GiNaC

1.8.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 GiNaC Namespace Reference

Namespaces

- 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 anticommutates with all other gammas.
- class diracgammaL

  This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).
- class diracgammaR

  This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).
- 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...
- 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 $(i_1, i_2, ..., i_k)$, such that

  $$B \leq i_j < N$$

  .
- class multi_iterator_counter_indv

  The class multi_iterator_counter_indv defines a multi_iterator $(i_1, i_2, ..., i_k)$, such that

  $$B \leq i_j < N_j$$

  .
- class multi_iterator_ordered

  The class multi_iterator_ordered defines a multi_iterator $(i_1, i_2, ..., i_k)$, such that

  $$B \leq i_j < N$$

  and

  $$i_j < i_{j+1}.$$

  It is assumed that $k > 0$ and $N - B \geq k$.
- class multi_iterator_ordered_eq

  The class multi_iterator_ordered_eq defines a multi_iterator $(i_1, i_2, ..., i_k)$, such that

  $$B \leq i_j < N$$

  and

  $$i_j \leq i_{j+1}.$$

  It is assumed that $k > 0$.

- class multi_iterator_ordered_eq_indv

  The class multi_iterator_ordered_eq_indv defines a multi_iterator $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N_j$$

  and

  $$i_j \leq i_{j+1}.$$

  .
- class multi_iterator_permutation

  The class multi_iterator_permutation defines a multi_iterator $(i_1, i_2, ..., i_k)$*, 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.*
- 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 and 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 spmapkey
- 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(∗ evalffunctype) ()
- 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_funcp) ()
- typedef ex(∗ evalf_funcp) ()
- typedef ex(∗ conjugate_funcp) ()
- typedef ex(∗ real_part_funcp) ()
- typedef ex(∗ imag_part_funcp) ()
- typedef ex(∗ expand_funcp) ()
- typedef ex(∗ derivative_funcp) ()
- typedef ex(∗ expl_derivative_funcp) ()
- typedef ex(∗ power_funcp) ()
- typedef ex(∗ series_funcp) ()

- typedef void(∗ print_funcp) ()
- typedef bool(∗ info_funcp) ()
- typedef ex(∗ eval_funcp_1) (const ex &)
- typedef ex(∗ evalf_funcp_1) (const ex &)
- typedef ex(∗ conjugate_funcp_1) (const ex &)
- typedef ex(∗ real_part_funcp_1) (const ex &)
- typedef ex(∗ imag_part_funcp_1) (const ex &)
- typedef ex(∗ expand_funcp_1) (const ex &, unsigned)
- typedef ex(∗ derivative_funcp_1) (const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_1) (const ex &, const symbol &)
- typedef ex(∗ power_funcp_1) (const ex &, const ex &)
- typedef ex(∗ series_funcp_1) (const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_1) (const ex &, const print_context &)
- typedef bool(∗ info_funcp_1) (const ex &, unsigned)
- typedef ex(∗ eval_funcp_2) (const ex &, const ex &)
- typedef ex(∗ evalf_funcp_2) (const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_2) (const ex &, const ex &)
- typedef ex(∗ real_part_funcp_2) (const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_2) (const ex &, const ex &)
- typedef ex(∗ expand_funcp_2) (const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_2) (const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_2) (const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_2) (const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_2) (const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_2) (const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_3) (const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_3) (const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)

- typedef ex(∗ expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)

- typedef ex(∗ conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ series_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ print_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ expl_derivative_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ power_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ 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)
- typedef void(∗ print_funcp_9) (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_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ eval_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ evalf_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ conjugate_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ real_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ imag_part_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ expand_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)

- typedef ex(∗ derivative_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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)
- typedef void(∗ 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 &)
- typedef bool(∗ info_funcp_10) (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_11) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ expand_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_11) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ 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 symbol &)
- typedef ex(∗ 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 &)
- typedef ex(∗ series_funcp_11) (const ex &, 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_11) (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_11) (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_12) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ expand_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ derivative_funcp_12) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ 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 symbol &)
- typedef ex(∗ 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 &)
- typedef ex(∗ series_funcp_12) (const ex &, 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_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 &)
- typedef bool(∗ 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 &, unsigned)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &, unsigned)
- typedef ex(∗ 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 &, unsigned)
- typedef ex(∗ 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 symbol &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 relational &, int, unsigned)
- typedef void(∗ 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 print_context &)
- typedef bool(∗ 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 &, unsigned)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &)
- typedef ex(∗ 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 &, unsigned)
- typedef ex(∗ 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 &, 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 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 &, 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 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 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 &, 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 tinfo_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)) GI↩ NAC_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~mu -> (1, mu) and a\ -> (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~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 &rll, 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)
- static ex get_clifford_comp (const ex &e, const ex &c, bool root=true)

  Auxiliary structure to define a function for striping 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_O↩ PT(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+I*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 &rll)

  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, 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 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 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 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 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 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 unlink_ex (const std::string filename)

  Closes all linked .so files that have the supplied filename.

- void swap (expair &e1, expair &e2)
- 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↩ _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_float). 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("\ta"). 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("\athrm{Li}_2"))
- static ex Li3_eval (const ex &x)
- REGISTER_FUNCTION (Li3, eval_func(Li3_eval). latex_name("\athrm{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("\eta^\rime"))
- 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_expl_derivative (const ex &arg, const symbol &s)
- REGISTER_FUNCTION (Order, eval_func(Order_eval). series_func(Order_series). latex_name("\athcal{O}"). expl_derivative_func(Order_expl_derivative). conjugate_func(Order_conjugate). real_part_func(Order_real_part). imag_part_func(Order_imag_part))
- ex lsolve (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("\og \amma"))

- 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("\amma"))
- 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("\athrm{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)
- REGISTER_FUNCTION (exp, eval_func(exp_eval). evalf_func(exp_evalf). 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("\xp"))
- 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)
- REGISTER_FUNCTION (log, eval_func(log_eval). evalf_func(log_evalf). 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("\n"))
- 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)
- REGISTER_FUNCTION (sin, eval_func(sin_eval). evalf_func(sin_evalf). derivative_func(sin_deriv). real↩ _part_func(sin_real_part). imag_part_func(sin_imag_part). conjugate_func(sin_conjugate). latex_↩ name("\in"))
- 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). 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("\os"))
- 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). 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("\an"))
- 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)
- REGISTER_FUNCTION (asin, eval_func(asin_eval). evalf_func(asin_evalf). derivative_func(asin_deriv). conjugate_func(asin_conjugate). latex_name("\rcsin"))
- 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). derivative_func(acos_deriv). conjugate_func(acos_conjugate). latex_name("\rccos"))
- 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)
- REGISTER_FUNCTION (atan, eval_func(atan_eval). evalf_func(atan_evalf). derivative_func(atan_deriv). series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\rctan"))
- 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)
- REGISTER_FUNCTION (atan2, eval_func(atan2_eval). 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). derivative_func(sinh_deriv). real_part_func(sinh_real_part). imag_part_func(sinh_imag_part). conjugate_func(sinh_conjugate). latex_name("\inh"))
- 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). derivative_func(cosh_deriv). real_part_func(cosh_real_part). imag_part_func(cosh_imag_part). conjugate_func(cosh_conjugate). latex_name("\osh"))
- 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). 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("\anh"))
- 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). 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). 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). 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(lst(p1,...,pr), lst(a1,...,ar)) such that n = p1^a1 ...
- 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 &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_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 e1,e2 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_idecoded_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_O↩ PT(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!/(p1!*p2!*...*pk!) where n = p1+p2+...+pk, 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, "\i", domain::positive)

  Pi.
- const constant Euler ("Euler", EulerEvalf, "\amma_E", domain::positive)

  Euler's constant.
- const constant Catalan ("Catalan", CatalanEvalf, "G", domain::positive)

  Catalan's constant.
- static unsigned const crctab [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
- 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_120
- const numeric * _num_60_p
- const ex _ex_60 = _ex_60
- const numeric * _num_48_p
- const ex _ex_48 = _ex_48
- const numeric * _num_30_p
- const ex _ex_30 = _ex_30
- const numeric * _num_25_p
- const ex _ex_25 = _ex_25
- const numeric * _num_24_p
- const ex _ex_24 = _ex_24
- const numeric * _num_20_p
- const ex _ex_20 = _ex_20

- const numeric * _num_18_p
- const ex _ex_18 = _ex_18
- const numeric * _num_15_p
- const ex _ex_15 = _ex_15
- const numeric * _num_12_p
- const ex _ex_12 = _ex_12
- const numeric * _num_11_p
- const ex _ex_11 = _ex_11
- const numeric * _num_10_p
- const ex _ex_10 = _ex_10
- const numeric * _num_9_p
- const ex _ex_9 = _ex_9
- const numeric * _num_8_p
- const ex _ex_8 = _ex_8
- const numeric * _num_7_p
- const ex _ex_7 = _ex_7
- const numeric * _num_6_p
- const ex _ex_6 = _ex_6
- const numeric * _num_5_p
- const ex _ex_5 = _ex_5
- const numeric * _num_4_p
- const ex _ex_4 = _ex_4
- const numeric * _num_3_p
- const ex _ex_3 = _ex_3
- const numeric * _num_2_p
- const ex _ex_2 = _ex_2
- const numeric * _num_1_p
- const ex _ex_1 = _ex_1
- const numeric * _num_1_2_p
- const ex _ex_1_2 = _ex_1_2
- const numeric * _num_1_3_p
- const ex _ex_1_3 = _ex_1_3
- const numeric * _num_1_4_p
- const ex _ex_1_4 = _ex_1_4
- const numeric * _num0_p
- const ex _ex0 = _ex0
- const numeric * _num1_4_p
- const ex _ex1_4 = _ex1_4
- const numeric * _num1_3_p
- const ex _ex1_3 = _ex1_3
- const numeric * _num1_2_p
- const ex _ex1_2 = _ex1_2
- const numeric * _num1_p
- const ex _ex1 = _ex1
- const numeric * _num2_p
- const ex _ex2 = _ex2
- const numeric * _num3_p
- const ex _ex3 = _ex3
- const numeric * _num4_p
- const ex _ex4 = _ex4
- const numeric * _num5_p
- const ex _ex5 = _ex5
- const numeric * _num6_p
- const ex _ex6 = _ex6
- const numeric * _num7_p

- const ex _ex7 = _ex7
- const numeric * _num8_p
- const ex _ex8 = _ex8
- const numeric * _num9_p
- const ex _ex9 = _ex9
- const numeric * _num10_p
- const ex _ex10 = _ex10
- const numeric * _num11_p
- const ex _ex11 = _ex11
- const numeric * _num12_p
- const ex _ex12 = _ex12
- const numeric * _num15_p
- const ex _ex15 = _ex15
- const numeric * _num18_p
- const ex _ex18 = _ex18
- const numeric * _num20_p
- const ex _ex20 = _ex20
- const numeric * _num24_p
- const ex _ex24 = _ex24
- const numeric * _num25_p
- const ex _ex25 = _ex25
- const numeric * _num30_p
- const ex _ex30 = _ex30
- const numeric * _num48_p
- const ex _ex48 = _ex48
- const numeric * _num60_p
- const ex _ex60 = _ex60
- const numeric * _num120_p
- const ex _ex120 = _ex120

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 &, 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 &, 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 &, 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 &,
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 &,
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 &,
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 &, 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
&, 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 &, 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
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 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 &, 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 &, 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 &, 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 &, 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 &, 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 = typedef 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() [1/32]

```
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.

Referenced by operator<<().

5.1.3.5 read_unsigned()

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

Read unsigned integer quantity from stream.

Referenced by operator>>().

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.

References n, and write_unsigned().

5.1.3.7 operator<<() [2/16]

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

Write archive to binary data stream.

References GiNaC::archive::atoms, GiNaC::archive::exprs, GINACLIB_ARCHIVE_VERSION, GiNaC::archive::nodes, and write_unsigned().

5.1.3.8 operator>>() [1/3]

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

Read archive_node from binary data stream.

References n, and read_unsigned().

5.1.3.9 operator>>() [2/3]

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

Read archive from binary data stream.

References GiNaC::archive::atoms, GiNaC::archive::exprs, GINACLIB_ARCHIVE_AGE, GINACLIB_ARCHIVE_VERSION, GiNaC::archive::inverse_atoms, GiNaC::archive::nodes, and read_unsigned().

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() [2/32]

```
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() [3/32]

```
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 -> (1, mu) and a\ -> (a.ix, ix)

5.1.3.27 dirac_ONE()

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

Create a Clifford unity object.

Parameters

rl	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

mu	Index (must be of class varidx or a derived class)
metr	Metric (should be indexed, tensmetric or a derived class, or a matrix)
rl	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

mu	Index (must be of class varidx or a derived class)
rl	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

rl	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

rl	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

rl	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

e	Original expression
dim	Dimension of index
rl	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 != 4 dimensions when gamma5 is involved.

Parameters

e	Expression to take the trace of
rls	Set of representation labels
trONE	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 != 4 dimensions when gamma5 is involved.

Parameters

e	Expression to take the trace of
rll	List of representation labels
trONE	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 != 4 dimensions when gamma5 is involved.

Parameters

e	Expression to take the trace of
rl	Representation label
trONE	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

e	Expression to be processed
rl	Value of representation label
options	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

e	Expression to be processed defines if clifford_ONE 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

v	List or vector of coordinates
mu	Index (must be of class varidx or a derived class)
metr	Metric (should be indexed, tensmetric or a derived class, or a matrix)
rl	Representation label
e	Clifford unit object

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 )
```

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 striping 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)/ pow(e.i, 2) for non-degenerate cases (i.e. neither pow(e.i, 2) = 0).

Parameters

e	Clifford expression to be decomposed into components
c	Clifford unit defining the metric for splitting (should have numeric dimension of indices)
algebraic	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

ti	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() [4/32]

```
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

iv3	Vector of 3 indices
iv2	Vector of 2 indices, must be a subset of iv3
sig	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

rl	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

a	Index
rl	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

a	First index
b	Second index
c	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

a	First index
b	Second index
c	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+I∗f.a.b.c.

References c, color_d(), color_f(), and I.

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

e	Expression to take the trace of
rls	Set of representation labels

References _ex0, _ex1, _ex3, color_h(), color_T(), 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(), 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

e	Expression to take the trace of
rll	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

e	Expression to take the trace of
rl	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() [5/32]

```
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.

References GiNaC::ex::bp.

Referenced by GiNaC::integral::conjugate(), GiNaC::pseries::conjugate(), GiNaC::matrix::conjugate(), GiNaC::add::conjugate(), GiNaC::mul::conjugate(), GiNaC::power::conjugate(), GiNaC::expair::conjugate(), GiNaC::container< C >::conjugate(), GiNaC::pseries::eval_integ(), GiNaC::integral::evalf(), GiNaC::pseries::evalm(), GiNaC::integral::expand(), GiNaC::power::expand(), GiNaC::indexed::expand(), GiNaC::ncmul::expandchildren(), GiNaC::mul::expandchildren(), GiNaC::make_flat_inserter::make_flat_inserter(), GiNaC::idx::map(), GiNaC::power::map(), GiNaC::relational::map(), GiNaC::basic::map(), GiNaC::const_iterator::operator==(), GiNaC::internal::_iter_rep::operator==(), GiNaC::idx::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::basic::subs(), and GiNaC::container< C >::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::mul::algebraic_subs_mul(), GiNaC::basic::calchash(), GiNaC::basic::derivative(), GiNaC::mul::do_print_python_repr(), GiNaC::container< C >::do_print_tree(), GiNaC::basic::do_print_tree(), GiNaC::basic::eval_integ(), GiNaC::basic::evalf(), GiNaC::basic::evalm(), GiNaC::basic::expand(), GiNaC::mul::has(), GiNaC::basic::has(), GiNaC::container< C >::let_op(), GiNaC::basic::map(), GiNaC::basic::match(), GiNaC::clifford::nops(), 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(), GiNaC::indexed::expand(), GiNaC::structure< T, ComparisonPolicy >::expand(), GiNaC::function::expand(), 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 GiNaC::numeric::conjugate(), conjugate_evalf(), and 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::mul::has(), GiNaC::structure< T, ComparisonPolicy >::has(), and 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(), GiNaC::structure< T, ComparisonPolicy >::degree(), and replace_with_symbol().

5.1.3.103 ldegree()

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

References GiNaC::ex::ldegree().

Referenced by GiNaC::basic::collect(), and GiNaC::structure< T, ComparisonPolicy >::ldegree().

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::mul::coeff(), GiNaC::structure< T, ComparisonPolicy >::coeff(), GiNaC::basic::collect(), GiNaC::mul::eval(), GiNaC::make_flat_inserter::handle_factor(), iterated_integral_evalf_impl(), log_series(), simplify_indexed(), 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::add::do_print_csrc(), 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::do_print_csrc(), GiNaC::add::integer_content(), 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::structure< T, ComparisonPolicy >::normal(), 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().

Referenced by GiNaC::structure< T, ComparisonPolicy >::to_rational().

5.1.3.110 to_polynomial()

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

References GiNaC::ex::to_polynomial().

Referenced by GiNaC::structure< T, ComparisonPolicy >::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::structure< T, ComparisonPolicy >::collect(), and 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_eval(), eta_evalf(), H_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::structure< T, ComparisonPolicy >::evalm(), and 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(), lgamma_series(), Li2_series(), log_series(), psi1_series(), GiNaC::structure< T, ComparisonPolicy >::series(), and tgamma_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(), GiNaC::structure< T, ComparisonPolicy >::match(), 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(), simplify_indexed(), and simplify_indexed_product().

