.cpp, [1197](#)
- r
  - factor.cpp, [1192](#)
  - GiNaC::Eisenstein\_h\_kernel, [484](#)
- rank
  - GiNaC, [184](#), [185](#)
  - GiNaC::matrix, [752](#)
- rational
  - GiNaC::info\_flags, [688](#)
- rational\_function
  - GiNaC::info\_flags, [688](#)
- rational\_polynomial
  - GiNaC::info\_flags, [688](#)
- rbegin
  - GiNaC::container< C >, [430](#)
- read\_archive
  - GiNaC::basic, [335](#)
  - GiNaC::clifford, [366](#)
  - GiNaC::color, [386](#)
  - GiNaC::constant, [413](#)
  - GiNaC::container< C >, [425](#)
  - GiNaC::expairseq, [559](#)
  - GiNaC::fderivative, [586](#)
  - GiNaC::function, [604](#)
  - GiNaC::idx, [663](#)
  - GiNaC::indexed, [682](#)
  - GiNaC::integral, [697](#)
  - GiNaC::matrix, [745](#)
  - GiNaC::minkmetric, [764](#)
  - GiNaC::numeric, [859](#)
  - GiNaC::power, [918](#)
  - GiNaC::pseries, [964](#)
  - GiNaC::relational, [996](#)
  - GiNaC::spinidx, [1024](#)
  - GiNaC::symbol, [1089](#)
  - GiNaC::symmetry, [1100](#)
  - GiNaC::tensepsilon, [1118](#)
  - GiNaC::varidx, [1150](#)
  - GiNaC::wildcard, [1158](#)
- read\_real\_float
  - GiNaC, [201](#)
- read\_unsigned
  - GiNaC, [83](#)
- real
  - GiNaC, [222](#)
  - GiNaC::domain, [466](#)
  - GiNaC::info\_flags, [688](#)
  - GiNaC::numeric, [869](#)
- real\_part
  - GiNaC, [103](#)
  - GiNaC::add, [289](#)
  - GiNaC::basic, [333](#)
  - GiNaC::constant, [412](#)
  - GiNaC::container< C >, [426](#)
  - GiNaC::ex, [523](#)
  - GiNaC::function, [604](#)
  - GiNaC::indexed, [681](#)
  - GiNaC::matrix, [745](#)
  - GiNaC::mul, [785](#)
  - GiNaC::ncmul, [841](#)
  - GiNaC::numeric, [858](#)
  - GiNaC::power, [918](#)
  - GiNaC::pseries, [963](#)
  - GiNaC::realsymbol, [982](#)
  - GiNaC::symbol, [1088](#)
- real\_part\_conjugate
  - GiNaC, [126](#)
- real\_part\_eval
  - GiNaC, [125](#)
- real\_part\_evalf
  - GiNaC, [125](#)
- real\_part\_expl\_derivative
  - GiNaC, [126](#)
- real\_part\_f
  - GiNaC::function\_options, [646](#)
- real\_part\_func
  - GiNaC::function\_options, [622–624](#), [639](#)
- real\_part\_funcp
  - GiNaC, [60](#)
- real\_part\_funcp\_1

- GiNaC, [61](#)
- real\_part\_funcp\_10
  - GiNaC, [73](#)
- real\_part\_funcp\_11
  - GiNaC, [74](#)
- real\_part\_funcp\_12
  - GiNaC, [75](#)
- real\_part\_funcp\_13
  - GiNaC, [77](#)
- real\_part\_funcp\_14
  - GiNaC, [78](#)
- real\_part\_funcp\_2
  - GiNaC, [62](#)
- real\_part\_funcp\_3
  - GiNaC, [63](#)
- real\_part\_funcp\_4
  - GiNaC, [65](#)
- real\_part\_funcp\_5
  - GiNaC, [66](#)
- real\_part\_funcp\_6
  - GiNaC, [67](#)
- real\_part\_funcp\_7
  - GiNaC, [69](#)
- real\_part\_funcp\_8
  - GiNaC, [70](#)
- real\_part\_funcp\_9
  - GiNaC, [71](#)
- real\_part\_funcp\_exvector
  - GiNaC, [80](#)
- real\_part\_imag\_part
  - GiNaC, [126](#)
- real\_part\_print\_latex
  - GiNaC, [126](#)
- real\_part\_real\_part
  - GiNaC, [126](#)
- real\_part\_use\_exvector\_args
  - GiNaC::function\_options, [649](#)
- really\_subs\_idx