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(), symmetrize(), GiNaC::ex::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 antisymmetrize(), and GiNaC::ex::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().

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::mul::algebraic_subs_mul(), GiNaC::basic::calchash(), GiNaC::mul::do_print_python_repr(), GiNaC::basic::do_print_tree(), GiNaC::basic::has(), GiNaC::basic::map(), GiNaC::basic::match(), GiNaC::clifford↩ ::op(), GiNaC::structure< T, ComparisonPolicy >::op(), GiNaC::basic::operator[ ](), and GiNaC::basic::subs().

5.1.3.128 lhs()

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

References GiNaC::ex::lhs().

Referenced by std::less< GiNaC::ptr< T > >::operator()().

5.1.3.129 rhs()

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

References GiNaC::ex::rhs().

Referenced by lsolve(), GiNaC::ptr< GiNaC::basic >::operator!=(), std::less< GiNaC::ptr< T > >::operator()(), GiNaC::ptr< GiNaC::basic >::operator==(), GiNaC::matrix::solve(), and sqrfree_parfrac().

5.1.3.130 is_zero() [1/2]

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

References GiNaC::ex::is_zero().

Referenced by GiNaC::matrix::determinant(), GiNaC::matrix::determinant_minor(), GiNaC::power::eval(), GiNaC::tensepsilon::eval_indexed(), GiNaC::matrix::fraction_free_elimination(), GiNaC::matrix::gauss_elimination(), GiNaC::matrix::markowitz_elimination(), GiNaC::matrix::mul(), GiNaC::relational::operator safe_bool(), GiNaC::matrix::pivot(), GiNaC::clifford::same_metric(), and GiNaC::Kronecker_dz_kernel::series_coeff_impl().

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 GiNaC::clifford::subs(), and GiNaC::structure< T, ComparisonPolicy >::subs().

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.

References GiNaC::ex::bp.

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.

References GiNaC::ex::bp.

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

e	expression

Returns

reference to object of class T

See also

is_exactly_a<class T>()

References GiNaC::ex::bp, and GINAC_ASSERT.

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

expr	Expression to be compiled
sym	Symbol from the expression to become the function parameter
fp	Returned function pointer
filename	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

expr	Expression to be compiled
sym	Symbol from the expression to become the function parameter
fp	Returned function pointer
filename	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

expr	Expression to be compiled
sym	Symbol from the expression to become the function parameter
fp	Returned function pointer
filename	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 FUNCP_1P to the contained function.

The so-file has to be generated by compile_ex in advance.

Parameters

filename	Name of the so-file to open and link
fp	Returned function pointer

5.1.3.142 link_ex() [2/3]

```
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.

The so-file has to be generated by compile_ex in advance.

Parameters

filename	Name of the so-file to open and link
fp	Returned function pointer

5.1.3.143 link_ex() [3/3]

```
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.

The so-file has to be generated by compile_ex in advance.

Parameters

filename	Name of the so-file to open and link
fp	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

filename	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 conjugateepvector()

```
epvector* GiNaC::conjugateepvector (
             const epvector &  )
```

Complex conjugate every element of an epvector.

Returns zero if this does not change anything.

Referenced by GiNaC::pseries::conjugate().

5.1.3.147 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	*poly*	expression to factorize
in	*option*	options to influence the factorization

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 sqrfree_parfrac().

5.1.3.148 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [6/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
            fail ,
            basic ,
            print_func< print_context > &::do_print.  print_func< print_tree > &::do_print↩
_tree  )
```

5.1.3.149 GINAC_DECLARE_UNARCHIVER() [15/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            fail  )
```

5.1.3.150 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [7/32]

```
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  )
```

5.1.3.151 GINAC_BIND_UNARCHIVER() [15/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            fderivative  )
```

5.1.3.152 GINAC_DECLARE_UNARCHIVER() [16/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            fderivative  )
```

5.1.3.153 GINAC_BIND_UNARCHIVER() [16/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            function  )
```

5.1.3.154 GINAC_DECLARE_UNARCHIVER() [17/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              function  )
```

5.1.3.155 is_the_function()

```
template<typename T >
bool GiNaC::is_the_function (
              const ex & x )  [inline]
```

References x.

5.1.3.156 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::wildcard::calchash(), GiNaC::idx::calchash(), GiNaC::symbol::calchash(), GiNaC::relational::calchash(), GiNaC::symmetry::calchash(), GiNaC::basic::calchash(), and GiNaC::function::calchash().

5.1.3.157 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [8/32]

```
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  )
```

5.1.3.158 print_func< print_context >()

```
GiNaC::print_func< print_context > (
             &varidx::do_print  ) &
```

5.1.3.159 GINAC_BIND_UNARCHIVER() [17/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             idx  )
```

5.1.3.160 GINAC_BIND_UNARCHIVER() [18/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             varidx  )
```

5.1.3.161 GINAC_BIND_UNARCHIVER() [19/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             spinidx  )
```

5.1.3.162 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()(), GiNaC::tensor::replace_contr_index(), and reposition_dummy_↩indices().

5.1.3.163 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().

5.1.3.164 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

it	Pointer to start of index vector
itend	Pointer to end of index vector
out_free	Vector of free indices (returned, sorted)
out_dummy	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(), Gi↩ NaC::ncmul::get_free_indices(), GiNaC::mul::get_free_indices(), GiNaC::indexed::get_free_indices(), simplify_↩ indexed(), and simplify_indexed_product().

5.1.3.165 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(), and simplify_indexed_product().

5.1.3.166 GINAC_DECLARE_UNARCHIVER() [18/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              idx  )
```

5.1.3.167 GINAC_DECLARE_UNARCHIVER() [19/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              varidx  )
```

5.1.3.168 GINAC_DECLARE_UNARCHIVER() [20/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              spinidx  )
```

5.1.3.169 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

v	Index vector
out_free	Vector of free indices (returned, sorted)
out_dummy	Vector of dummy indices (returned, sorted)

References find_free_and_dummy().

5.1.3.170 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

v	Index vector
out_dummy	Vector of dummy indices (returned, sorted)

References find_free_and_dummy().

Referenced by GiNaC::indexed::get_dummy_indices().

5.1.3.171 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().

5.1.3.172 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().

5.1.3.173 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [9/32]

```
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  )
```

5.1.3.174 GINAC_BIND_UNARCHIVER() [20/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            indexed  )
```

5.1.3.175 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(), and simplify_indexed().

5.1.3.176 number_of_type()

```
template<class T >
size_t GiNaC::number_of_type (
            const exvector & v )
```

5.1.3.177 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

e	Expression to work on
local_dummy_indices	The set of dummy indices that appear in the expression "e"
global_dummy_indices	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, shaker_sort(), and GiNaC↩ ::ex::subs().

5.1.3.178 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.

Referenced by simplify_indexed(), and simplify_indexed_product().

5.1.3.179 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

e	Object to work on
variant_dummy_indices	The set of indices that might need repositioning (will be changed by this function)
moved_indices	The set of indices that have been repositioned (will be changed by this function)

Returns

true if 'e' was changed

References is_dummy_pair(), k, m, GiNaC::subs_options::no_pattern, GiNaC::ex::nops(), GiNaC::ex::op(), and GiNaC::ex::subs().

Referenced by simplify_indexed(), and simplify_indexed_product().

5.1.3.180 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(), and simplify_indexed_product().

5.1.3.181 idx_symmetrization()

```
template<class T >
ex GiNaC::idx_symmetrization (
            const ex & r,
            const exvector & local_dummy_indices )
```

References r, and symmetrize().

5.1.3.182 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.

References _ex2, GiNaC::ex::coeff(), coeff(), GiNaC::ex::expand(), find_free_and_dummy(), find_variant_indices(), hasindex(), indices_consistent(), GiNaC::ex::is_equal(), GiNaC::ex::is_zero(), k, GiNaC::ex::nops(), GiNaC::ex::op(), reposition_dummy_indices(), GiNaC::container_storage< C >::seq, simplify_indexed(), and simplify_indexed_product().

5.1.3.183 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.

References _ex0, _ex1, GiNaC::return_types::commutative, GiNaC::scalar_products::evaluate(), find_free_and_dummy(), find_variant_indices(), GINAC_ASSERT, GiNaC::scalar_products::is_defined(), GiNaC::ex::is_zero(), minimal_dim(), product_to_exvector(), r, reposition_dummy_indices(), and simplify_indexed().

Referenced by simplify_indexed().

5.1.3.184 hasindex()

```
bool GiNaC::hasindex (
             const ex & x,
             const ex & sym )
```

References GiNaC::ex::nops(), GiNaC::ex::op(), and x.

Referenced by simplify_indexed().

5.1.3.185 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(), GiNaC::ex::get_free_indices(), GiNaC::ex::nops(), and GiNaC::ex::op().

Referenced by GiNaC::mul::expand(), GiNaC::make_flat_inserter::handle_factor(), and rename_dummy_indices_uniquely().

5.1.3.186 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().

5.1.3.187 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::ncmul::expand(), GiNaC::mul::expand(), GiNaC::power::expand_mul(), GiNaC::make_flat_inserter::handle_factor(), and rename_dummy_indices_uniquely().

5.1.3.188 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().

5.1.3.189 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, rename_dummy_indices_uniquely(), and GiNaC::ex::subs().

5.1.3.190 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 get_all_dummy_indices_safely(), GiNaC::subs_options::no_index_renaming, GiNaC::subs_options::no_pattern, rename_dummy_indices_uniquely(), and GiNaC::ex::subs().

5.1.3.191 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

e	the given expression
subs_idx	indicates if variance of dummy indices should be neglected

References GiNaC::ex::expand(), get_all_dummy_indices(), idx, GiNaC::ex::map(), GiNaC::info_flags::nonnegint, GiNaC::ex::subs(), and to_int().

5.1.3.192 GINAC_DECLARE_UNARCHIVER() [21/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             indexed  )
```

5.1.3.193 conjugate_evalf()

```
static ex GiNaC::conjugate_evalf (
             const ex & arg )  [static]
```

References conjugate().

5.1.3.194 conjugate_eval()

```
static ex GiNaC::conjugate_eval (
             const ex & arg )  [static]
```

References GiNaC::ex::conjugate().

5.1.3.195 conjugate_print_latex()

```
static void GiNaC::conjugate_print_latex (
             const ex & arg,
             const print_context & c )  [static]
```

References c, and GiNaC::ex::print().

5.1.3.196 conjugate_conjugate()

```
static ex GiNaC::conjugate_conjugate (
             const ex & arg )  [static]
```

5.1.3.197 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.

5.1.3.198 conjugate_real_part()

```
static ex GiNaC::conjugate_real_part (
            const ex & arg )  [static]
```

References GiNaC::ex::real_part().

5.1.3.199 conjugate_imag_part()

```
static ex GiNaC::conjugate_imag_part (
            const ex & arg )  [static]
```

References GiNaC::ex::imag_part().

5.1.3.200 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().

5.1.3.201 conjugate_info()

```
static bool GiNaC::conjugate_info (
            const ex & arg,
            unsigned inf )  [static]
```

References func_arg_info().

5.1.3.202 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")  )
```

5.1.3.203 real_part_evalf()

```
static ex GiNaC::real_part_evalf (
            const ex & arg )  [static]
```

References real().

5.1.3.204 real_part_eval()

```
static ex GiNaC::real_part_eval (
            const ex & arg )  [static]
```

References GiNaC::ex::real_part().

5.1.3.205 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().

5.1.3.206 real_part_conjugate()

```
static ex GiNaC::real_part_conjugate (
            const ex & arg )  [static]
```

5.1.3.207 real_part_real_part()

```
static ex GiNaC::real_part_real_part (
            const ex & arg )  [static]
```

5.1.3.208 real_part_imag_part()

```
static ex GiNaC::real_part_imag_part (
            const ex & arg )  [static]
```

5.1.3.209 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().

5.1.3.210 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")  )
```

5.1.3.211 imag_part_evalf()

```
static ex GiNaC::imag_part_evalf (
             const ex & arg )  [static]
```

References imag().

5.1.3.212 imag_part_eval()

```
static ex GiNaC::imag_part_eval (
             const ex & arg )  [static]
```

References GiNaC::ex::imag_part().

5.1.3.213 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().

5.1.3.214 imag_part_conjugate()

```
static ex GiNaC::imag_part_conjugate (
            const ex & arg )  [static]
```

5.1.3.215 imag_part_real_part()

```
static ex GiNaC::imag_part_real_part (
            const ex & arg )  [static]
```

5.1.3.216 imag_part_imag_part()

```
static ex GiNaC::imag_part_imag_part (
            const ex & arg )  [static]
```

5.1.3.217 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.

5.1.3.218 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")  )
```

5.1.3.219 abs_evalf()

```
static ex GiNaC::abs_evalf (
            const ex & arg )  [static]
```

References abs(), and GiNaC::basic::hold().

5.1.3.220 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().

5.1.3.221 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.

5.1.3.222 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().

5.1.3.223 abs_print_latex()

```
static void GiNaC::abs_print_latex (
            const ex & arg,
            const print_context & c )  [static]
```

References c, and GiNaC::ex::print().

5.1.3.224 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().

5.1.3.225 abs_conjugate()

```
static ex GiNaC::abs_conjugate (
            const ex & arg )  [static]
```

References abs(), and GiNaC::basic::hold().

5.1.3.226 abs_real_part()

```
static ex GiNaC::abs_real_part (
            const ex & arg )  [static]
```

References abs(), and GiNaC::basic::hold().

5.1.3.227 abs_imag_part()

```
static ex GiNaC::abs_imag_part (
            const ex & arg )  [static]
```

5.1.3.228 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::numeric↩ ::info(), GiNaC::ex::info(), GiNaC::ex::is_equal(), is_even(), pow(), and GiNaC::info_flags::real.

5.1.3.229 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.

5.1.3.230 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) )
```

5.1.3.231 step_evalf()

```
static ex GiNaC::step_evalf (
            const ex & arg ) [static]
```

References GiNaC::basic::hold(), and step().

5.1.3.232 step_eval()

```
static ex GiNaC::step_eval (
            const ex & arg ) [static]
```

References GiNaC::basic::hold(), I, GiNaC::numeric::imag(), GiNaC::numeric::is_real(), GiNaC::numeric::is_zero(), GiNaC::ex::nops(), GiNaC::ex::op(), GiNaC::numeric::real(), and step().

5.1.3.233 step_series()

```
static ex GiNaC::step_series (
            const ex & arg,
            const relational & rel,
            int order,
            unsigned options ) [static]
```

References _ex0, GiNaC::ex::info(), GiNaC::numeric::is_zero(), GiNaC::subs_options::no_pattern, GiNaC::info_↩flags::numeric, options, real(), step(), GiNaC::ex::subs(), and GiNaC::series_options::suppress_branchcut.

5.1.3.234 step_conjugate()

```
static ex GiNaC::step_conjugate (
            const ex & arg ) [static]
```

References GiNaC::basic::hold(), and step().

5.1.3.235 step_real_part()

```
static ex GiNaC::step_real_part (
            const ex & arg )  [static]
```

References GiNaC::basic::hold(), and step().

5.1.3.236 step_imag_part()

```
static ex GiNaC::step_imag_part (
            const ex & arg )  [static]
```

5.1.3.237 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)  )
```

5.1.3.238 csgn_evalf()

```
static ex GiNaC::csgn_evalf (
            const ex & arg )  [static]
```

References csgn().

5.1.3.239 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().

5.1.3.240 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::numeric::is_zero(), GiNaC::subs_options::no_pattern, GiNaC::info_flags::numeric, options, real(), GiNaC::ex::subs(), and GiNaC::series_options::suppress_branchcut.

5.1.3.241 csgn_conjugate()

```
static ex GiNaC::csgn_conjugate (
             const ex & arg )  [static]
```

References csgn().

5.1.3.242 csgn_real_part()

```
static ex GiNaC::csgn_real_part (
             const ex & arg )  [static]
```

References csgn().

5.1.3.243 csgn_imag_part()

```
static ex GiNaC::csgn_imag_part (
             const ex & arg )  [static]
```

5.1.3.244 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_integer(), and GiNaC::info_flags::positive.

5.1.3.245 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)  )
```

5.1.3.246 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.

5.1.3.247 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.

5.1.3.248 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.

5.1.3.249 eta_conjugate()

```
static ex GiNaC::eta_conjugate (
            const ex & x,
            const ex & y )  [static]
```

References x.

5.1.3.250 eta_real_part()

```
static ex GiNaC::eta_real_part (
            const ex & x,
            const ex & y )  [static]
```

5.1.3.251 eta_imag_part()

```
static ex GiNaC::eta_imag_part (
            const ex & x,
            const ex & y )  [static]
```

References GiNaC::basic::hold(), I, and x.

5.1.3.252 REGISTER_FUNCTION() [7/36]

```
GiNaC::REGISTER_FUNCTION (
            eta ,
            eval_func(eta_eval).  evalf_func(eta_evalf).  series_func(eta_series).  latex_↩
name("\a").  set_symmetry(sy_symm(0, 1)).  conjugate_func(eta_conjugate).  real_part_func(eta_real_part).
imag_part_func(eta_imag_part)  )
```

5.1.3.253 Li2_evalf()

```
static ex GiNaC::Li2_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), Li2(), and x.

5.1.3.254 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(), I, GiNaC::ex::info(), GiNaC::ex::is_equal(), GiNaC::ex::is_zero(), Li2(), log(), GiNaC::info_↩
flags::numeric, Pi, and x.

5.1.3.255 Li2_deriv()

```
static ex GiNaC::Li2_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, GINAC_ASSERT, log(), and x.

5.1.3.256 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, I, 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, GiNaC::ex::op(), options, order, Pi, pow(), GiNaC::relational::rhs(), GiNaC::ex::series(), series(), GiNaC::ex::subs(), GiNaC::series_options::suppress_branchcut, x, and zeta().

Referenced by Li2_projection().

5.1.3.257 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.

5.1.3.258 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("\thrm{Li}_2")  )
```

5.1.3.259 Li3_eval()

```
static ex GiNaC::Li3_eval (
            const ex & x )  [static]
```

References GiNaC::ex::is_zero(), and x.

5.1.3.260 REGISTER_FUNCTION() [9/36]

```
GiNaC::REGISTER_FUNCTION (
             Li3 ,
             eval_func(Li3_eval).  latex_name("\thrm{Li}_3")  )
```

5.1.3.261 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().

5.1.3.262 zetaderiv_deriv()

```
static ex GiNaC::zetaderiv_deriv (
             const ex & n,
             const ex & x,
             unsigned deriv_param )  [static]
```

References GINAC_ASSERT, n, and x.

5.1.3.263 REGISTER_FUNCTION() [10/36]

```
GiNaC::REGISTER_FUNCTION (
             zetaderiv ,
             eval_func(zetaderiv_eval).  derivative_func(zetaderiv_deriv).  latex_name("\ta^\ime")
)
```

5.1.3.264 factorial_evalf()

```
static ex GiNaC::factorial_evalf (
             const ex & x )  [static]
```

References factorial(), GiNaC::basic::hold(), and x.

5.1.3.265 factorial_eval()

```
static ex GiNaC::factorial_eval (
            const ex & x )  [static]
```

References factorial(), GiNaC::basic::hold(), and x.

5.1.3.266 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.

5.1.3.267 factorial_conjugate()

```
static ex GiNaC::factorial_conjugate (
            const ex & x )  [static]
```

References factorial(), GiNaC::basic::hold(), and x.

5.1.3.268 factorial_real_part()

```
static ex GiNaC::factorial_real_part (
            const ex & x )  [static]
```

References factorial(), GiNaC::basic::hold(), and x.

5.1.3.269 factorial_imag_part()

```
static ex GiNaC::factorial_imag_part (
            const ex & x )  [static]
```

5.1.3.270 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)
)
```

5.1.3.271 binomial_evalf()

```
static ex GiNaC::binomial_evalf (
            const ex & x,
            const ex & y )  [static]
```

References binomial(), GiNaC::basic::hold(), and x.

5.1.3.272 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().

5.1.3.273 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.

5.1.3.274 binomial_conjugate()

```
static ex GiNaC::binomial_conjugate (
            const ex & x,
            const ex & y )  [static]
```

References binomial(), GiNaC::basic::hold(), and x.

5.1.3.275 binomial_real_part()

```
static ex GiNaC::binomial_real_part (
            const ex & x,
            const ex & y )  [static]
```

References binomial(), GiNaC::basic::hold(), and x.

5.1.3.276 binomial_imag_part()

```
static ex GiNaC::binomial_imag_part (
             const ex & x,
             const ex & y )  [static]
```

5.1.3.277 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)  )
```

5.1.3.278 Order_eval()

```
static ex GiNaC::Order_eval (
             const ex & x )  [static]
```

References _ex0, _ex1, GiNaC::ex::is_zero(), m, and x.

5.1.3.279 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.

5.1.3.280 Order_conjugate()

```
static ex GiNaC::Order_conjugate (
             const ex & x )  [static]
```

References x.

5.1.3.281 Order_real_part()

```
static ex GiNaC::Order_real_part (
            const ex & x )  [static]
```

References x.

5.1.3.282 Order_imag_part()

```
static ex GiNaC::Order_imag_part (
            const ex & x )  [static]
```

References GiNaC::ex::info(), GiNaC::info_flags::real, and x.

5.1.3.283 Order_expl_derivative()

```
static ex GiNaC::Order_expl_derivative (
            const ex & arg,
            const symbol & s )  [static]
```

References GiNaC::ex::diff().

5.1.3.284 REGISTER_FUNCTION() [13/36]

```
GiNaC::REGISTER_FUNCTION (
            Order ,
            eval_func(Order_eval).  series_func(Order_series).  latex_name("\thcal{O}").
expl_derivative_func(Order_expl_derivative).  conjugate_func(Order_conjugate).  real_part_↩
func(Order_real_part).  imag_part_func(Order_imag_part)  )
```

5.1.3.285 lsolve()

```
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, GiNaC::ex::nops(), GiNaC::ex::op(), options, r, GiNaC::info_flags::relation_equal, rhs(), GiNaC::matrix::rows(), Gi↩NaC::matrix::solve(), GiNaC::info_flags::symbol, and syms.

5.1.3.286 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

f	Function f(x)
x	Symbol f(x)
x1	lower interval limit
x2	upper interval limit

Exceptions

runtime_error	(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.

5.1.3.287 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().

5.1.3.288 zeta() [2/3]

```
template<typename T1 , typename T2 >
function GiNaC::zeta (
             const T1 & p1,
             const T2 & p2 )  [inline]
```

References GiNaC::zeta2_SERIAL::serial.

5.1.3.289 is_the_function< zeta_SERIAL >()

```
template<>
bool GiNaC::is_the_function< zeta_SERIAL > (
            const ex & x )  [inline]
```

References x.

5.1.3.290 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().

5.1.3.291 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.

5.1.3.292 is_the_function< G_SERIAL >()

```
template<>
bool GiNaC::is_the_function< G_SERIAL > (
            const ex & x )  [inline]
```

References x.

5.1.3.293 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().

5.1.3.294 psi() [2/4]

```
template<typename T1 , typename T2 >
function GiNaC::psi (
            const T1 & p1,
            const T2 & p2 )  [inline]
```

References GiNaC::psi2_SERIAL::serial.

5.1.3.295 is_the_function< psi_SERIAL >()

```
template<>
bool GiNaC::is_the_function< psi_SERIAL > (
            const ex & x )  [inline]
```

References x.

5.1.3.296 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().

5.1.3.297 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.

5.1.3.298 is_the_function< iterated_integral_SERIAL >()

```
template<>
bool GiNaC::is_the_function< iterated_integral_SERIAL > (
            const ex & x )  [inline]
```

References x.

5.1.3.299 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(), and GiNaC::integral::series().

5.1.3.300 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.

5.1.3.301 EllipticK_evalf()

```
static ex GiNaC::EllipticK_evalf (
            const ex & k )  [static]
```

References GiNaC::ex::evalf(), k, GiNaC::info_flags::numeric, Pi, and sqrt().

5.1.3.302 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.

5.1.3.303 EllipticK_deriv()

```
static ex GiNaC::EllipticK_deriv (
            const ex & k,
            unsigned deriv_param )  [static]
```

References k.

5.1.3.304 EllipticK_series()

```
static ex GiNaC::EllipticK_series (
            const ex & k,
            const relational & rel,
            int order,
            unsigned options )  [static]
```

References _ex0, _ex1, _ex_1, binomial(), k, GiNaC::subs_options::no_pattern, order, Pi, pow(), GiNaC::ex::series(), and GiNaC::ex::subs().

5.1.3.305 EllipticK_print_latex()

```
static void GiNaC::EllipticK_print_latex (
            const ex & k,
            const print_context & c )  [static]
```

References c, and k.

5.1.3.306 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()  )
```

5.1.3.307 EllipticE_evalf()

```
static ex GiNaC::EllipticE_evalf (
            const ex & k )  [static]
```

References GiNaC::ex::evalf(), k, GiNaC::info_flags::numeric, Pi, and sqrt().

5.1.3.308 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.

5.1.3.309 EllipticE_deriv()

```
static ex GiNaC::EllipticE_deriv (
            const ex & k,
            unsigned deriv_param )  [static]
```

References k.

5.1.3.310 EllipticE_series()

```
static ex GiNaC::EllipticE_series (
            const ex & k,
            const relational & rel,
            int order,
            unsigned options )  [static]
```

References _ex0, _ex1, _ex_1, binomial(), k, GiNaC::subs_options::no_pattern, order, Pi, pow(), GiNaC::ex::series(), and GiNaC::ex::subs().

5.1.3.311 EllipticE_print_latex()

```
static void GiNaC::EllipticE_print_latex (
            const ex & k,
            const print_context & c )  [static]
```

References c, and k.

5.1.3.312 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()  )
```

5.1.3.313 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::info_flags::numeric, one, and to_int().

Referenced by iterated_integral2_evalf(), and iterated_integral3_evalf().

5.1.3.314 iterated_integral2_evalf()

```
static ex GiNaC::iterated_integral2_evalf (
            const ex & kernel_lst,
            const ex & lambda )  [static]
```

References iterated_integral_evalf_impl().

5.1.3.315 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().

5.1.3.316 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.

5.1.3.317 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.

5.1.3.318 lgamma_evalf()

```
static ex GiNaC::lgamma_evalf (
             const ex & x )  [static]
```

References lgamma(), and x.

5.1.3.319 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

GiNaC::pole_error("lgamma_eval()	logarithmic pole",0)

References _ex_1, factorial(), GiNaC::ex::info(), GiNaC::info_flags::integer, lgamma(), log(), GiNaC::info_flags::numeric, GiNaC::info_flags::posint, and x.

5.1.3.320 lgamma_deriv()

```
static ex GiNaC::lgamma_deriv (
             const ex & x,
             unsigned deriv_param )  [static]
```

References GINAC_ASSERT, psi(), and x.

5.1.3.321 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, series(), and GiNaC::ex::subs().

5.1.3.322 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.

5.1.3.323 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("\g \mma")  )
```

5.1.3.324 tgamma_evalf()

```
static ex GiNaC::tgamma_evalf (
             const ex & x )  [static]
```

References tgamma(), and x.

5.1.3.325 tgamma_eval()

```
static ex GiNaC::tgamma_eval (
             const ex & x )  [static]
```

Evaluation of 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

pole_error("tgamma_eval()	simple pole",0)

References _num1_2_p, _num1_p, _num2_p, _num_2_p, abs(), doublefactorial(), factorial(), GiNaC::ex::info(), GiNaC::numeric::is_even(), GiNaC::numeric::is_integer(), GiNaC::numeric::is_positive(), n, GiNaC::info_flags::numeric, Pi, pow(), sqrt(), tgamma(), and x.

5.1.3.326 tgamma_deriv()

```
static ex GiNaC::tgamma_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References GINAC_ASSERT, psi(), tgamma(), and x.

5.1.3.327 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, series(), GiNaC::ex::subs(), and tgamma().

5.1.3.328 tgamma_conjugate()

```
static ex GiNaC::tgamma_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), tgamma(), and x.

5.1.3.329 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("\mma")  )
```

5.1.3.330 beta_evalf()

```
static ex GiNaC::beta_evalf (
            const ex & x,
            const ex & y )  [static]
```

References exp(), lgamma(), and x.

5.1.3.331 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.

5.1.3.332 beta_deriv()

```
static ex GiNaC::beta_deriv (
            const ex & x,
            const ex & y,
            unsigned deriv_param )  [static]
```

References GINAC_ASSERT, psi(), and x.

5.1.3.333 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().

5.1.3.334 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("\thrm{B}").  set_symmetry(sy_symm(0, 1))  )
```

5.1.3.335 psi1_evalf()

```
static ex GiNaC::psi1_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), psi(), and x.

5.1.3.336 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.

5.1.3.337 psi1_deriv()

```
static ex GiNaC::psi1_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, GINAC_ASSERT, psi(), and x.

5.1.3.338 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(), series(), and GiNaC::ex::subs().

5.1.3.339 psi2_evalf()

```
static ex GiNaC::psi2_evalf (
             const ex & n,
             const ex & x )  [static]
```

References GiNaC::basic::hold(), n, psi(), and x.

5.1.3.340 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(), is_integer(), log(), m, n, GiNaC::info_flags::numeric, GiNaC::info_flags::posint, pow(), psi(), tgamma(), x, and zeta().

5.1.3.341 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.

5.1.3.342 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().

5.1.3.343 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.

5.1.3.344 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(), log(), GiNaC::ex::nops(), GiNaC::info_flags::numeric, GiNaC::ex::op(), GiNaC::info_flags::positive, pow(), and x.

5.1.3.345 G3_evalf()

```
static ex GiNaC::G3_evalf (
             const ex & x_,
             const ex & s_,
             const ex & y )  [static]
```

References _ex0, _ex1, GiNaC::ex::begin(), GiNaC::container< C >::begin(), GiNaC::ex::end(), factorial(), G(), GiNaC::basic::hold(), GiNaC::ex::info(), log(), GiNaC::container< C >::nops(), GiNaC::ex::nops(), GiNaC::info_flags::numeric, GiNaC::ex::op(), GiNaC::info_flags::positive, pow(), GiNaC::info_flags::real, and x.

5.1.3.346 G3_eval()

```
static ex GiNaC::G3_eval (
             const ex & x_,
             const ex & s_,
             const ex & y )  [static]
```

References _ex0, _ex1, GiNaC::ex::begin(), GiNaC::container< C >::begin(), GiNaC::info_flags::crational, GiNaC::ex::end(), factorial(), G(), GiNaC::basic::hold(), GiNaC::ex::info(), log(), GiNaC::container< C >::nops(), GiNaC::ex::nops(), GiNaC::info_flags::numeric, GiNaC::ex::op(), GiNaC::info_flags::positive, pow(), GiNaC::info_flags::real, and x.

5.1.3.347 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, to_int(), and x.

5.1.3.348 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(), to_int(), x, and zeta().

5.1.3.349 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.

5.1.3.350 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.

5.1.3.351 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.

5.1.3.352 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()  )
```

5.1.3.353 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, to_int(), and x.

5.1.3.354 S_eval()

```
static ex GiNaC::S_eval (
             const ex & n,
             const ex & p,
             const ex & x )  [static]
```

References _ex0, 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().

5.1.3.355 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.

5.1.3.356 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.

5.1.3.357 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.

5.1.3.358 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()  )
```

5.1.3.359 H_evalf()

```
static ex GiNaC::H_evalf (
             const ex & x1,
             const ex & x2 )  [static]
```

References abs(), GiNaC::container< C >::begin(), GiNaC::container< C >::end(), GiNaC::ex::evalf(), evalf(), I, GiNaC::ex::info(), GiNaC::info_flags::integer, m, GiNaC::container< C >::nops(), GiNaC::ex::nops(), GiNaC↩
::container< C >::op(), GiNaC::ex::op(), Pi, GiNaC::ex::subs(), to_int(), and x.

5.1.3.360 H_eval()

```
static ex GiNaC::H_eval (
             const ex & m_,
             const ex & x )  [static]
```

References _ex0, _ex1, _ex_1, GiNaC::info_flags::crational, GiNaC::ex::evalf(), factorial(), GiNaC::ex::info(), Gi↩
NaC::info_flags::integer, log(), m, n, GiNaC::info_flags::numeric, pow(), step(), and x.

5.1.3.361 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.

5.1.3.362 H_deriv()

```
static ex GiNaC::H_deriv (
             const ex & m_,
             const ex & x,
             unsigned deriv_param )  [static]
```

References _ex0, _ex1, _ex_1, GINAC_ASSERT, m, and x.

5.1.3.363 H_print_latex()

```
static void GiNaC::H_print_latex (
             const ex & m_,
             const ex & x,
             const print_context & c )  [static]
```

References c, m, GiNaC::ex::print(), and x.

5.1.3.364 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()  )
```

5.1.3.365 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().

5.1.3.366 zeta1_eval()

```
static ex GiNaC::zeta1_eval (
            const ex & m )  [static]
```

References _ex0, _ex_1_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().

5.1.3.367 zeta1_deriv()

```
static ex GiNaC::zeta1_deriv (
            const ex & m,
            unsigned deriv_param )  [static]
```

References _ex0, _ex1, GINAC_ASSERT, and m.

5.1.3.368 zeta1_print_latex()

```
static void GiNaC::zeta1_print_latex (
            const ex & m_,
            const print_context & c )  [static]
```

References c, m, and GiNaC::ex::print().

5.1.3.369 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().

5.1.3.370 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().

5.1.3.371 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.

5.1.3.372 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(), c, and m.

5.1.3.373 exp_evalf()

```
static ex GiNaC::exp_evalf (
             const ex & x )  [static]
```

References exp(), GiNaC::basic::hold(), and x.

5.1.3.374 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.

5.1.3.375 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.

5.1.3.376 exp_deriv()

```
static ex GiNaC::exp_deriv (
             const ex & x,
             unsigned deriv_param )  [static]
```

References exp(), GINAC_ASSERT, and x.

5.1.3.377 exp_real_part()

```
static ex GiNaC::exp_real_part (
             const ex & x )  [static]
```

References cos(), exp(), imag_part(), real_part(), and x.

5.1.3.378 exp_imag_part()

```
static ex GiNaC::exp_imag_part (
             const ex & x )  [static]
```

References exp(), imag_part(), real_part(), sin(), and x.

5.1.3.379 exp_conjugate()

```
static ex GiNaC::exp_conjugate (
             const ex & x )  [static]
```

References GiNaC::ex::conjugate(), exp(), and x.

5.1.3.380 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.