  - GiNaC::subs\_options, [1076](#)
- realsymbol
  - GiNaC::realsymbol, [982](#)
- recombine\_pair\_to\_ex
  - GiNaC::add, [292](#)
  - GiNaC::expairseq, [562](#)
  - GiNaC::mul, [789](#)
- reduced\_matrix
  - GiNaC, [182](#)
- reeval\_ncmul
  - GiNaC, [186](#)
  - GiNaC::ncmul, [843](#)
- refcount
  - GiNaC::refcounted, [986](#)
- refcounted
  - GiNaC::refcounted, [985](#)
- reference
  - GiNaC::const\_iterator, [396](#)
  - GiNaC::const\_postorder\_iterator, [401](#)
  - GiNaC::const\_preorder\_iterator, [404](#)
- REGISTER\_FUNCTION
  - function.h, [1213](#)
  - GiNaC, [125–127](#), [129](#), [131–137](#), [139](#), [143](#), [144](#), [146](#), [148](#), [152–154](#), [157](#), [159](#), [161–169](#), [171–173](#)
- register\_new
  - GiNaC::function, [607](#)
- registered\_class\_info
  - GiNaC, [82](#)
- registered\_class\_options
  - GiNaC::registered\_class\_options, [987](#)
- registered\_functions
  - GiNaC::function, [607](#)
- registrar.cpp, [1264](#)
- registrar.h, [1265](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS, [1266](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_COMMON, [1266](#)
  - GINAC\_DECLARE\_REGISTERED\_CLASS\_NO\_CTORS, [1266](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS, [1267](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT, [1267](#)
  - GINAC\_IMPLEMENT\_REGISTERED\_CLASS\_OPT\_T, [1267](#)
- relation
  - GiNaC::info\_flags, [688](#)
- relation\_equal
  - GiNaC::info\_flags, [688](#)
- relation\_greater
  - GiNaC::info\_flags, [688](#)
- relation\_greater\_or\_equal
  - GiNaC::info\_flags, [688](#)
- relation\_less
  - GiNaC::info\_flags, [688](#)
- relation\_less\_or\_equal
  - GiNaC::info\_flags, [688](#)
- relation\_not\_equal
  - GiNaC::info\_flags, [688](#)
- relational
  - GiNaC::relational, [995](#)
- relational.cpp, [1268](#)
- relational.h, [1268](#)
- relative\_integration\_error
  - GiNaC::integral, [699](#)
- rem
  - GiNaC, [190](#)
- remember
  - GiNaC::function\_options, [643](#)
- remember.cpp, [1269](#)
- remember.h, [1269](#)
- remember\_assoc\_size
  - GiNaC::function\_options, [648](#)
- remember\_size
  - GiNaC::function\_options, [648](#)
- remember\_strategy
  - GiNaC::function\_options, [648](#)

- GiNaC::remember\_table, 1004
  - GiNaC::remember\_table\_list, 1009
- remember\_table
  - GiNaC::remember\_table, 1002
- remember\_table\_entry
  - GiNaC::function, 608
  - GiNaC::remember\_table\_entry, 1006
- remember\_table\_list
  - GiNaC::remember\_table\_list, 1009
- remember\_tables
  - GiNaC::remember\_table, 1003
- remove\_all
  - GiNaC::container< C >, 429
- remove\_dirac\_ONE
  - GiNaC, 91
- remove\_first
  - GiNaC::container< C >, 429
- remove\_last
  - GiNaC::container< C >, 429
- remove\_reference
  - GiNaC::refcounted, 985
- rename\_dummy\_indices
  - GiNaC, 120
- rename\_dummy\_indices\_uniquely
  - GiNaC, 122, 123
- rend
  - GiNaC::container< C >, 430
- replace\_contr\_index
  - GiNaC::tensor, 1130