5.1.3.381 REGISTER_FUNCTION() [22/36]

```
GiNaC::REGISTER_FUNCTION (
            exp ,
            eval_func(exp_eval).  evalf_func(exp_evalf).  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("\p")  )
```

5.1.3.382 log_evalf()

```
static ex GiNaC::log_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), log(), and x.

5.1.3.383 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, GiNa↩
C::info_flags::numeric, GiNaC::ex::op(), Pi, GiNaC::info_flags::rational, GiNaC::info_flags::real, and x.

5.1.3.384 log_deriv()

```
static ex GiNaC::log_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex_1, GINAC_ASSERT, and x.

5.1.3.385 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(), GiNa↩
C::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(), Gi↩
NaC::ex::series(), series(), GiNaC::ex::subs(), and GiNaC::series_options::suppress_branchcut.

5.1.3.386 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.

5.1.3.387 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.

5.1.3.388 log_expand()

```
static ex GiNaC::log_expand (
            const ex & arg,
            unsigned options )  [static]
```

References _ex1, _ex_1, GiNaC::ex::begin(), GiNaC::ex::end(), GiNaC::ex::expand(), expand(), GiNaC::expand_options::expand_function_args, 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, and GiNaC::status_flags::purely_indefinite.

5.1.3.389 log_conjugate()

```
static ex GiNaC::log_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::info(), GiNaC::ex::is_zero(), log(), GiNaC::info_flags::positive, and x.

5.1.3.390 REGISTER_FUNCTION() [23/36]

```
GiNaC::REGISTER_FUNCTION (
            log ,
            eval_func(log_eval).  evalf_func(log_evalf).  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("\")  )
```

5.1.3.391 sin_evalf()

```
static ex GiNaC::sin_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), sin(), and x.

5.1.3.392 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.

5.1.3.393 sin_deriv()

```
static ex GiNaC::sin_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References cos(), GINAC_ASSERT, and x.

5.1.3.394 sin_real_part()

```
static ex GiNaC::sin_real_part (
            const ex & x )  [static]
```

References cosh(), imag_part(), real_part(), sin(), and x.

5.1.3.395 sin_imag_part()

```
static ex GiNaC::sin_imag_part (
            const ex & x )  [static]
```

References cos(), imag_part(), real_part(), sinh(), and x.

5.1.3.396 sin_conjugate()

```
static ex GiNaC::sin_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), sin(), and x.

5.1.3.397 REGISTER_FUNCTION() [24/36]

```
GiNaC::REGISTER_FUNCTION (
            sin ,
            eval_func(sin_eval).  evalf_func(sin_evalf).  derivative_func(sin_deriv).  real↩
_part_func(sin_real_part).  imag_part_func(sin_imag_part).  conjugate_func(sin_conjugate).
latex_name("\n")  )
```

5.1.3.398 cos_evalf()

```
static ex GiNaC::cos_evalf (
            const ex & x )  [static]
```

References cos(), GiNaC::basic::hold(), and x.

5.1.3.399 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.

5.1.3.400 cos_deriv()

```
static ex GiNaC::cos_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References GINAC_ASSERT, sin(), and x.

5.1.3.401 cos_real_part()

```
static ex GiNaC::cos_real_part (
            const ex & x )  [static]
```

References cos(), cosh(), imag_part(), real_part(), and x.

5.1.3.402 cos_imag_part()

```
static ex GiNaC::cos_imag_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), sin(), sinh(), and x.

5.1.3.403 cos_conjugate()

```
static ex GiNaC::cos_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), cos(), and x.

5.1.3.404 REGISTER_FUNCTION() [25/36]

```
GiNaC::REGISTER_FUNCTION (
            cos ,
            eval_func(cos_eval).  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("\s")  )
```

5.1.3.405 tan_evalf()

```
static ex GiNaC::tan_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), tan(), and x.

5.1.3.406 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.

5.1.3.407 tan_deriv()

```
static ex GiNaC::tan_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, _ex2, GINAC_ASSERT, tan(), and x.

5.1.3.408 tan_real_part()

```
static ex GiNaC::tan_real_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), tan(), and x.

5.1.3.409 tan_imag_part()

```
static ex GiNaC::tan_imag_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), tan(), tanh(), and x.

5.1.3.410 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.

5.1.3.411 tan_conjugate()

```
static ex GiNaC::tan_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), tan(), and x.

5.1.3.412 REGISTER_FUNCTION() [26/36]

```
GiNaC::REGISTER_FUNCTION (
            tan ,
            eval_func(tan_eval).  evalf_func(tan_evalf).  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("\n")  )
```

5.1.3.413 asin_evalf()

```
static ex GiNaC::asin_evalf (
            const ex & x )  [static]
```

References asin(), GiNaC::basic::hold(), and x.

5.1.3.414 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.

5.1.3.415 asin_deriv()

```
static ex GiNaC::asin_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex2, _ex_1_2, GINAC_ASSERT, and x.

5.1.3.416 asin_conjugate()

```
static ex GiNaC::asin_conjugate (
            const ex & x )  [static]
```

References _num1_p, _num_1_p, asin(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::is_zero(), and x.

5.1.3.417 REGISTER_FUNCTION() [27/36]

```
GiNaC::REGISTER_FUNCTION (
            asin ,
            eval_func(asin_eval).  evalf_func(asin_evalf).  derivative_func(asin_deriv).
conjugate_func(asin_conjugate).  latex_name("\csin")  )
```

5.1.3.418 acos_evalf()

```
static ex GiNaC::acos_evalf (
            const ex & x )  [static]
```

References acos(), GiNaC::basic::hold(), and x.

5.1.3.419 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.

5.1.3.420 acos_deriv()

```
static ex GiNaC::acos_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex2, _ex_1_2, GINAC_ASSERT, and x.

5.1.3.421 acos_conjugate()

```
static ex GiNaC::acos_conjugate (
            const ex & x )  [static]
```

References _num1_p, _num_1_p, acos(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::is_zero(), and x.

5.1.3.422 REGISTER_FUNCTION() [28/36]

```
GiNaC::REGISTER_FUNCTION (
            acos ,
            eval_func(acos_eval).  evalf_func(acos_evalf).  derivative_func(acos_deriv).
conjugate_func(acos_conjugate).  latex_name("\ccos")  )
```

5.1.3.423 atan_evalf()

```
static ex GiNaC::atan_evalf (
            const ex & x )  [static]
```

References atan(), GiNaC::basic::hold(), and x.

5.1.3.424 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.

5.1.3.425 atan_deriv()

```
static ex GiNaC::atan_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, _ex2, _ex_1, GINAC_ASSERT, and x.

5.1.3.426 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::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.

5.1.3.427 atan_conjugate()

```
static ex GiNaC::atan_conjugate (
            const ex & x )  [static]
```

References _num1_p, _num_1_p, atan(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::info(), GiNaC::numeric::is_zero(), GiNaC::info_flags::real, GiNaC::ex::real_part(), and x.

5.1.3.428 REGISTER_FUNCTION() [29/36]

```
GiNaC::REGISTER_FUNCTION (
            atan ,
            eval_func(atan_eval).  evalf_func(atan_evalf).  derivative_func(atan_deriv).
series_func(atan_series).  conjugate_func(atan_conjugate).  latex_name("\ctan")  )
```

5.1.3.429 atan2_evalf()

```
static ex GiNaC::atan2_evalf (
            const ex & y,
            const ex & x )  [static]
```

References atan(), and x.

5.1.3.430 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.

5.1.3.431 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.

5.1.3.432 REGISTER_FUNCTION() [30/36]

```
GiNaC::REGISTER_FUNCTION (
             atan2 ,
             eval_func(atan2_eval).  evalf_func(atan2_evalf).  derivative_func(atan2_deriv)  )
```

5.1.3.433 sinh_evalf()

```
static ex GiNaC::sinh_evalf (
             const ex & x )  [static]
```

References GiNaC::basic::hold(), sinh(), and x.

5.1.3.434 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::numeric::is_zero(), GiNaC::ex::is_zero(), GiNaC::info_↩ flags::negative, GiNaC::info_flags::numeric, GiNaC::ex::op(), Pi, real(), sin(), sinh(), sqrt(), and x.

5.1.3.435 sinh_deriv()

```
static ex GiNaC::sinh_deriv (
             const ex & x,
             unsigned deriv_param )  [static]
```

References cosh(), GINAC_ASSERT, and x.

5.1.3.436 sinh_real_part()

```
static ex GiNaC::sinh_real_part (
            const ex & x )  [static]
```

References cos(), imag_part(), real_part(), sinh(), and x.

5.1.3.437 sinh_imag_part()

```
static ex GiNaC::sinh_imag_part (
            const ex & x )  [static]
```

References cosh(), imag_part(), real_part(), sin(), and x.

5.1.3.438 sinh_conjugate()

```
static ex GiNaC::sinh_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), sinh(), and x.

5.1.3.439 REGISTER_FUNCTION() [31/36]

```
GiNaC::REGISTER_FUNCTION (
            sinh ,
            eval_func(sinh_eval).  evalf_func(sinh_evalf).  derivative_func(sinh_deriv).
real_part_func(sinh_real_part).  imag_part_func(sinh_imag_part).  conjugate_func(sinh_conjugate).
latex_name("\nh")  )
```

5.1.3.440 cosh_evalf()

```
static ex GiNaC::cosh_evalf (
            const ex & x )  [static]
```

References cosh(), GiNaC::basic::hold(), and x.

5.1.3.441 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::numeric::is_zero(), GiNaC::ex::is_zero(), GiNaC::info_flags::negative, GiNaC::info_flags::numeric, GiNaC::ex::op(), Pi, real(), sqrt(), and x.

5.1.3.442 cosh_deriv()

```
static ex GiNaC::cosh_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References GINAC_ASSERT, sinh(), and x.

5.1.3.443 cosh_real_part()

```
static ex GiNaC::cosh_real_part (
            const ex & x )  [static]
```

References cos(), cosh(), imag_part(), real_part(), and x.

5.1.3.444 cosh_imag_part()

```
static ex GiNaC::cosh_imag_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), sin(), sinh(), and x.

5.1.3.445 cosh_conjugate()

```
static ex GiNaC::cosh_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), cosh(), and x.

5.1.3.446 REGISTER_FUNCTION() [32/36]

```
GiNaC::REGISTER_FUNCTION (
            cosh ,
            eval_func(cosh_eval).  evalf_func(cosh_evalf).  derivative_func(cosh_deriv).
real_part_func(cosh_real_part).  imag_part_func(cosh_imag_part).  conjugate_func(cosh_conjugate).
latex_name("\sh")  )
```

5.1.3.447 tanh_evalf()

```
static ex GiNaC::tanh_evalf (
            const ex & x )  [static]
```

References GiNaC::basic::hold(), tanh(), and x.

5.1.3.448 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::numeric::is_zero(), GiNaC::ex::is_zero(), GiNaC↩
::info_flags::negative, GiNaC::info_flags::numeric, GiNaC::ex::op(), Pi, real(), sqrt(), tan(), tanh(), and x.

5.1.3.449 tanh_deriv()

```
static ex GiNaC::tanh_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, _ex2, GINAC_ASSERT, tanh(), and x.

5.1.3.450 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.

5.1.3.451 tanh_real_part()

```
static ex GiNaC::tanh_real_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), tan(), tanh(), and x.

5.1.3.452 tanh_imag_part()

```
static ex GiNaC::tanh_imag_part (
            const ex & x )  [static]
```

References imag_part(), real_part(), tan(), tanh(), and x.

5.1.3.453 tanh_conjugate()

```
static ex GiNaC::tanh_conjugate (
            const ex & x )  [static]
```

References GiNaC::ex::conjugate(), tanh(), and x.

5.1.3.454 REGISTER_FUNCTION() [33/36]

```
GiNaC::REGISTER_FUNCTION (
            tanh ,
            eval_func(tanh_eval).  evalf_func(tanh_evalf).  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("\nh")  )
```

5.1.3.455 asinh_evalf()

```
static ex GiNaC::asinh_evalf (
            const ex & x )  [static]
```

References asinh(), GiNaC::basic::hold(), and x.

5.1.3.456 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.

5.1.3.457 asinh_deriv()

```
static ex GiNaC::asinh_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, _ex2, _ex_1_2, GINAC_ASSERT, and x.

5.1.3.458 asinh_conjugate()

```
static ex GiNaC::asinh_conjugate (
            const ex & x )  [static]
```

References _num1_p, _num_1_p, asinh(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::info(), GiNaC::numeric::is_zero(), GiNaC::info_flags::real, GiNaC::ex::real_part(), and x.

5.1.3.459 REGISTER_FUNCTION() [34/36]

```
GiNaC::REGISTER_FUNCTION (
            asinh ,
            eval_func(asinh_eval).  evalf_func(asinh_evalf).  derivative_func(asinh_deriv).
conjugate_func(asinh_conjugate)  )
```

5.1.3.460 acosh_evalf()

```
static ex GiNaC::acosh_evalf (
            const ex & x )  [static]
```

References acosh(), GiNaC::basic::hold(), and x.

5.1.3.461 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.

5.1.3.462 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.

5.1.3.463 acosh_conjugate()

```
static ex GiNaC::acosh_conjugate (
            const ex & x )  [static]
```

References _num1_p, acosh(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::is_zero(), and x.

5.1.3.464 REGISTER_FUNCTION() [35/36]

```
GiNaC::REGISTER_FUNCTION (
            acosh ,
            eval_func(acosh_eval).  evalf_func(acosh_evalf).  derivative_func(acosh_deriv).
conjugate_func(acosh_conjugate)  )
```

5.1.3.465 atanh_evalf()

```
static ex GiNaC::atanh_evalf (
            const ex & x )  [static]
```

References atanh(), GiNaC::basic::hold(), and x.

5.1.3.466 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.

5.1.3.467 atanh_deriv()

```
static ex GiNaC::atanh_deriv (
            const ex & x,
            unsigned deriv_param )  [static]
```

References _ex1, _ex2, _ex_1, GINAC_ASSERT, and x.

5.1.3.468 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.

5.1.3.469 atanh_conjugate()

```
static ex GiNaC::atanh_conjugate (
            const ex & x )  [static]
```

References _num1_p, _num_1_p, atanh(), GiNaC::ex::conjugate(), GiNaC::basic::hold(), GiNaC::ex::imag_part(), GiNaC::ex::is_zero(), and x.

5.1.3.470 REGISTER_FUNCTION() [36/36]

```
GiNaC::REGISTER_FUNCTION (
            atanh ,
            eval_func(atanh_eval).  evalf_func(atanh_evalf).  derivative_func(atanh_deriv).
series_func(atanh_series).  conjugate_func(atanh_conjugate)  )
```

5.1.3.471 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [10/32]

```
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  )
```

5.1.3.472 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().

5.1.3.473 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().

5.1.3.474 GINAC_BIND_UNARCHIVER() [21/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             integral  )
```

5.1.3.475 GINAC_DECLARE_UNARCHIVER() [22/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             integral  )
```

5.1.3.476 ifactor()

```
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 ...

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().

5.1.3.477 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(), GiNaC::numeric::is_odd(), mod(), n, GiNaC::container< C >::nops(), and GiNaC↩ ::container< C >::op().

5.1.3.478 kronecker_symbol()

```
numeric GiNaC::kronecker_symbol (
             const numeric & a,
             const numeric & n )
```

Returns the Kronecker symbol a: integer n: integer.

This routine defines kronecker_symbol(1,0) = 1 kronecker_symbol(-1,0) = 1 kronecker_symbol(a,0) = 0, a != 1,-1

In particular kronecker_symbol(-1,0) = 1 (in agreement with Sage)

Ref.: Toshitsune Miyake, Modular Forms, Chapter 3.1

References ifactor(), GiNaC::numeric::is_even(), n, GiNaC::container< C >::op(), and pow().

Referenced by primitive_dirichlet_character().

5.1.3.479 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().

5.1.3.480 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().

5.1.3.481 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(), and x.

5.1.3.482 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().

5.1.3.483 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [11/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             integration_kernel ,
             basic ,
             print_func< print_context > &::do_print  )
```

5.1.3.484 GINAC_BIND_UNARCHIVER() [22/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             integration_kernel  )
```

5.1.3.485 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [12/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             basic_log_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.486 GINAC_BIND_UNARCHIVER() [23/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             basic_log_kernel  )
```

5.1.3.487 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [13/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
              multiple_polylog_kernel ,
              integration_kernel ,
              print_func< print_context > &::do_print  )
```

5.1.3.488 GINAC_BIND_UNARCHIVER() [24/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              multiple_polylog_kernel  )
```

5.1.3.489 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [14/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
              ELi_kernel ,
              integration_kernel ,
              print_func< print_context > &::do_print  )
```

5.1.3.490 GINAC_BIND_UNARCHIVER() [25/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              ELi_kernel  )
```

5.1.3.491 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [15/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
              Ebar_kernel ,
              integration_kernel ,
              print_func< print_context > &::do_print  )
```

5.1.3.492 GINAC_BIND_UNARCHIVER() [26/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              Ebar_kernel  )
```

5.1.3.493 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [16/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             Kronecker_dtau_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.494 GINAC_BIND_UNARCHIVER() [27/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             Kronecker_dtau_kernel  )
```

5.1.3.495 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [17/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             Kronecker_dz_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.496 GINAC_BIND_UNARCHIVER() [28/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             Kronecker_dz_kernel  )
```

5.1.3.497 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [18/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             Eisenstein_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.498 GINAC_BIND_UNARCHIVER() [29/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             Eisenstein_kernel  )
```

5.1.3.499 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [19/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             Eisenstein_h_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.500 GINAC_BIND_UNARCHIVER() [30/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             Eisenstein_h_kernel  )
```

5.1.3.501 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [20/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             modular_form_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.502 GINAC_BIND_UNARCHIVER() [31/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             modular_form_kernel  )
```

5.1.3.503 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [21/32]

```
GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT (
             user_defined_kernel ,
             integration_kernel ,
             print_func< print_context > &::do_print  )
```

5.1.3.504 GINAC_BIND_UNARCHIVER() [32/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             user_defined_kernel  )
```

5.1.3.505 GINAC_DECLARE_UNARCHIVER() [23/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            integration_kernel  )
```

5.1.3.506 GINAC_DECLARE_UNARCHIVER() [24/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            basic_log_kernel  )
```

5.1.3.507 GINAC_DECLARE_UNARCHIVER() [25/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            multiple_polylog_kernel  )
```

5.1.3.508 GINAC_DECLARE_UNARCHIVER() [26/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            ELi_kernel  )
```

5.1.3.509 GINAC_DECLARE_UNARCHIVER() [27/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            Ebar_kernel  )
```

5.1.3.510 GINAC_DECLARE_UNARCHIVER() [28/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            Kronecker_dtau_kernel  )
```

5.1.3.511 GINAC_DECLARE_UNARCHIVER() [29/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            Kronecker_dz_kernel  )
```

5.1.3.512 GINAC_DECLARE_UNARCHIVER() [30/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             Eisenstein_kernel  )
```

5.1.3.513 GINAC_DECLARE_UNARCHIVER() [31/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             Eisenstein_h_kernel  )
```

5.1.3.514 GINAC_DECLARE_UNARCHIVER() [32/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             modular_form_kernel  )
```

5.1.3.515 GINAC_DECLARE_UNARCHIVER() [33/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             user_defined_kernel  )
```

5.1.3.516 GINAC_DECLARE_UNARCHIVER() [34/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             lst  )
```

5.1.3.517 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [22/32]

```
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.

5.1.3.518 GINAC_BIND_UNARCHIVER() [33/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              matrix  )
```

5.1.3.519 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().

5.1.3.520 diag_matrix() [1/2]

```
ex GiNaC::diag_matrix (
              const lst & l )
```

Convert list of diagonal elements to matrix.

References GiNaC::container< C >::nops().

5.1.3.521 diag_matrix() [2/2]

```
ex GiNaC::diag_matrix (
              std::initializer_list< ex > l )
```

5.1.3.522 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().

5.1.3.523 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().

5.1.3.524 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().

5.1.3.525 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().

5.1.3.526 GINAC_DECLARE_UNARCHIVER() [35/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             matrix  )
```

5.1.3.527 nops() [2/2]

```
size_t GiNaC::nops (
            const matrix & m )  [inline]
```

References m.

5.1.3.528 expand() [2/2]

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

References m, and options.

5.1.3.529 evalf() [2/2]

```
ex GiNaC::evalf (
            const matrix & m )  [inline]
```

References m.

5.1.3.530 rows()

```
unsigned GiNaC::rows (
            const matrix & m )  [inline]
```

References m.

Referenced by lst_to_matrix(), and reduced_matrix().

5.1.3.531 cols()

```
unsigned GiNaC::cols (
            const matrix & m )  [inline]
```

References m.

Referenced by lst_to_matrix(), and reduced_matrix().

5.1.3.532 transpose()

```
matrix GiNaC::transpose (
            const matrix & m )  [inline]
```

References m.

Referenced by GiNaC::clifford::get_metric().

5.1.3.533 determinant()

```
ex GiNaC::determinant (
            const matrix & m,
            unsigned options = determinant_algo::automatic )  [inline]
```

References m, and options.

5.1.3.534 trace()

```
ex GiNaC::trace (
            const matrix & m )  [inline]
```

References m.

5.1.3.535 charpoly()

```
ex GiNaC::charpoly (
            const matrix & m,
            const ex & lambda )  [inline]
```

References m.

5.1.3.536 inverse() [1/3]

```
matrix GiNaC::inverse (
            const matrix & m )  [inline]
```

References GiNaC::solve_algo::automatic, and m.

Referenced by multiply_lcm().

5.1.3.537 inverse() [2/3]

```
matrix GiNaC::inverse (
            const matrix & m,
            unsigned algo )  [inline]
```

References m.

5.1.3.538 rank() [1/2]

```
unsigned GiNaC::rank (
            const matrix & m )  [inline]
```

References m.

5.1.3.539 rank() [2/2]

```
unsigned GiNaC::rank (
            const matrix & m,
            unsigned solve_algo )  [inline]
```

References m.

5.1.3.540 unit_matrix() [2/2]

```
ex GiNaC::unit_matrix (
            unsigned x )  [inline]
```

Create a x times x unit matrix.

References unit_matrix(), and x.

5.1.3.541 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().

5.1.3.542 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [23/32]

```
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  )
```

5.1.3.543 tryfactsubs()

```
bool GiNaC::tryfactsubs (
             const ex & origfactor,
             const ex & patternfactor,
             int & nummatches,
             exmap & repls )
```

References GiNaC::ex::info(), GiNaC::ex::match(), GiNaC::ex::op(), and to_int().

Referenced by algebraic_match_mul_with_mul(), GiNaC::mul::algebraic_subs_mul(), and GiNaC::power::subs().

5.1.3.544 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 > & subsed,
             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. subsed[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 factor(), GINAC_ASSERT, GiNaC::expairseq::nops(), GiNaC::ex::nops(), GiNaC::expairseq::op(), GiNaC::ex::op(), and tryfactsubs().

Referenced by GiNaC::mul::algebraic_subs_mul(), and GiNaC::mul::has().

5.1.3.545 GINAC_BIND_UNARCHIVER() [34/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             mul  )
```

5.1.3.546 GINAC_DECLARE_UNARCHIVER() [36/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              mul  )
```

5.1.3.547 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [24/32]

```
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  )
```

5.1.3.548 reeval_ncmul()

```
ex GiNaC::reeval_ncmul (
              const exvector & v )
```

5.1.3.549 hold_ncmul()

```
ex GiNaC::hold_ncmul (
              const exvector & v )
```

References _ex1, and GiNaC::status_flags::evaluated.

Referenced by GiNaC::color::eval_ncmul(), GiNaC::structure< T, ComparisonPolicy >::eval_ncmul(), and GiNaC↩
::basic::eval_ncmul().

5.1.3.550 GINAC_BIND_UNARCHIVER() [35/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              ncmul  )
```

5.1.3.551 GINAC_DECLARE_UNARCHIVER() [37/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              ncmul  )
```

5.1.3.552 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

e	expression to search
x	first symbol found (returned)

Returns

"false" if no symbol was found, "true" otherwise

References GiNaC::ex::nops(), GiNaC::ex::op(), and x.

Referenced by divide(), frac_cancel(), and GiNaC::ex::unit().

5.1.3.553 add_symbol()

```
static void GiNaC::add_symbol (
             const ex & s,
             sym_desc_vec & v )  [static]
```

Referenced by collect_symbols().

5.1.3.554 collect_symbols()

```
static void GiNaC::collect_symbols (
             const ex & e,
             sym_desc_vec & v )  [static]
```

References add_symbol(), GiNaC::ex::nops(), and GiNaC::ex::op().

Referenced by get_symbol_stats().

5.1.3.555 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

a	first multivariate polynomial
b	second multivariate polynomial
v	vector of sym_desc 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().

5.1.3.556 lcmcoeff()

```
static numeric GiNaC::lcmcoeff (
            const ex & e,
            const numeric & l )  [static]
```

References _num1_p, c, GiNaC::ex::info(), lcm(), GiNaC::ex::nops(), GiNaC::ex::op(), pow(), and GiNaC::info_flags::rational.

Referenced by heur_gcd(), lcm_of_coefficients_denominators(), and multiply_lcm().

5.1.3.557 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 Q[X] to Z[X].

Parameters

e	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().

5.1.3.558 multiply_lcm()

```
static ex GiNaC::multiply_lcm (
             const ex & e,
             const numeric & lcm )  [static]
```

Bring polynomial from Q[X] to Z[X] by multiplying in the previously determined LCM of the coefficient's denominators.

Parameters

e	multivariate polynomial (need not be expanded)
lcm	LCM to multiply in

References _num1_p, inverse(), GiNaC::ex::is_equal(), GiNaC::numeric::is_rational(), lcm(), lcmcoeff(), GiNaC::ex::nops(), GiNaC::ex::op(), and pow().

Referenced by frac_cancel(), and sqrfree().

5.1.3.559 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 Q[x].

It satisfies a(x)=b(x)∗q(x)+r(x).

Parameters

a	first polynomial in x (dividend)
b	second polynomial in x (divisor)
x	a and b are polynomials in x
check_args	check whether a and b are polynomials with rational coefficients (defaults to "true")

Returns

quotient of a and b in Q[x]

References _ex1, GiNaC::ex::coeff(), GiNaC::ex::degree(), divide(), GiNaC::ex::expand(), 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().

5.1.3.560 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 Q[x].

It satisfies a(x)=b(x)∗q(x)+r(x).

Parameters

a	first polynomial in x (dividend)
b	second polynomial in x (divisor)
x	a and b are polynomials in x
check_args	check whether a and b are polynomials with rational coefficients (defaults to "true")

Returns

remainder of a(x) and b(x) in Q[x]

References _ex0, GiNaC::ex::coeff(), GiNaC::ex::degree(), divide(), GiNaC::ex::expand(), 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(), irem(), and sqrfree_parfrac().

5.1.3.561 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 degree(n, x) $<$ degree(D, x).

Parameters

a	rational function in x
x	a is a function of x

Returns

decomposed function.

References denom(), numer(), numer_denom(), GiNaC::ex::op(), quo(), rem(), and x.

5.1.3.562 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 Q[x].

Parameters

a	first polynomial in x (dividend)
b	second polynomial in x (divisor)
x	a and b are polynomials in x
check_args	check whether a and b are polynomials with rational coefficients (defaults to "true")

Returns

pseudo-remainder of a(x) and b(x) in Q[x]

References _ex0, _ex1, GiNaC::ex::coeff(), GiNaC::ex::degree(), GiNaC::ex::expand(), expand(), GiNaC::ex::info(), GiNaC::ex::is_zero(), pow(), r, GiNaC::info_flags::rational_polynomial, and x.

Referenced by sr_gcd().

5.1.3.563 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 Q[x].

Parameters

a	first polynomial in x (dividend)
b	second polynomial in x (divisor)
x	a and b are polynomials in x
check_args	check whether a and b are polynomials with rational coefficients (defaults to "true")

Returns

sparse pseudo-remainder of a(x) and b(x) in Q[x]

References _ex0, _ex1, GiNaC::ex::coeff(), GiNaC::ex::degree(), GiNaC::ex::expand(), expand(), GiNaC::ex::info(), GiNaC::ex::is_zero(), pow(), r, GiNaC::info_flags::rational_polynomial, and x.

5.1.3.564 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 Q[X].

Parameters

a	first multivariate polynomial (dividend)
b	second multivariate polynomial (divisor)
q	quotient (returned)
check_args	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(), GiNaC::ex::expand(), 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, to_int(), and x.

Referenced by find_common_factor(), GiNaC::matrix::fraction_free_elimination(), gcd(), quo(), and rem().

5.1.3.565 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 Z[X].

This functions works like divide() but the input and output polynomials are in Z[X] instead of QX. 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

a	first multivariate polynomial (dividend)
b	second multivariate polynomial (divisor)
q	quotient (returned)
var	iterator to first element of vector of sym_desc 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, GiNaC::ex::begin(), c, GiNaC::ex::coeff(), GiNaC::ex::degree(), GiNaC::ex::expand(), 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(), to_int(), and x.

Referenced by heur_gcd_z(), and sr_gcd().

5.1.3.566 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

a	first multivariate polynomial
b	second multivariate polynomial
var	iterator to first element of vector of sym_desc structs

Returns

the GCD as a new expression

See also

gcd

References _ex0, _ex1, c, GiNaC::ex::content(), GiNaC::ex::degree(), divide_in_z(), gcd(), pow(), prem(), GiNaC::ex::primpart(), psi(), r, and x.

Referenced by gcd().

5.1.3.567 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().

5.1.3.568 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

a	first integer multivariate polynomial (expanded)
b	second integer multivariate polynomial (expanded)
ca	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
cb	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
var	iterator to first element of vector of sym_desc structs
res	the GCD (returned)

Returns

true if GCD was computed, false otherwise.

See also

gcd

Exceptions

gcdheu_failed()	

References GiNaC::ex::degree(), divide_in_z(), GiNaC::ex::expand(), gcd(), 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().

5.1.3.569 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

a	first rational multivariate polynomial (expanded)
b	second rational multivariate polynomial (expanded)
ca	cofactor of polynomial a (returned), nullptr to suppress calculation of cofactor
cb	cofactor of polynomial b (returned), nullptr to suppress calculation of cofactor
var	iterator to first element of vector of sym_desc structs
res	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().

5.1.3.570 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_pow(), GINAC_ASSERT, GiNaC::ex::is_equal(), GiNaC::ex::ldegree(), GiNaC::ex::op(), pow(), and to_int().

Referenced by gcd().

5.1.3.571 gcd_pf_mul()

```
static ex GiNaC::gcd_pf_mul (
             const ex & a,
             const ex & b,
             ex * ca,
             ex * cb )  [static]
```

References gcd(), GINAC_ASSERT, GiNaC::ex::nops(), and GiNaC::ex::op().

Referenced by gcd().

5.1.3.572 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 Z[X].

Optionally also compute the cofactors of a and b, defined by a = ca $*$ gcd(a, b) and b = cb $*$ gcd(a, b).

Parameters

a	first multivariate polynomial
b	second multivariate polynomial
ca	pointer to expression that will receive the cofactor of a, or nullptr
cb	pointer to expression that will receive the cofactor of b, or nullptr
check_args	check whether a and b are polynomials with rational coefficients (defaults to "true")

Returns

the GCD as a new expression

References _ex0, _ex1, GiNaC::ex::begin(), divide(), GiNaC::ex::expand(), expand(), gcd_pf_mul(), gcd_pf_pow(), get_symbol_stats(), heur_gcd(), GiNaC::ex::info(), GiNaC::ex::integer_content(), GiNaC::ex::is_equal(), GiNaC::numeric::is_zero(), GiNaC::ex::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_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().

5.1.3.573 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().

5.1.3.574 lcm() [1/2]

```
ex GiNaC::lcm (
              const ex & a,
              const ex & b,
              bool check_args )
```

Compute LCM (Least Common Multiple) of multivariate polynomials in Z[X].

Parameters

a	first multivariate polynomial
b	second multivariate polynomial
check_args	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(), and GiNaC::info_flags::rational_polynomial.

Referenced by GiNaC::add::integer_content(), lcmcoeff(), multiply_lcm(), and sqrfree().

5.1.3.575 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

a	multivariate polynomial over Z[X], treated here as univariate polynomial in x (needs not be expanded).
x	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().

5.1.3.576 sqrfree()

```
ex GiNaC::sqrfree (
             const ex & a,
             const lst & l )
```

Compute a square-free factorization of a multivariate polynomial in Q[X].

Parameters

a	multivariate polynomial over Q[X] (needs not be expanded)
l	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 $i = 1, \ldots, r$;
3. the degree of the polynomial p_i is strictly positive for $i = 1, \ldots, r$;
4. the polynomial $\Pi_{i=1}^{r} p_i(X)$ 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_yun(), and x.

5.1.3.577 sqrfree_parfrac()

```
ex GiNaC::sqrfree_parfrac (
            const ex & a,
            const symbol & x )
```

Compute square-free partial fraction decomposition of rational function a(x).

Parameters

a	rational function over Z[x], treated as univariate polynomial in x
x	variable to factor in

Returns

decomposed rational function

References _ex1, GiNaC::ex::coeff(), GiNaC::basic::coeff(), coeff(), GiNaC::ex::degree(), denom(), GiNaC::ex↩ ::expand(), factor(), k, numer(), numer_denom(), GiNaC::ex::op(), pow(), quo(), rem(), rhs(), GiNaC::matrix::solve(), sqrfree_yun(), to_int(), and x.

5.1.3.578 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(2∗x)∗cos(exp(2∗x)+1)∗exp(x) The first factor GiNaC denotes by something like symbol1 and will record: e =symbol1∗cos(symbol1 + 1)∗exp(x) repl = {symbol1 : exp(2∗x)} Similarly, the second factor would be denoted as symbol2 and we will have e =symbol1∗symbol2∗exp(x) repl = {symbol1 : exp(2∗x), symbol2 : cos(symbol1 + 1)} Denoting the third term as symbol3 GiNaC is willing to re-think exp(2∗x) as 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(2∗x)" shall be deleted from the replacement table repl. However, this will create illegal substitution "symbol2 : cos(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 =symbol3^2∗symbol2∗symbol3 repl = {symbol2 : cos(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::pseries::normal(), GiNaC::power::normal(), GiNaC::numeric::normal(), GiNaC::basic::normal(), GiNaC::power::to_polynomial(), GiNaC::numeric::to_polynomial(), GiNaC::basic::to_polynomial(), GiNaC::power::to_rational(), GiNaC::numeric::to_rational(), and GiNaC::basic::to_rational().

5.1.3.579 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().

5.1.3.580 frac_cancel()

```
static ex GiNaC::frac_cancel (
            const ex & n,
            const ex & d )  [static]
```

Fraction cancellation.

Parameters

n	numerator
d	denominator

Returns

cancelled fraction {n, d} as a list

References _ex1, _ex_1, _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().

5.1.3.581 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(), gcd(), 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().

5.1.3.582 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().

5.1.3.583 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.

5.1.3.584 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [25/32]

```
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.