- replace\_dim
  - GiNaC::idx, 666
- replace\_with\_symbol
  - GiNaC, 199
- reposition\_dummy\_indices
  - GiNaC, 120
  - GiNaC::indexed, 686
- representation\_label
  - GiNaC::clifford, 370
  - GiNaC::color, 388
- reserve
  - GiNaC::container\_storage< C >, 434
- rest
  - GiNaC::expair, 548
- result
  - GiNaC::remember\_table\_entry, 1007
- result\_type
  - GiNaC::map\_function, 735
- resultant
  - GiNaC, 201
- return\_type
  - GiNaC::add, 290
  - GiNaC::basic, 333
  - GiNaC::clifford, 367
  - GiNaC::color, 387
  - GiNaC::ex, 538
  - GiNaC::expairseq, 560
  - GiNaC::fail, 575
  - GiNaC::function, 606
  - GiNaC::function\_options, 648
  - GiNaC::indexed, 683
  - GiNaC::integral, 696
  - GiNaC::matrix, 746
  - GiNaC::minkmetric, 764
  - GiNaC::mul, 788
  - GiNaC::ncmul, 842
  - GiNaC::power, 919
  - GiNaC::relational, 997
  - GiNaC::structure< T, ComparisonPolicy >, 1052
  - GiNaC::su3d, 1059
  - GiNaC::su3f, 1064
  - GiNaC::tensdelta, 1112
  - GiNaC::tensepsilon, 1119
  - GiNaC::tensmetric, 1125
  - GiNaC::tensor, 1130
- return\_type\_tinfo
  - GiNaC::add, 290
  - GiNaC::basic, 333
  - GiNaC::clifford, 367
  - GiNaC::color, 387
  - GiNaC::ex, 538
  - GiNaC::function, 606
  - GiNaC::function\_options, 648
  - GiNaC::indexed, 683
  - GiNaC::integral, 696
  - GiNaC::mul, 788
  - GiNaC::ncmul, 842
  - GiNaC::power, 919
  - GiNaC::relational, 998
  - GiNaC::structure< T, ComparisonPolicy >, 1052
- rh
  - GiNaC::relational, 1000
- rhs
  - GiNaC, 109
  - GiNaC::ex, 522
  - GiNaC::relational, 999
- rl
  - GiNaC::return\_type\_t, 1011
- root
  - GiNaC::archive::archived\_ex, 315
- rotate\_left
  - GiNaC, 249
- row
  - GiNaC::matrix, 756
- rows
  - GiNaC, 183
  - GiNaC::matrix, 746
- s
  - GiNaC::const\_postorder\_iterator, 402
  - GiNaC::const\_preorder\_iterator, 405
  - GiNaC::derivative\_map\_function, 438
  - GiNaC::Eisenstein\_h\_kernel, 484
  - GiNaC::print\_context, 924
  - GiNaC::symbolset, 1094
- S\_deriv
  - GiNaC, 153
- S\_eval

- GiNaC, [152](#)
- S\_evalf
  - GiNaC, [152](#)
- S\_print\_latex
  - GiNaC, [153](#)
- S\_series
  - GiNaC, [153](#)
- safe\_bool
  - GiNaC::relational, [994](#)
- same\_metric
  - GiNaC::clifford, [368](#)
- scalar\_mul\_indexed
  - GiNaC::basic, [332](#)
  - GiNaC::matrix, [744](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1051](#)
- seq
  - GiNaC::container\_storage< C >, [434](#)
  - GiNaC::expairseq, [567](#)
  - GiNaC::pseries, [969](#)
  - GiNaC::remember\_table\_entry, [1007](#)
- serial
  - GiNaC::constant, [415](#)
  - GiNaC::function, [608](#)
  - GiNaC::G2\_SERIAL, [651](#)
  - GiNaC::G3\_SERIAL, [652](#)
  - GiNaC::iterated\_integral2\_SERIAL, [709](#)
  - GiNaC::iterated\_integral3\_SERIAL, [710](#)
  - GiNaC::psi1\_SERIAL, [971](#)
  - GiNaC::psi2\_SERIAL, [971](#)
  - GiNaC::symbol, [1092](#)
  - GiNaC::zeta1\_SERIAL, [1160](#)
  - GiNaC::zeta2\_SERIAL, [1161](#)
- series
  - GiNaC, [107](#)
  - GiNaC::add, [287](#)
  - GiNaC::basic, [329](#)