5.1.3.585 make_real_float()

```
static const cln::cl_F GiNaC::make_real_float (
             const cln::cl_idecoded_float & dec )  [static]
```

Construct a floating point number from sign, mantissa, and exponent.

References x.

Referenced by read_real_float().

5.1.3.586 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().

5.1.3.587 GINAC_BIND_UNARCHIVER() [36/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            numeric  )
```

5.1.3.588 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().

5.1.3.589 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 abs(), c, and x.

Referenced by GiNaC::numeric::print_numeric(), and print_real_cl_N().

5.1.3.590 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().

5.1.3.591 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().

5.1.3.592 coerce()

```
template<typename T1 , typename T2 >
static bool GiNaC::coerce (
             T1 & dst,
             const T2 & arg )  [inline], [static]
```

Referenced by print_real_cl_N().

5.1.3.593 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

5.1.3.594 coerce< unsigned int, cln::cl_I >()

```
template<>
bool GiNaC::coerce< unsigned int, cln::cl_I > (
            unsigned int & dst,
            const cln::cl_I & arg )  [inline]
```

5.1.3.595 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().

5.1.3.596 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_value(), 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().

5.1.3.597 log()

```
const numeric GiNaC::log (
            const numeric & x )
```

Natural logarithm.

Parameters

x	complex number

Returns

arbitrary precision numerical log(x).

Exceptions

pole_error("log()	logarithmic pole",0)

References GiNaC::ex::is_zero(), and x.

Referenced by atan(), 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_(), Li2_deriv(), Li2_eval(), Li2_projection(), 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().

5.1.3.598 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(), tan_series(), and tgamma().

5.1.3.599 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().

5.1.3.600 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().

5.1.3.601 asin()

```
const numeric GiNaC::asin (
             const numeric & x )
```

Numeric inverse sine (trigonometric function).

Returns

arbitrary precision numerical asin(x).

References x.

Referenced by asin_conjugate(), asin_eval(), asin_evalf(), cos_eval(), sin_eval(), and tan_eval().

5.1.3.602 acos()

```
const numeric GiNaC::acos (
            const numeric & x )
```

Numeric inverse cosine (trigonometric function).

Returns

arbitrary precision numerical acos(x).

References x.

Referenced by acos_conjugate(), acos_eval(), acos_evalf(), cos_eval(), sin_eval(), and tan_eval().

5.1.3.603 atan() [1/2]

```
const numeric GiNaC::atan (
            const numeric & x )
```

Numeric arcustangent.

Parameters

x	complex number

Returns

atan(x)

Exceptions

pole_error("atan()	logarithmic pole",0) if x==I or x==-I.

References _num1_p, abs(), atan(), 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().

5.1.3.604 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

y	complex number
x	complex number

Returns

-I∗log((x+I∗y)/sqrt(x^2+y^2)), which is equal to atan(y/x) if y and x are both real.

Exceptions

pole_error("atan()	logarithmic pole",0) if y/x==+I or y/x==-I.

References _num0_p, GiNaC::numeric::is_real(), GiNaC::numeric::is_zero(), GiNaC::ex::is_zero(), log(), sqrt(), GiNaC::numeric::to_cl_N(), and x.

Referenced by atan().

5.1.3.605 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().

5.1.3.606 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().

5.1.3.607 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().

5.1.3.608 asinh()

```
const numeric GiNaC::asinh (
             const numeric & x )
```

Numeric inverse hyperbolic sine (trigonometric function).

Returns

arbitrary precision numerical asinh(x).

References x.

Referenced by asinh_conjugate(), asinh_eval(), asinh_evalf(), cosh_eval(), sinh_eval(), and tanh_eval().

5.1.3.609 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().

5.1.3.610 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().

5.1.3.611 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.

5.1.3.612 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 abs(), Li2_series(), log(), x, and zeta().

Referenced by Li2_().

5.1.3.613 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 abs(), Li2_projection(), log(), value, and zeta().

Referenced by Li2().

5.1.3.614 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().

5.1.3.615 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.

Referenced by Li2_(), and Li2_projection().

5.1.3.616 guess_precision()

```
static cln::float_format_t GiNaC::guess_precision (
            const cln::cl_N & x )  [static]
```

References x.

Referenced by lgamma(), and tgamma().

5.1.3.617 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(), log(), sin(), GiNaC::lanczos_coeffs::sufficiently_accurate(), and x.

Referenced by beta_evalf(), lgamma(), lgamma_conjugate(), lgamma_eval(), lgamma_evalf(), and lgamma_series().

5.1.3.618 lgamma() [2/2]

```
const numeric GiNaC::lgamma (
             const numeric & x )
```

References lgamma(), and x.

5.1.3.619 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(), sin(), sqrt(), GiNaC::lanczos_coeffs::sufficiently_accurate(), and x.

Referenced by beta_eval(), beta_series(), psi2_eval(), tgamma(), tgamma_conjugate(), tgamma_deriv(), tgamma_eval(), tgamma_evalf(), and tgamma_series().

5.1.3.620 tgamma() [2/2]

```
const numeric GiNaC::tgamma (
             const numeric & x )
```

References tgamma(), and x.

5.1.3.621 psi() [3/4]

```
const numeric GiNaC::psi (
             const numeric & x )
```

The psi function (aka polygamma function).

This is only a stub!

5.1.3.622 psi() [4/4]

```
const numeric GiNaC::psi (
             const numeric & n,
             const numeric & x )
```

The psi functions (aka polygamma functions).

This is only a stub!

5.1.3.623 factorial()

```
const numeric GiNaC::factorial (
             const numeric & n )
```

Factorial combinatorial function.

Parameters

n	integer argument >= 0

Exceptions

range_error	(argument must be integer >= 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().

5.1.3.624 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

n	integer argument >= -1

Returns

n!! == n ∗ (n-2) ∗ (n-4) ∗ ... ∗ ({1|2}) with 0!! == (-1)!! == 1

Exceptions

range_error	(argument must be integer >= -1)

References _num1_p, _num_1_p, and n.

Referenced by tgamma_eval().

5.1.3.625 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 negative, the formula binomial(n,k) == (-1)^k∗binomial(k-n-1,k) is used to compute the result.

References _num0_p, _num1_p, _num_1_p, k, n, and GiNaC::numeric::power().

Referenced by binomial_conjugate(), binomial_eval(), binomial_evalf(), binomial_real_part(), binomial_sym(), EllipticE_series(), EllipticK_series(), GiNaC::power::expand_add(), GiNaC::power::imag_part(), and GiNaC↩ ::power::real_part().

5.1.3.626 bernoulli()

```
const numeric GiNaC::bernoulli (
              const numeric & nn )
```

Bernoulli number.

The nth Bernoulli number is the coefficient of x^n/n! in the expansion of the function x/(e^x-1).

Returns

the nth Bernoulli number (a rational number).

Exceptions

range_error	(argument must be integer >= 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().

5.1.3.627 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

range_error	(argument must be an integer)

References _num0_p, m, and n.

5.1.3.628 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(), H_evalf(), GiNaC::power::imag_part(), GiNaC::mul::integer_content(), GiNaC::numeric::integer_content(), is_discriminant_of_quadratic_number_field(), Li2_(), Li2_projection(), log_real_part(), GiNaC::add::max_coefficient(), GiNaC::mul::max_coefficient(), GiNaC::numeric::max_coefficient(), GiNaC::matrix::pivot(), print_real_number(), GiNaC::power::real_part(), tgamma_eval(), GiNaC::ex::unitcontprim(), and zeta1_eval().

5.1.3.629 mod()

```
const numeric GiNaC::mod (
            const numeric & a,
            const numeric & b )
```

Modulus (in positive representation).

In general, mod(a,b) has the sign of b or is zero, and rem(a,b) has the sign of a or is zero. This is different from Maple's modp, where the sign of b is ignored. It is in agreement with Mathematica's Mod.

Returns

a mod b in the range [0,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(), smod(), and tan_eval().

5.1.3.630 smod()

```
const numeric GiNaC::smod (
            const numeric & a_,
            const numeric & b_ )
```

Modulus (in symmetric representation).

Returns

a mod b in the range [-iquo(abs(b),2), iquo(abs(b),2)].

References _num0_p, GiNaC::numeric::is_integer(), m, mod(), and GiNaC::numeric::to_cl_N().

Referenced by GiNaC::add::smod(), GiNaC::mul::smod(), and GiNaC::numeric::smod().

5.1.3.631 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

overflow_error	(division by zero) if b is zero.

References _num0_p, GiNaC::numeric::is_integer(), GiNaC::numeric::is_zero(), rem(), and GiNaC::numeric::to_cl_N().

Referenced by GiNaC::Eisenstein_h_kernel::coefficient_a0(), GiNaC::Eisenstein_h_kernel::coefficient_an(), and ifactor().

5.1.3.632 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 obeyes 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

overflow_error	(division by zero) if b is zero.

References _num0_p, GiNaC::numeric::is_integer(), GiNaC::numeric::is_zero(), and GiNaC::numeric::to_cl_N().

5.1.3.633 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

overflow_error	(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().

5.1.3.634 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 obeyes the relation r == a - iquo(a,b,r)∗b.

Returns

truncated quotient of a/b and remainder stored in r if both are integer, 0 otherwise.

Exceptions

overflow_error	(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().

5.1.3.635 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().

5.1.3.636 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().

Referenced by GiNaC::numeric::denom(), and GiNaC::numeric::numer().

5.1.3.637 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

x	numeric argument

Returns

square root of x. Branch cut along negative real axis, the negative real axis itself where imag(x)==0 and real(x)<0 belongs to the upper part where imag(x)>0.

References x.

Referenced by atan(), 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().

5.1.3.638 isqrt()

```
const numeric GiNaC::isqrt (
             const numeric & x )
```

Integer numeric square root.

References _num0_p, and x.

Referenced by heur_gcd_z().

5.1.3.639 PiEvalf()

```
ex GiNaC::PiEvalf ( )
```

Floating point evaluation of Archimedes' constant Pi.

5.1.3.640 EulerEvalf()

```
ex GiNaC::EulerEvalf ( )
```

Floating point evaluation of Euler's constant gamma.

5.1.3.641 CatalanEvalf()

```
ex GiNaC::CatalanEvalf ( )
```

Floating point evaluation of Catalan's constant.

5.1.3.642 operator<<() [6/16]

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

References GiNaC::_numeric_digits::print().

5.1.3.643 GINAC_DECLARE_UNARCHIVER() [38/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            numeric  )
```

5.1.3.644 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_an(), 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(), 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::Kronecker_dz_kernel::series_coeff_impl(), sprem(), sqrfree_parfrac(), sr_gcd(), GiNaC::power::subs(), tgamma_eval(), GiNaC::power::to_polynomial(), GiNaC::power::to_rational(), and zeta1_eval().

5.1.3.645 inverse() [3/3]

```
const numeric GiNaC::inverse (
            const numeric & x )  [inline]
```

References x.

5.1.3.646 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().

5.1.3.647 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().

5.1.3.648 is_zero() [2/2]

```
bool GiNaC::is_zero (
            const numeric & x )  [inline]
```

References GiNaC::ex::is_zero(), and x.

5.1.3.649 is_positive()

```
bool GiNaC::is_positive (
            const numeric & x )  [inline]
```

References x.

Referenced by beta_eval(), and GiNaC::mul::info().

5.1.3.650 is_negative()

```
bool GiNaC::is_negative (
            const numeric & x )  [inline]
```

References x.

Referenced by GiNaC::power::do_print_latex(), GiNaC::power::eval(), frac_cancel(), and GiNaC::mul::info().

5.1.3.651 is_integer()

```
bool GiNaC::is_integer (
            const numeric & x )  [inline]
```

References x.

Referenced by beta_eval(), binomial_eval(), GiNaC::power::coeff(), csgn_power(), GiNaC::power::degree(), GiNaC::power::expand(), GiNaC::power::ldegree(), GiNaC::pseries::power_const(), psi1_eval(), psi2_eval(), replace_with_symbol(), and GiNaC::power::series().

5.1.3.652 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().

5.1.3.653 is_nonneg_integer()

```
bool GiNaC::is_nonneg_integer (
            const numeric & x )  [inline]
```

References x.

5.1.3.654 is_even()

```
bool GiNaC::is_even (
            const numeric & x )  [inline]
```

References x.

Referenced by abs_power(), and GiNaC::Kronecker_dz_kernel::series_coeff_impl().

5.1.3.655 is_odd()

```
bool GiNaC::is_odd (
            const numeric & x )  [inline]
```

References x.

5.1.3.656 is_prime()

```
bool GiNaC::is_prime (
            const numeric & x )  [inline]
```

References x.

5.1.3.657 is_rational()

```
bool GiNaC::is_rational (
            const numeric & x )  [inline]
```

References x.

Referenced by beta_eval(), and replace_with_symbol().

5.1.3.658 is_real()

```
bool GiNaC::is_real (
            const numeric & x )  [inline]
```

References x.

Referenced by beta_eval(), fsolve(), G2_evalf(), and Li2_series().

5.1.3.659 is_cinteger()

```
bool GiNaC::is_cinteger (
            const numeric & x )  [inline]
```

References x.

5.1.3.660 is_crational()

```
bool GiNaC::is_crational (
            const numeric & x )  [inline]
```

References x.

5.1.3.661 to_int()

```
int GiNaC::to_int (
            const numeric & x )  [inline]
```

References x.

Referenced by GiNaC::pseries::add_series(), GiNaC::power::coeff(), GiNaC::power::degree(), divide(), divide_in_z(), GiNaC::power::do_print_csrc(), GiNaC::matrix::eval_indexed(), GiNaC::tensdelta::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::su3d::eval_indexed(), GiNaC::spinmetric::eval_indexed(), GiNaC::tensepsilon::eval_indexed(), expand_dummy_sum(), gcd_pf_pow(), H_evalf(), iterated_integral_evalf_impl(), GiNaC::power::ldegree(), Li_eval(), Li_evalf(), GiNaC::pseries::mul_series(), GiNaC::basic::operator[](), GiNaC::pseries::power_const(), S_eval(), S_evalf(), GiNaC::pseries::series(), GiNaC::mul::series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::ELi_kernel::series_coeff_impl(), GiNaC::Ebar_kernel::series_coeff_impl(), GiNaC::Kronecker_dtau_kernel::series_coeff_impl(), sqrfree_parfrac(), and tryfactsubs().

5.1.3.662 to_long()

```
long GiNaC::to_long (
            const numeric & x )  [inline]
```

References x.

Referenced by GiNaC::power::imag_part(), and GiNaC::power::real_part().

5.1.3.663 to_double()

```
double GiNaC::to_double (
            const numeric & x )  [inline]
```

References x.

5.1.3.664 real()

```
const numeric GiNaC::real (
            const numeric & x )  [inline]
```

References x.

Referenced by cosh_eval(), csgn_series(), GiNaC::mul::info(), real_part_evalf(), sinh_eval(), step_series(), and tanh_eval().

5.1.3.665 imag()

```
const numeric GiNaC::imag (
            const numeric & x )  [inline]
```

References x.

Referenced by eta_eval(), eta_evalf(), G2_eval(), G2_evalf(), and imag_part_evalf().

5.1.3.666 numer() [2/2]

```
const numeric GiNaC::numer (
            const numeric & x )  [inline]
```

References GiNaC::ex::numer(), and x.

5.1.3.667 denom() [2/2]

```
const numeric GiNaC::denom (
            const numeric & x )  [inline]
```

References GiNaC::ex::denom(), and x.

5.1.3.668 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--(), and operator-=().

5.1.3.669 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/=().

5.1.3.670 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-(), and operator-=().

5.1.3.671 operator+() [1/4]

```
const ex GiNaC::operator+ (
             const ex & lh,
             const ex & rh )
```

References exadd().

5.1.3.672 operator-() [1/4]

```
const ex GiNaC::operator- (
             const ex & lh,
             const ex & rh )
```

References exadd(), and exminus().

5.1.3.673 operator*() [1/2]

```
const ex GiNaC::operator* (
             const ex & lh,
             const ex & rh )
```

References exmul().

5.1.3.674 operator/() [1/2]

```
const ex GiNaC::operator/ (
             const ex & lh,
             const ex & rh )
```

References _ex_1, and exmul().

5.1.3.675 operator+() [2/4]

```
const numeric GiNaC::operator+ (
             const numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::add().

5.1.3.676 operator-() [2/4]

```
const numeric GiNaC::operator- (
             const numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::sub().

5.1.3.677 operator∗() [2/2]

```
const numeric GiNaC::operator* (
             const numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::mul().

5.1.3.678 operator/() [2/2]

```
const numeric GiNaC::operator/ (
             const numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::div().

5.1.3.679 operator+=() [1/2]

```
ex & GiNaC::operator+= (
             ex & lh,
             const ex & rh )
```

References exadd().

5.1.3.680 operator-=() [1/2]

```
ex & GiNaC::operator-= (
             ex & lh,
             const ex & rh )
```

References exadd(), and exminus().

5.1.3.681 operator*=() [1/2]

```
ex & GiNaC::operator*= (
             ex & lh,
             const ex & rh )
```

References exmul().