  - GiNaC::Eisenstein\_h\_kernel, [481](#)
  - GiNaC::Eisenstein\_kernel, [491](#)
  - GiNaC::ex, [528](#)
  - GiNaC::fderivative, [585](#)
  - GiNaC::function, [603](#)
  - GiNaC::integral, [698](#)
  - GiNaC::integration\_kernel, [705](#)
  - GiNaC::modular\_form\_kernel, [771](#)
  - GiNaC::mul, [785](#)
  - GiNaC::power, [916](#)
  - GiNaC::pseries, [962](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1048](#)
  - GiNaC::symbol, [1086](#)
- series\_coeff
  - GiNaC::integration\_kernel, [707](#)
- series\_coeff\_impl
  - GiNaC::basic\_log\_kernel, [346](#)
  - GiNaC::Ebar\_kernel, [474](#)
  - GiNaC::ELi\_kernel, [503](#)
  - GiNaC::integration\_kernel, [706](#)
  - GiNaC::Kronecker\_dtau\_kernel, [718](#)
  - GiNaC::Kronecker\_dz\_kernel, [726](#)
  - GiNaC::multiple\_polylog\_kernel, [830](#)
- series\_f
  - GiNaC::function\_options, [647](#)
- series\_func
  - GiNaC::function\_options, [634–636](#), [640](#)
- series\_funcp
  - GiNaC, [60](#)
- series\_funcp\_1
  - GiNaC, [62](#)
- series\_funcp\_10
  - GiNaC, [73](#)
- series\_funcp\_11
  - GiNaC, [75](#)
- series\_funcp\_12
  - GiNaC, [76](#)
- series\_funcp\_13
  - GiNaC, [78](#)
- series\_funcp\_14
  - GiNaC, [79](#)
- series\_funcp\_2
  - GiNaC, [63](#)
- series\_funcp\_3
  - GiNaC, [64](#)
- series\_funcp\_4
  - GiNaC, [65](#)
- series\_funcp\_5
  - GiNaC, [67](#)
- series\_funcp\_6
  - GiNaC, [68](#)
- series\_funcp\_7
  - GiNaC, [69](#)
- series\_funcp\_8
  - GiNaC, [71](#)
- series\_funcp\_9
  - GiNaC, [72](#)
- series\_funcp\_exvector
  - GiNaC, [80](#)
- series\_to\_poly
  - GiNaC, [234](#)
- series\_use\_exvector\_args
  - GiNaC::function\_options, [650](#)
- series\_vec
  - GiNaC::integration\_kernel, [707](#)
- set
  - GiNaC::matrix, [749](#)
- set\_cache\_step
  - GiNaC::integration\_kernel, [706](#)
- set\_name
  - GiNaC::function\_options, [616](#)
  - GiNaC::symbol, [1090](#)
- set\_print\_context
  - GiNaC, [230](#)
- set\_print\_func
  - GiNaC, [235](#)
  - GiNaC::function\_options, [644](#)
  - GiNaC::registered\_class\_options, [988](#)
- set\_print\_options
  - GiNaC, [230](#)

- set\_refcount
  - GiNaC::refcounted, 985
- set\_return\_type
  - GiNaC::function\_options, 643
- set\_symmetry
  - GiNaC::function\_options, 643
- set\_TeX\_name
  - GiNaC::symbol, 1090
- set\_type
  - GiNaC::symmetry, 1101
- setflag
  - GiNaC::basic, 338
- shaker\_sort
  - GiNaC, 250
- share
  - GiNaC::ex, 541
- shift\_exponents
  - GiNaC::pseries, 968
- show\_statistics
  - GiNaC::remember\_table, 1003
- simplify\_indexed
  - GiNaC, 107, 121
  - GiNaC::ex, 535
  - GiNaC::indexed, 686
- simplify\_indexed\_product
  - GiNaC, 121
  - GiNaC::indexed, 686
- sin
  - GiNaC, 205
- sin\_conjugate
  - GiNaC, 160
- sin\_deriv
  - GiNaC, 160
- sin\_eval
  - GiNaC, 160
- sin\_evalf
  - GiNaC, 159
- sin\_imag\_part
  - GiNaC, 160
- sin\_real\_part
  - GiNaC, 160
- sinh
  - GiNaC, 207
- sinh\_conjugate
  - GiNaC, 168
- sinh\_deriv