5.1.3.682 operator/=() [1/2]

```
ex & GiNaC::operator/= (
             ex & lh,
             const ex & rh )
```

References _ex_1, and exmul().

5.1.3.683 operator+=() [2/2]

```
numeric & GiNaC::operator+= (
             numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::add().

5.1.3.684 operator-=() [2/2]

```
numeric & GiNaC::operator-= (
             numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::sub().

5.1.3.685 operator*=() [2/2]

```
numeric & GiNaC::operator*= (
             numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::mul().

5.1.3.686 operator/=() [2/2]

```
numeric & GiNaC::operator/= (
             numeric & lh,
             const numeric & rh )
```

References GiNaC::numeric::div().

5.1.3.687 operator+() [3/4]

```
const ex GiNaC::operator+ (
             const ex & lh )
```

5.1.3.688 operator-() [3/4]

```
const ex GiNaC::operator- (
             const ex & lh )
```

References exminus().

5.1.3.689 operator+() [4/4]

```
const numeric GiNaC::operator+ (
             const numeric & lh )
```

5.1.3.690 operator-() [4/4]

```
const numeric GiNaC::operator- (
             const numeric & lh )
```

References _num_1_p, and GiNaC::numeric::mul().

5.1.3.691 operator++() [1/4]

```
ex & GiNaC::operator++ (
             ex & rh )
```

Expression prefix increment.

Adds 1 and returns incremented ex.

References _ex1, and exadd().

5.1.3.692 operator--() [1/4]

```
ex & GiNaC::operator-- (
             ex & rh )
```

Expression prefix decrement.

Subtracts 1 and returns decremented ex.

References _ex_1, and exadd().

5.1.3.693 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().

5.1.3.694 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().

5.1.3.695 operator++() [3/4]

```
numeric & GiNaC::operator++ (
             numeric & rh )
```

Numeric prefix increment.

Adds 1 and returns incremented number.

References _num1_p, and GiNaC::numeric::add().

5.1.3.696 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().

5.1.3.697 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().

5.1.3.698 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().

5.1.3.699 operator==()

```
const relational GiNaC::operator== (
             const ex & lh,
             const ex & rh )
```

References GiNaC::relational::equal.

5.1.3.700 operator"!=()

```
const relational GiNaC::operator!= (
            const ex & lh,
            const ex & rh )
```

References GiNaC::relational::not_equal.

5.1.3.701 operator<()

```
const relational GiNaC::operator< (
            const ex & lh,
            const ex & rh )
```

References GiNaC::relational::less.

5.1.3.702 operator<=()

```
const relational GiNaC::operator<= (
            const ex & lh,
            const ex & rh )
```

References GiNaC::relational::less_or_equal.

5.1.3.703 operator>()

```
const relational GiNaC::operator> (
            const ex & lh,
            const ex & rh )
```

References GiNaC::relational::greater.

5.1.3.704 operator>=()

```
const relational GiNaC::operator>= (
            const ex & lh,
            const ex & rh )
```

References GiNaC::relational::greater_or_equal.

5.1.3.705 my_ios_index()

```
static int GiNaC::my_ios_index ( )  [static]
```

Referenced by get_print_context(), and set_print_context().

5.1.3.706 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().

5.1.3.707 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<<(), and set_print_options().

5.1.3.708 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(), GiNaC::print_context::options, and options.

Referenced by csrc(), csrc_cl_N(), csrc_double(), csrc_float(), dflt(), latex(), python(), python_repr(), set_print_↩ options(), and tree().

5.1.3.709 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().

5.1.3.710 set_print_options()

```
static void GiNaC::set_print_options (
             std::ostream & s,
             unsigned options )  [static]
```

References get_print_context(), GiNaC::print_context::options, options, and set_print_context().

Referenced by dflt(), index_dimensions(), and no_index_dimensions().

5.1.3.711 operator<<() [7/16]

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

References get_print_context(), and GiNaC::ex::print().

5.1.3.712 operator>>() [3/3]

```
std::istream & GiNaC::operator>> (
             std::istream & is,
             ex & e )
```

5.1.3.713 dflt()

```
std::ostream & GiNaC::dflt (
             std::ostream & os )
```

References set_print_context(), and set_print_options().

5.1.3.714 latex()

```
std::ostream & GiNaC::latex (
             std::ostream & os )
```

References set_print_context().

5.1.3.715 python()

```
std::ostream & GiNaC::python (
             std::ostream & os )
```

References set_print_context().

5.1.3.716 python_repr()

```
std::ostream & GiNaC::python_repr (
             std::ostream & os )
```

References set_print_context().

5.1.3.717 tree()

```
std::ostream & GiNaC::tree (
             std::ostream & os )
```

References set_print_context().

Referenced by GiNaC::class_info< OPT >::dump_hierarchy().

5.1.3.718 csrc()

```
std::ostream & GiNaC::csrc (
             std::ostream & os )
```

References set_print_context().

5.1.3.719 csrc_float()

```
std::ostream & GiNaC::csrc_float (
             std::ostream & os )
```

References set_print_context().

5.1.3.720 csrc_double()

```
std::ostream & GiNaC::csrc_double (
            std::ostream & os )
```

References set_print_context().

5.1.3.721 csrc_cl_N()

```
std::ostream & GiNaC::csrc_cl_N (
            std::ostream & os )
```

References set_print_context().

5.1.3.722 index_dimensions()

```
std::ostream & GiNaC::index_dimensions (
            std::ostream & os )
```

References get_print_options(), GiNaC::print_options::print_index_dimensions, and set_print_options().

5.1.3.723 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().

5.1.3.724 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [26/32]

```
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  )
```

5.1.3.725 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(), and x.

Referenced by GiNaC::power::do_print_csrc().

5.1.3.726 GINAC_BIND_UNARCHIVER() [37/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             power  )
```

5.1.3.727 GINAC_DECLARE_UNARCHIVER() [39/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             power  )
```

5.1.3.728 pow() [2/3]

```
ex GiNaC::pow (
             const ex & b,
             const ex & e )  [inline]
```

Symbolic exponentiation.

Returns a power-object as a new expression.

Parameters

b	the basis expression
e	the exponent expression

5.1.3.729 pow() [3/3]

```
template<typename T1 , typename T2 >
ex GiNaC::pow (
```

```
            const T1 & b,
            const T2 & e )  [inline]
```

5.1.3.730 sqrt() [2/2]

```
ex GiNaC::sqrt (
            const ex & a )  [inline]
```

Square root expression.

Returns a power-object with exponent 1/2.

References _ex1_2.

5.1.3.731 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.

5.1.3.732 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [27/32]

```
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  )
```

5.1.3.733 GINAC_BIND_UNARCHIVER() [38/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            pseries  )
```

5.1.3.734 GINAC_DECLARE_UNARCHIVER() [40/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
            pseries  )
```

5.1.3.735 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

e	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().

5.1.3.736 is_terminating()

```
bool GiNaC::is_terminating (
            const pseries & s )  [inline]
```

References GiNaC::pseries::is_terminating().

5.1.3.737 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.

5.1.3.738 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 )
```

Add or replace a print method.

References options.

5.1.3.739 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 )
```

Add or replace a print method.

References options.

5.1.3.740 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [28/32]

```
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  )
```

5.1.3.741 GINAC_BIND_UNARCHIVER() [39/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             relational  )
```

5.1.3.742 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().

5.1.3.743 GINAC_DECLARE_UNARCHIVER() [41/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             relational  )
```

5.1.3.744 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [29/32]

```
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.

5.1.3.745 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().

5.1.3.746 GINAC_BIND_UNARCHIVER() [40/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             symbol  )
```

5.1.3.747 GINAC_BIND_UNARCHIVER() [41/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             realsymbol  )
```

5.1.3.748 GINAC_BIND_UNARCHIVER() [42/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             possymbol  )
```

5.1.3.749 GINAC_DECLARE_UNARCHIVER() [42/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             symbol  )
```

5.1.3.750 GINAC_DECLARE_UNARCHIVER() [43/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             realsymbol  )
```

5.1.3.751 GINAC_DECLARE_UNARCHIVER() [44/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             possymbol  )
```

5.1.3.752 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [30/32]

```
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.

5.1.3.753 GINAC_BIND_UNARCHIVER() [43/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
             symmetry  )
```

5.1.3.754 index0()

```
static const symmetry& GiNaC::index0 ( )  [static]
```

Referenced by antisymmetric2(), antisymmetric3(), antisymmetric4(), symmetric2(), symmetric3(), and symmetric4().

5.1.3.755 index1()

```
static const symmetry& GiNaC::index1 ( )  [static]
```

Referenced by antisymmetric2(), antisymmetric3(), antisymmetric4(), symmetric2(), symmetric3(), and symmetric4().

5.1.3.756 index2()

```
static const symmetry& GiNaC::index2 ( )  [static]
```

Referenced by antisymmetric3(), antisymmetric4(), symmetric3(), and symmetric4().

5.1.3.757 index3()

```
static const symmetry& GiNaC::index3 ( )  [static]
```

Referenced by antisymmetric4(), and symmetric4().

5.1.3.758 not_symmetric()

```
const symmetry & GiNaC::not_symmetric ( )
```

Referenced by GiNaC::indexed::read_archive().

5.1.3.759 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().

5.1.3.760 symmetric3()

```
const symmetry & GiNaC::symmetric3 ( )
```

References index0(), index1(), index2(), and GiNaC::symmetry::symmetric.

Referenced by color_d().

5.1.3.761 symmetric4()

```
const symmetry & GiNaC::symmetric4 ( )
```

References index0(), index1(), index2(), index3(), and GiNaC::symmetry::symmetric.

5.1.3.762 antisymmetric2()

```
const symmetry & GiNaC::antisymmetric2 ( )
```

References GiNaC::symmetry::antisymmetric, index0(), and index1().

Referenced by epsilon_tensor(), and spinor_metric().

5.1.3.763 antisymmetric3()

```
const symmetry & GiNaC::antisymmetric3 ( )
```

References GiNaC::symmetry::antisymmetric, index0(), index1(), and index2().

Referenced by color_f(), and epsilon_tensor().

5.1.3.764 antisymmetric4()

```
const symmetry & GiNaC::antisymmetric4 ( )
```

References GiNaC::symmetry::antisymmetric, index0(), index1(), index2(), and index3().

Referenced by lorentz_eps().

5.1.3.765 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

v	Start of expression vector
symm	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

References GiNaC::symmetry::antisymmetric, GiNaC::ex::begin(), GiNaC::symmetry::cyclic, cyclic_permutation(), GiNaC::ex::end(), GINAC_ASSERT, last, permutation_sign(), shaker_sort(), symm(), and GiNaC::symmetry::symmetric.

Referenced by GiNaC::indexed::eval(), and GiNaC::function::eval().

5.1.3.766 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 GiNaC::ex::antisymmetrize(), antisymmetrize(), canonicalize(), GiNaC::indexed::read_archive(), GiNaC::ex::symmetrize(), and symmetrize().

5.1.3.767 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().

5.1.3.768 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().

5.1.3.769 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().

5.1.3.770 GINAC_DECLARE_UNARCHIVER() [45/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             symmetry  )
```

5.1.3.771 sy_none() [1/4]

```
symmetry GiNaC::sy_none ( )  [inline]
```

5.1.3.772 sy_none() [2/4]

```
symmetry GiNaC::sy_none (
             const symmetry & c1,
             const symmetry & c2 )  [inline]
```

References GiNaC::symmetry::none.

5.1.3.773 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.

5.1.3.774 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.

5.1.3.775 sy_symm() [1/4]

```
symmetry GiNaC::sy_symm ( )  [inline]
```

References GiNaC::symmetry::set_type(), and GiNaC::symmetry::symmetric.

Referenced by GiNaC::indexed::read_archive().

5.1.3.776 sy_symm() [2/4]

```
symmetry GiNaC::sy_symm (
             const symmetry & c1,
             const symmetry & c2 )  [inline]
```

References GiNaC::symmetry::symmetric.

5.1.3.777 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.

5.1.3.778 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.

5.1.3.779 sy_anti() [1/4]

```
symmetry GiNaC::sy_anti ( )  [inline]
```

References GiNaC::symmetry::antisymmetric, and GiNaC::symmetry::set_type().

Referenced by GiNaC::indexed::read_archive().

5.1.3.780 sy_anti() [2/4]

```
symmetry GiNaC::sy_anti (
            const symmetry & c1,
            const symmetry & c2 )  [inline]
```

References GiNaC::symmetry::antisymmetric.

5.1.3.781 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.

5.1.3.782 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.

5.1.3.783 sy_cycl() [1/4]

```
symmetry GiNaC::sy_cycl ( )  [inline]
```

References GiNaC::symmetry::cyclic, and GiNaC::symmetry::set_type().

5.1.3.784 sy_cycl() [2/4]

```
symmetry GiNaC::sy_cycl (
             const symmetry & c1,
             const symmetry & c2 )  [inline]
```

References GiNaC::symmetry::cyclic.

5.1.3.785 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.

5.1.3.786 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.

5.1.3.787 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().

5.1.3.788 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().

5.1.3.789 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().

5.1.3.790 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [31/32]

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

5.1.3.791 print_func< print_dflt >() [3/3]

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

References GiNaC::status_flags::evaluated, GiNaC::status_flags::expanded, and GiNaC::basic::setflag().

5.1.3.792 GINAC_BIND_UNARCHIVER() [44/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            minkmetric  )
```

5.1.3.793 GINAC_BIND_UNARCHIVER() [45/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            tensepsilon  )
```

5.1.3.794 GINAC_BIND_UNARCHIVER() [46/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
            tensdelta  )
```

5.1.3.795 GINAC_BIND_UNARCHIVER() [47/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              tensmetric  )
```

5.1.3.796 GINAC_BIND_UNARCHIVER() [48/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              spinmetric  )
```

5.1.3.797 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

i1	First index
i2	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().

5.1.3.798 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

i1	First index
i2	Second index

Returns

newly constructed metric tensor

References symmetric2().

Referenced by GiNaC::tensepsilon::contract_with().

5.1.3.799 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 diag(1,-1,-1,...) (negative signature, the default) or diag(-1,1,1,...) (positive signature).

Parameters

i1	First index
i2	Second index
pos_sig	Whether the signature is positive

Returns

newly constructed Lorentz metric tensor

References symmetric2().

Referenced by GiNaC::tensepsilon::contract_with().

5.1.3.800 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 [[[[0, 1]], [[-1, 0]]]].

Parameters

i1	First index
i2	Second index

Returns

newly constructed spinor metric tensor

References antisymmetric2().

5.1.3.801 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

i1	First index
i2	Second index

Returns

newly constructed epsilon tensor

References _ex2, antisymmetric2(), GiNaC::basic::hold(), GiNaC::ex::is_equal(), and GiNaC::ex::op().

5.1.3.802 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

i1	First index
i2	Second index
i3	Third index

Returns

newly constructed epsilon tensor

References _ex3, antisymmetric3(), GiNaC::basic::hold(), GiNaC::ex::is_equal(), and GiNaC::ex::op().

5.1.3.803 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

i1	First index
i2	Second index
i3	Third index
i4	Fourth index
pos_sig	Whether the signature of the metric is positive

Returns

newly constructed epsilon tensor

References _ex4, antisymmetric4(), GiNaC::basic::hold(), GiNaC::ex::is_equal(), and GiNaC::ex::op().

5.1.3.804 GINAC_DECLARE_UNARCHIVER() [46/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             tensdelta  )
```

5.1.3.805 GINAC_DECLARE_UNARCHIVER() [47/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             tensmetric  )
```

5.1.3.806 GINAC_DECLARE_UNARCHIVER() [48/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             minkmetric  )
```

5.1.3.807 GINAC_DECLARE_UNARCHIVER() [49/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             spinmetric  )
```

5.1.3.808 GINAC_DECLARE_UNARCHIVER() [50/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
             tensepsilon  )
```

5.1.3.809 log2()

```
unsigned GiNaC::log2 (
             unsigned n )
```

Integer binary logarithm.

References k, and n.

Referenced by GiNaC::remember_table::remember_table().

5.1.3.810 multinomial_coefficient()

```
const numeric GiNaC::multinomial_coefficient (
             const std::vector< unsigned > & p )
```

Compute the multinomial coefficient n!/(p1!∗p2!∗...∗pk!) where n = p1+p2+...+pk, i.e.

p is a partition of n.

References GiNaC::numeric::div(), factorial(), and n.

Referenced by GiNaC::power::expand_add().

5.1.3.811 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::idx::calchash(), GiNaC::symmetry::calchash(), GiNaC::relational::calchash(), GiNaC↩ ::basic::calchash(), and GiNaC::function::calchash().

5.1.3.812 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().

5.1.3.813 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::wildcard::calchash(), GiNaC::constant::calchash(), GiNaC::symbol::calchash(), GiNaC↩ ::numeric::calchash(), GiNaC::function::calchash(), and make_hash_seed().

5.1.3.814 permutation_sign() [1/2]

```
template<class It >
int GiNaC::permutation_sign (
             It first,
             It last )
```

References last, swap(), and std::swap().

Referenced by canonicalize(), GiNaC::matrix::determinant(), GiNaC::tensepsilon::eval_indexed(), and symm().

5.1.3.815 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.

5.1.3.816 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 canonicalize(), find_free_and_dummy(), and rename_dummy_indices().

5.1.3.817 cyclic_permutation()

```
template<class It , class Swap >
void GiNaC::cyclic_permutation (
             It first,
             It last,
             It new_first,
             Swap swapit )
```

References last.

Referenced by canonicalize().

5.1.3.818 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<<().

5.1.3.819 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.

5.1.3.820 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.821 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.822 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.823 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.824 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.825 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.826 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.827 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.828 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($n_0, n_1, ...$).

References format_index_value().

5.1.3.829 GINAC_IMPLEMENT_REGISTERED_CLASS_OPT() [32/32]

```
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.