  - GiNaC, 167
- sinh\_eval
  - GiNaC, 167
- sinh\_evalf
  - GiNaC, 167
- sinh\_imag\_part
  - GiNaC, 168
- sinh\_real\_part
  - GiNaC, 168
- size
  - GiNaC::basic\_multi\_iterator< T >, 349
- smod
  - GiNaC, 214
- GiNaC::add, 288
- GiNaC::basic, 330
- GiNaC::ex, 534
- GiNaC::mul, 786
- GiNaC::numeric, 857
- GiNaC::structure< T, ComparisonPolicy >, 1049
- solve
  - GiNaC::matrix, 752
- sort
  - GiNaC::container< C >, 429
- sort\_
  - GiNaC::container< C >, 428
- spinidx
  - GiNaC::spinidx, 1023
- spinor\_metric
  - GiNaC, 246
- split\_ex\_to\_pair
  - GiNaC::add, 291
  - GiNaC::expairseq, 561
  - GiNaC::mul, 788
- spm
  - GiNaC::scalar\_products, 1014
- spmap
  - GiNaC, 81
- spmapkey
  - GiNaC::spmapkey, 1034
- sprem
  - GiNaC, 191
- sqrfree
  - GiNaC, 198
- sqrfree\_parfrac
  - GiNaC, 198
- sqrfree\_yun
  - GiNaC, 197
- sqrt
  - GiNaC, 217, 233
- sr\_gcd
  - GiNaC, 193
- STATISTICS
  - normal.cpp, 1247
- std, 272
  - swap, 272
- std::equal\_to< GiNaC::ex >, 504
  - operator(), 504
- std::hash< GiNaC::ex >, 655
  - operator(), 656
- std::less< GiNaC::ptr< T > >, 729
  - operator(), 729
- std::less< ptr< T > >
  - GiNaC::ptr< T >, 975
- step
  - GiNaC, 219
  - GiNaC::numeric, 863
- step\_conjugate
  - GiNaC, 130
- step\_eval
  - GiNaC, 130

- step\_evalf
  - GiNaC, [130](#)
- step\_imag\_part
  - GiNaC, [130](#)
- step\_real\_part
  - GiNaC, [130](#)
- step\_series
  - GiNaC, [130](#)
- STLT
  - GiNaC::container< C >, [422](#)
  - GiNaC::container\_storage< C >, [433](#)
- store\_remember\_table
  - GiNaC::function, [607](#)
- struct\_compare
  - GiNaC::compare\_all\_equal< T >, [389](#)
  - GiNaC::compare\_bitwise< T >, [390](#)
  - GiNaC::compare\_std\_less< T >, [391](#)
- struct\_is\_equal
  - GiNaC::compare\_all\_equal< T >, [389](#)
  - GiNaC::compare\_bitwise< T >, [390](#)
  - GiNaC::compare\_std\_less< T >, [391](#)
- structure
  - GiNaC::structure< T, ComparisonPolicy >, [1041](#)
- structure.h, [1270](#)
- sub
  - GiNaC::matrix, [747](#)
  - GiNaC::numeric, [860](#)
- sub\_dyn
  - GiNaC::numeric, [861](#)
- sub\_matrix
  - GiNaC, [182](#)
- subs
  - GiNaC, [110](#)
  - GiNaC::basic, [326](#)
  - GiNaC::clifford, [369](#)
  - GiNaC::container< C >, [425](#)
  - GiNaC::ex, [524](#), [525](#)
  - GiNaC::expairseq, [558](#)
  - GiNaC::idx, [662](#)
  - GiNaC::matrix, [743](#)
  - GiNaC::numeric, [856](#)
  - GiNaC::power, [916](#)
  - GiNaC::pseries, [962](#)
  - GiNaC::relational, [996](#)
  - GiNaC::structure< T, ComparisonPolicy >, [1045](#)
  - GiNaC::symbol, [1087](#)
- subs\_algebraic
  - GiNaC::subs\_options, [1076](#)
- subs\_no\_pattern
  - GiNaC::subs\_options, [1076](#)
- subs\_one\_level
  - GiNaC::basic, [336](#)
- subchildren
  - GiNaC::container< C >, [431](#)
  - GiNaC::expairseq, [567](#)
- subvalue
  - GiNaC, [173](#)
- successful\_hits
  - GiNaC::remember\_table\_entry, [1007](#)
- sufficiently\_accurate
  - GiNaC::lanczos\_coeffs, [728](#)
- suppress\_branchcut
  - GiNaC::series\_options, [1015](#)
- swap
  - GiNaC, [109](#), [114](#)
  - GiNaC::ex, [517](#)
  - GiNaC::expair, [547](#)
  - GiNaC::ptr< T >, [974](#)
  - std, [272](#)
- swapped
  - GiNaC::sy\_swap, [1078](#)
- sy\_anti
  - GiNaC, [242](#)
- sy\_cycl
  - GiNaC, [242](#), [243](#)
- sy\_is\_less
  - GiNaC::sy\_is\_less, [1076](#)
  - GiNaC::symmetry, [1102](#)
- sy\_none
  - GiNaC, [240](#), [241](#)
- sy\_swap
  - GiNaC::sy\_swap, [1077](#)
  - GiNaC::symmetry, [1102](#)
- sy\_symm
  - GiNaC, [241](#), [242](#)
- sym
  - GiNaC::sym\_desc, [1080](#)
- sym\_desc
  - GiNaC::sym\_desc, [1079](#)
- sym\_desc\_vec
  - GiNaC, [81](#)
- symbol
  - GiNaC::info\_flags, [688](#)
  - GiNaC::symbol, [1085](#)
- symbol.cpp, [1271](#)
- symbol.h, [1271](#)
- symbolic\_matrix
  - GiNaC, [182](#), [185](#)
- symbolset
  - GiNaC::symbolset, [1093](#)
- symm
  - GiNaC, [239](#)
  - GiNaC::terminfo, [1132](#)
- symmetric
  - GiNaC::symmetry, [1099](#)
- symmetric2
  - GiNaC, [238](#)
- symmetric3
  - GiNaC, [238](#)
- symmetric4
  - GiNaC, [238](#)
- symmetrize
  - GiNaC, [108](#), [240](#), [243](#)
  - GiNaC::ex, [537](#)
- symmetrize\_cyclic
  - GiNaC, [108](#), [240](#), [243](#)

- GiNaC::ex, [538](#)
- symmetry
  - GiNaC::symmetry, [1099](#)
- symmetry.cpp, [1272](#)
- symmetry.h, [1273](#)
- symmetry\_type
  - GiNaC::symmetry, [1099](#)
- symminfo
  - GiNaC::symminfo, [1105](#)
- symmterm
  - GiNaC::symminfo, [1105](#)
- syms
  - factor.cpp, [1198](#)
- syms\_wox
  - factor.cpp, [1197](#)
- symtree
  - GiNaC::function\_options, [650](#)
  - GiNaC::indexed, [687](#)
- synthesize\_func
  - GiNaC, [58](#)
- table\_size
  - GiNaC::remember\_table, [1004](#)
- tan
  - GiNaC, [205](#)
- tan\_conjugate
  - GiNaC, [163](#)
- tan\_deriv
  - GiNaC, [162](#)
- tan\_eval
  - GiNaC, [162](#)
- tan\_evalf
  - GiNaC, [162](#)
- tan\_imag\_part
  - GiNaC, [163](#)
- tan\_real\_part
  - GiNaC, [162](#)
- tan\_series
  - GiNaC, [163](#)
- tanh
  - GiNaC, [208](#)
- tanh\_conjugate
  - GiNaC, [170](#)
- tanh\_deriv
  - GiNaC, [170](#)
- tanh\_eval
  - GiNaC, [170](#)
- tanh\_evalf
  - GiNaC, [169](#)
- tanh\_imag\_part
  - GiNaC, [170](#)
- tanh\_real\_part
  - GiNaC, [170](#)
- tanh\_series
  - GiNaC, [170](#)
- tau
  - GiNaC::Kronecker\_dz\_kernel, [726](#)
- tcoeff
  - GiNaC::ex, [527](#)
- tensepsilon
  - GiNaC::tensepsilon, [1117](#)
- tensor
  - GiNaC, [255](#)
- tensor.cpp, [1275](#)
- tensor.h, [1276](#)
- terminfo
  - GiNaC::terminfo, [1132](#)
- test
  - GiNaC::has\_distance< T >, [654](#)
- test\_and\_set\_nparams
  - GiNaC::function\_options, [644](#)
- TEST\_PERMUTATION
  - color.cpp, [1177](#)
- TeX\_name
  - GiNaC::constant, [415](#)
  - GiNaC::function\_options, [645](#)
  - GiNaC::symbol, [1092](#)
- tgamma
  - GiNaC, [211](#)
- tgamma\_conjugate
  - GiNaC, [147](#)
- tgamma\_deriv
  - GiNaC, [147](#)
- tgamma\_eval
  - GiNaC, [147](#)
- tgamma\_evalf
  - GiNaC, [147](#)
- tgamma\_series
  - GiNaC, [147](#)
- thiscontainer
  - GiNaC::clifford, [367](#)
  - GiNaC::color, [387](#)
  - GiNaC::container< C >, [427](#)
  - GiNaC::fderivative, [586](#)
  - GiNaC::function, [603](#)
  - GiNaC::indexed, [683](#)
  - GiNaC::ncmul, [841](#)
- thisexpairseq
  - GiNaC::add, [291](#)
  - GiNaC::expairseq, [560](#), [561](#)
  - GiNaC::mul, [788](#)
- tinfo
  - GiNaC::return\_type\_t, [1011](#)
- tinfo\_key
  - GiNaC::registered\_class\_options, [988](#)
- to\_cl\_N
  - GiNaC::numeric, [869](#)
- to\_double
  - GiNaC, [222](#)
  - GiNaC::numeric, [869](#)
- to\_int
  - GiNaC, [222](#)
  - GiNaC::numeric, [868](#)
- to\_long
  - GiNaC, [222](#)
  - GiNaC::numeric, [869](#)
- to\_polynomial

- GiNaC, 106
- GiNaC::basic, 330
- GiNaC::ex, 530
- GiNaC::expairseq, 558
- GiNaC::numeric, 857
- GiNaC::power, 917
- GiNaC::structure< T, ComparisonPolicy >, 1049
- GiNaC::symbol, 1088
- to\_rational
  - GiNaC, 105
  - GiNaC::basic, 329
  - GiNaC::ex, 529
  - GiNaC::expairseq, 557
  - GiNaC::numeric, 856
  - GiNaC::power, 917
  - GiNaC::structure< T, ComparisonPolicy >, 1049
  - GiNaC::symbol, 1087
- toggle\_dot
  - GiNaC::spinidx, 1025
- toggle\_variance
  - GiNaC::varidx, 1151
- toggle\_variance\_dot
  - GiNaC::spinidx, 1025
- too\_late
  - GiNaC::\_numeric\_digits, 277
- trace
  - GiNaC, 184
  - GiNaC::matrix, 750
- trace\_string
  - GiNaC, 89
- transpose
  - GiNaC, 183
  - GiNaC::matrix, 749
- traverse
  - GiNaC::ex, 526
- traverse\_postorder
  - GiNaC::ex, 526
- traverse\_preorder
  - GiNaC::ex, 525
- tree
  - GiNaC, 231
- tree\_node
  - GiNaC::class\_info< OPT >::tree\_node, 1134
- trig\_info
  - GiNaC, 160
- trivial
  - GiNaC::composition\_generator, 393
- tryfactsubs
  - GiNaC, 185
- type
  - GiNaC::archive\_node::property, 952
  - GiNaC::archive\_node::property\_info, 953
  - GiNaC::symmetry, 1103
- uintvector
  - GiNaC, 81
- unarch\_map
  - GiNaC::unarchive\_table\_t, 1135
- unarch\_table\_instance
  - GiNaC, 254
- unarchive
  - GiNaC::archive\_node, 309
- unarchive\_ex
  - GiNaC::archive, 297
- unarchive\_map\_t
  - GiNaC, 58
- unarchive\_table\_t
  - GiNaC::unarchive\_table\_t, 1135
- unatomize
  - GiNaC::archive, 300
- unique
  - GiNaC::container< C >, 429
- unique\_
  - GiNaC::container< C >, 428, 431
- unit
  - factor.cpp, 1197
  - GiNaC::ex, 531
- unit\_matrix
  - GiNaC, 181, 185
- unitcontprim
  - GiNaC::ex, 534
- unlikely
  - compiler.h, 1179
- unlink\_ex
  - GiNaC, 113
- unsignedvector
  - GiNaC, 81
- USE\_REMEMBER
  - normal.cpp, 1246
- use\_remember
  - GiNaC::function\_options, 648
- use\_return\_type
  - GiNaC::function\_options, 648
- USE\_SAME\_DEGREE\_FACTOR
  - factor.cpp, 1192
- use\_sr\_gcd
  - GiNaC::gcd\_options, 652
- USE\_TRIAL\_DIVISION
  - normal.cpp, 1247
- usecount
  - GiNaC::unarchive\_table\_t, 1135
- used\_indices
  - GiNaC::make\_flat\_inserter, 733
- user\_defined\_kernel
  - GiNaC::user\_defined\_kernel, 1142
- uses\_Laurent\_series
  - GiNaC::Eisenstein\_h\_kernel, 482
  - GiNaC::Eisenstein\_kernel, 493
  - GiNaC::integration\_kernel, 706
  - GiNaC::modular\_form\_kernel, 772
  - GiNaC::user\_defined\_kernel, 1143
- utils.cpp, 1277
- utils.h, 1279
  - DEFAULT\_COMPARE, 1281
  - DEFAULT\_CTOR, 1281
  - DEFAULT\_PRINT, 1281
  - DEFAULT\_PRINT\_LATEX, 1281

- utils\_multi\_iterator.h, [1281](#)
- v
  - GiNaC::basic\_multi\_iterator< T >, [351](#)
  - GiNaC::sy\_is\_less, [1077](#)
  - GiNaC::sy\_swap, [1078](#)
- v1
  - GiNaC::spmapkey, [1035](#)
- v2
  - GiNaC::spmapkey, [1035](#)
- v\_internal
  - GiNaC::multi\_iterator\_shuffle< T >, [819](#)
- v\_orig
  - GiNaC::multi\_iterator\_shuffle< T >, [820](#)
- validate
  - GiNaC::indexed, [686](#)
  - GiNaC::symmetry, [1101](#)
- value
  - factor.cpp, [1192](#)
  - GiNaC::archive\_node::property, [952](#)
  - GiNaC::composition\_generator::coolmulti::element, [495](#)
  - GiNaC::has\_distance< T >, [654](#)
  - GiNaC::idx, [667](#)
  - GiNaC::numeric, [872](#)
- value\_type
  - GiNaC::const\_iterator, [395](#)
  - GiNaC::const\_postorder\_iterator, [400](#)
  - GiNaC::const\_preorder\_iterator, [403](#)
- var
  - GiNaC::pseries, [969](#)
- varidx
  - GiNaC::varidx, [1149](#)
- version.h, [1283](#)
  - GINAC\_LT\_AGE, [1284](#)
  - GINAC\_LT\_CURRENT, [1284](#)
  - GINAC\_LT\_REVISION, [1284](#)
  - GINACLIB\_ARCHIVE\_AGE, [1284](#)
  - GINACLIB\_ARCHIVE\_VERSION, [1284](#)
  - GINACLIB\_MAJOR\_VERSION, [1284](#)
  - GINACLIB\_MICRO\_VERSION, [1284](#)
  - GINACLIB\_MINOR\_VERSION, [1284](#)
  - GINACLIB\_STR, [1284](#)
  - GINACLIB\_STR\_HELPER, [1284](#)
  - GINACLIB\_VERSION, [1284](#)
- version\_major
  - GiNaC, [257](#)
- version\_micro
  - GiNaC, [257](#)
- version\_minor
  - GiNaC, [257](#)
- vn
  - factor.cpp, [1198](#)
- vnlst
  - factor.cpp, [1198](#)
- wild
  - GiNaC, [254](#)
- wildcard
  - GiNaC::wildcard, [1157](#)
  - wildcard.cpp, [1285](#)
  - wildcard.h, [1285](#)
  - write\_real\_float
    - GiNaC, [202](#)
  - write\_unsigned
    - GiNaC, [83](#)
- x
  - factor.cpp, [1196](#)
  - GiNaC::basic\_partition\_generator::mpartition2, [774](#)
  - GiNaC::Ebar\_kernel, [474](#)
  - GiNaC::ELi\_kernel, [503](#)
  - GiNaC::integral, [699](#)
  - GiNaC::user\_defined\_kernel, [1143](#)
  - integration\_kernel.cpp, [1236](#)
- y
  - GiNaC::Ebar\_kernel, [475](#)
  - GiNaC::ELi\_kernel, [504](#)
- yes\_type
  - GiNaC::has\_distance< T >, [654](#)
- z
  - GiNaC::Kronecker\_dtau\_kernel, [718](#)
  - GiNaC::multiple\_polylog\_kernel, [831](#)
- z\_j
  - GiNaC::Kronecker\_dz\_kernel, [726](#)
- zeta
  - GiNaC, [140](#), [210](#)
- zeta1\_deriv
  - GiNaC, [155](#)
- zeta1\_eval
  - GiNaC, [155](#)
- zeta1\_evalf
  - GiNaC, [155](#)
- zeta1\_print\_latex
  - GiNaC, [155](#)
- zeta2\_deriv
  - GiNaC, [156](#)
- zeta2\_eval
  - GiNaC, [155](#)
- zeta2\_evalf
  - GiNaC, [155](#)
- zeta2\_print\_latex
  - GiNaC, [156](#)
- zetaderiv\_deriv
  - GiNaC, [135](#)
- zetaderiv\_eval
  - GiNaC, [135](#)