5.1.3.830 GINAC_BIND_UNARCHIVER() [49/49]

```
GiNaC::GINAC_BIND_UNARCHIVER (
              wildcard  )
```

5.1.3.831 haswild()

```
bool GiNaC::haswild (
              const ex & x )
```

Check whether x has a wildcard anywhere as a subexpression.

References GiNaC::ex::nops(), GiNaC::ex::op(), and x.

Referenced by GiNaC::integral::eval().

5.1.3.832 GINAC_DECLARE_UNARCHIVER() [51/51]

```
GiNaC::GINAC_DECLARE_UNARCHIVER (
              wildcard  )
```

5.1.3.833 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.

(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(), GiNaC::Kronecker_dz_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'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'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 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().

5.1.4.15 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().

5.1.4.16 next_print_context_id

```
unsigned GiNaC::next_print_context_id = 0
```

Next free ID for print_context types.

5.1.4.17 version_major

```
const int GiNaC::version_major = GINACLIB_MAJOR_VERSION
```

5.1.4.18 version_minor

```
const int GiNaC::version_minor = GINACLIB_MINOR_VERSION
```

5.1.4.19 version_micro

```
const int GiNaC::version_micro = GINACLIB_MICRO_VERSION
```

5.1.4.20 _num_120_p

```
const numeric * GiNaC::_num_120_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.21 _ex_120

```
const ex GiNaC::_ex_120 = _ex_120
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.22 _num_60_p

```
const numeric * GiNaC::_num_60_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.23 _ex_60

```
const ex GiNaC::_ex_60 = _ex_60
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.24 _num_48_p

```
const numeric * GiNaC::_num_48_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.25 _ex_48

```
const ex GiNaC::_ex_48 = _ex_48
```

Referenced by Li2_eval(), Li_eval(), GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.26 _num_30_p

```
const numeric * GiNaC::_num_30_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.27 _ex_30

```
const ex GiNaC::_ex_30 = _ex_30
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.28 _num_25_p

```
const numeric * GiNaC::_num_25_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.29 _ex_25

```
const ex GiNaC::_ex_25 = _ex_25
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.30 _num_24_p

```
const numeric * GiNaC::_num_24_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.31 _ex_24

```
const ex GiNaC::_ex_24 = _ex_24
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.32 _num_20_p

```
const numeric * GiNaC::_num_20_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.33 _ex_20

```
const ex GiNaC::_ex_20 = _ex_20
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.34 _num_18_p

```
const numeric * GiNaC::_num_18_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.35 _ex_18

```
const ex GiNaC::_ex_18 = _ex_18
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.36 _num_15_p

```
const numeric * GiNaC::_num_15_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.37 _ex_15

```
const ex GiNaC::_ex_15 = _ex_15
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.38 _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().

5.1.4.39 _ex_12

```
const ex GiNaC::_ex_12 = _ex_12
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.40 _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().

5.1.4.41 _ex_11

```
const ex GiNaC::_ex_11 = _ex_11
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.42 _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().

5.1.4.43 _ex_10

```
const ex GiNaC::_ex_10 = _ex_10
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.44 _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().

5.1.4.45 _ex_9

```
const ex GiNaC::_ex_9 = _ex_9
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.46 _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().

5.1.4.47 _ex_8

```
const ex GiNaC::_ex_8 = _ex_8
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.48 _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().

5.1.4.49 _ex_7

```
const ex GiNaC::_ex_7 = _ex_7
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.50 _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().

5.1.4.51 _ex_6

```
const ex GiNaC::_ex_6 = _ex_6
```

Referenced by GiNaC::su3d::eval_indexed(), GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_↩ init().

5.1.4.52 _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().

5.1.4.53 _ex_5

```
const ex GiNaC::_ex_5 = _ex_5
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.54 _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().

5.1.4.55 _ex_4

```
const ex GiNaC::_ex_4 = _ex_4
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.56 _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().

5.1.4.57 _ex_3

```
const ex GiNaC::_ex_3 = _ex_3
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.58 _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().

5.1.4.59 _ex_2

```
const ex GiNaC::_ex_2 = _ex_2
```

Referenced by GiNaC::spinmetric::contract_with(), GiNaC::library_init::library_init(), and GiNaC::library_init↩ ::∼library_init().

5.1.4.60 _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--(), GiNaC::add::print_add(), GiNaC::mul::print_overall_coeff(), psi1_eval(), and psi2_eval().

5.1.4.61 _ex_1

```
const ex GiNaC::_ex_1 = _ex_1
```

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::power::do_print_csrc(), GiNaC::mul::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/=(), 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().

5.1.4.62 _num_1_2_p

```
const numeric * GiNaC::_num_1_2_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.63 _ex_1_2

```
const ex GiNaC::_ex_1_2 = _ex_1_2
```

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().

5.1.4.64 _num_1_3_p

```
const numeric * GiNaC::_num_1_3_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.65 _ex_1_3

```
const ex GiNaC::_ex_1_3 = _ex_1_3
```

Referenced by cos_eval(), GiNaC::su3d::eval_indexed(), GiNaC::library_init::library_init(), sin_eval(), and GiNaC::library_init::~library_init().

5.1.4.66 _num_1_4_p

```
const numeric * GiNaC::_num_1_4_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.67 _ex_1_4

```
const ex GiNaC::_ex_1_4 = _ex_1_4
```

Referenced by atan2_eval(), atan_eval(), cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and GiNaC::library_init::~library_init().

5.1.4.68 _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(), 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().

5.1.4.69 _ex0

```
const ex GiNaC::_ex0 = _ex0
```

Referenced by acos_eval(), acosh_eval(), GiNaC::add::add(), GiNaC::pseries::add_series(), asinh_eval(), atan2_eval(), atan_eval(), atan_series(), atanh_eval(), atanh_series(), beta_eval(), binomial_sym(), GiNaC::pseries::coeff(), GiNaC::add::coeff(), GiNaC::mul::coeff(), GiNaC::power::coeff(), GiNaC::ncmul::coeff(), GiNaC::numeric::coeff(), GiNaC::basic::coeff(), GiNaC::basic::collect(), color_trace(), GiNaC::ex::content(), cos_eval(), csgn_series(), GiNaC::constant::derivative(), GiNaC::idx::derivative(), GiNaC::symbol::derivative(), GiNaC::indexed::derivative(), GiNaC::basic::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::integral::eval(), GiNaC::mul::eval(), GiNaC::power::eval(), GiNaC::indexed::eval(), GiNaC::function::eval(), GiNaC::tensdelta::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::su3d::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::pseries::mul_series(), Order_eval(), GiNaC::pseries::power_const(), prem(), GiNaC::ex::primpart(), rem(), S_deriv(), S_eval(), GiNaC::symbol::series(), GiNaC::ex::series(), GiNaC::basic::series(), simplify_indexed_product(), 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().

5.1.4.70 _num1_4_p

```
const numeric * GiNaC::_num1_4_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.71 _ex1_4

```
const ex GiNaC::_ex1_4 = _ex1_4
```

Referenced by atan2_eval(), atan_eval(), cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and GiNaC::library_init::~library_init().

5.1.4.72 _num1_3_p

```
const numeric * GiNaC::_num1_3_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.73 _ex1_3

```
const ex GiNaC::_ex1_3 = _ex1_3
```

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().

5.1.4.74 _num1_2_p

```
const numeric * GiNaC::_num1_2_p
```

Referenced by GiNaC::library_init::library_init(), psi2_eval(), and tgamma_eval().

5.1.4.75 _ex1_2

```
const ex GiNaC::_ex1_2 = _ex1_2
```

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().

5.1.4.76 _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::add::integer_content(), GiNaC::basic::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++(), 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().

5.1.4.77 _ex1

```
const ex GiNaC::_ex1 = _ex1
```

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::power::coeff(), GiNaC::ncmul::coeff(), GiNaC::basic::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::matrix::contract_with(), GiNaC::su3t::contract_with(), GiNaC::su3f::contract_with(), GiNaC::su3d::contract_with(), GiNaC::spinmetric::contract_with(), GiNaC::tensepsilon::contract_with(), cos_eval(), cosh_eval(), GiNaC::mul::default_overall_coeff(), GiNaC::symbol::derivative(), GiNaC::mul::derivative(), GiNaC::power::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::power::eval(), GiNaC::ncmul::eval(), GiNaC::tensdelta::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::spinmetric::eval_indexed(), GiNaC::mul::evalm(), exp_eval(), GiNaC::mul::expair_needs_further_processing(), GiNaC::ncmul::expand(), GiNaC::power::expand(), GiNaC::mul::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::power::expand_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(), H_deriv(), H_eval(), hold_ncmul(), 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::mul::mul(), GiNaC::pseries::mul_series(), GiNaC::symbol::normal(), GiNaC::pseries::normal(), GiNaC::power::normal(), GiNaC::basic::normal(), 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::symbol::series(), GiNaC::pseries::series(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::basic::series(), simplify_indexed_product(), sin_eval(), sinh_eval(), GiNaC::add::split_ex_to_pair(), GiNaC::mul::split_ex_to_pair(), sprem(), sqrfree_parfrac(), 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().

5.1.4.78 _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().

5.1.4.79 _ex2

```
const ex GiNaC::_ex2 = _ex2
```

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(), simplify_indexed(), sin_eval(), sinh_eval(), tan_deriv(), tan_eval(), tanh_deriv(), tanh_eval(), and GiNaC::library_init::~library_init().

5.1.4.80 _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().

5.1.4.81 _ex3

```
const ex GiNaC::_ex3 = _ex3
```

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().

5.1.4.82 _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().

5.1.4.83 _ex4

```
const ex GiNaC::_ex4 = _ex4
```

Referenced by GiNaC::library_init::library_init(), lorentz_eps(), and GiNaC::library_init::~library_init().

5.1.4.84 _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().

5.1.4.85 _ex5

```
const ex GiNaC::_ex5 = _ex5
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and GiNaC::library_init::∼library_init().

5.1.4.86 _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().

5.1.4.87 _ex6

```
const ex GiNaC::_ex6 = _ex6
```

Referenced by cos_eval(), Li2_eval(), GiNaC::library_init::library_init(), sin_eval(), and GiNaC::library_init↩ ::∼library_init().

5.1.4.88 _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().

5.1.4.89 _ex7

```
const ex GiNaC::_ex7 = _ex7
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.90 _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().

5.1.4.91 _ex8

```
const ex GiNaC::_ex8 = _ex8
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.92 _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().

5.1.4.93 _ex9

```
const ex GiNaC::_ex9 = _ex9
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.94 _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().

5.1.4.95 _ex10

```
const ex GiNaC::_ex10 = _ex10
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.96 _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().

5.1.4.97 _ex11

```
const ex GiNaC::_ex11 = _ex11
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.98 _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().

5.1.4.99 _ex12

```
const ex GiNaC::_ex12 = _ex12
```

Referenced by Li2_eval(), GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.100 _num15_p

```
const numeric * GiNaC::_num15_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and tan_eval().

5.1.4.101 _ex15

```
const ex GiNaC::_ex15 = _ex15
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::∼library_init().

5.1.4.102 _num18_p

```
const numeric * GiNaC::_num18_p
```

Referenced by GiNaC::library_init::library_init(), and sin_eval().

5.1.4.103 _ex18

```
const ex GiNaC::_ex18 = _ex18
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.104 _num20_p

```
const numeric * GiNaC::_num20_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and tan_eval().

5.1.4.105 _ex20

```
const ex GiNaC::_ex20 = _ex20
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.106 _num24_p

```
const numeric * GiNaC::_num24_p
```

Referenced by cos_eval(), and GiNaC::library_init::library_init().

5.1.4.107 _ex24

```
const ex GiNaC::_ex24 = _ex24
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.108 _num25_p

```
const numeric * GiNaC::_num25_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and tan_eval().

5.1.4.109 _ex25

```
const ex GiNaC::_ex25 = _ex25
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.110 _num30_p

```
const numeric * GiNaC::_num30_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and tan_eval().

5.1.4.111 _ex30

```
const ex GiNaC::_ex30 = _ex30
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.112 _num48_p

```
const numeric * GiNaC::_num48_p
```

Referenced by GiNaC::library_init::library_init().

5.1.4.113 _ex48

```
const ex GiNaC::_ex48 = _ex48
```

Referenced by GiNaC::library_init::library_init(), and GiNaC::library_init::~library_init().

5.1.4.114 _num60_p

```
const numeric * GiNaC::_num60_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), and tan_eval().

5.1.4.115 _ex60

```
const ex GiNaC::_ex60 = _ex60
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), sin_eval(), tan_eval(), and GiNaC::library_init↩ ::∼library_init().

5.1.4.116 _num120_p

```
const numeric * GiNaC::_num120_p
```

Referenced by cos_eval(), GiNaC::library_init::library_init(), and sin_eval().

5.1.4.117 _ex120

```
const ex GiNaC::_ex120 = _ex120
```

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>`

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 dumber 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:

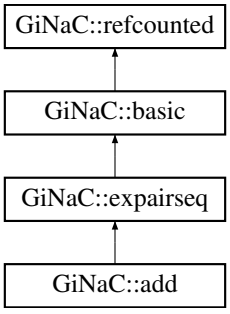
- numeric.h
- numeric.cpp

6.3 GiNaC::add Class Reference

Sum of expressions.

```
#include <add.h>
```

Inheritance diagram for GiNaC::add:



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

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
- ex thisexpairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false) const override
- expair split_ex_to_pair (const ex &e) const override
- 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

- 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

Friends

- class mul
- class power

Additional Inherited Members

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 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::ex::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.

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::ncmul::coeff(), GiNaC::ex::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) -> c
- +(x;0) -> 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::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]
```

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]
```

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]
```

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::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::denom(), GiNaC::ex::info(), GiNaC::ex::is_equal(), GiNaC::ex::is_zero(), GiNaC::numer(), 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 Function 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 printraw (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

e	the expression to be archived
name	name under which the expression is stored

References add_node(), atomize(), and exprs.

Referenced by 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

sym_lst	list of pre-defined symbols
name	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

sym_lst	list of pre-defined symbols
index	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

sym_lst	list of pre-defined symbols
name	receives the name of the expression
index	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, 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 Function 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(), GiNaC::operator<<(), GiNaC::operator>>(), printraw(), 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(), GiNaC::operator<<(), GiNaC::operator>>(), printraw(), 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(), GiNaC::operator<<(), GiNaC::operator>>(), printraw(), and unatomize().

6.4.6.4 inverse_atoms

```
std::map<std::string, archive_atom> GiNaC::archive::inverse_atoms  [mutable], [private]
```

Referenced by atomize(), clear(), and GiNaC::operator>>().

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>
```

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 }

  Property data types.
- 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 printraw (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::dynallocated, 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::bp, e, has_expression, props, PTYPE_BOOL, PTYPE_NODE, PTYPE_STRING, PTYPE_UNSIGNED, and GiNaC::archive::unatomize().

6.5.6 Friends And Related Function 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>
```

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:

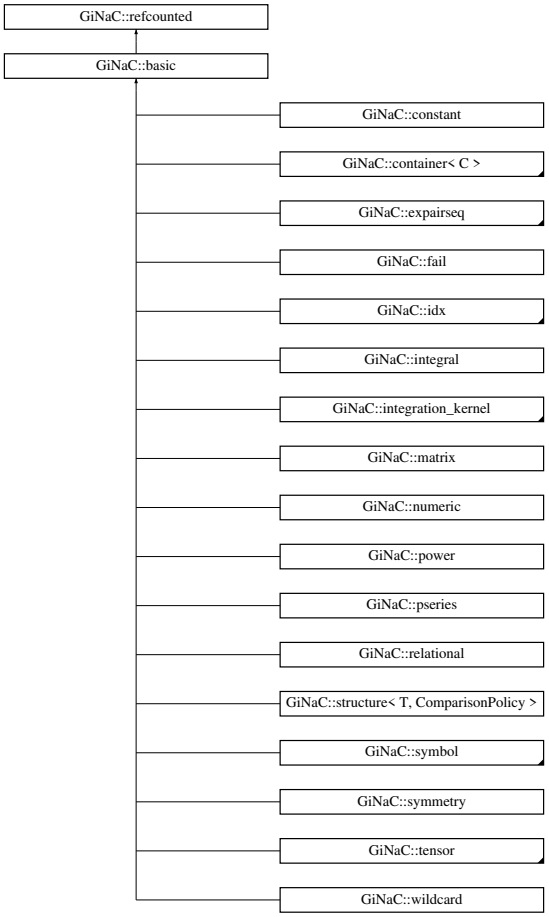
- archive.h

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:



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.

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::possymbol, and GiNaC::realsymbol.

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::function, GiNaC::indexed, GiNaC::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::expairseq, GiNaC::fderivative, GiNaC::ncmul, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::pseries, GiNaC::symbol, and GiNaC::integral.

Referenced by GiNaC::ex::construct_from_basic().

6.8.3.4 evalf()

```
ex GiNaC::basic::evalf ( ) const  [virtual]
```

Evaluate object numerically.

Reimplemented in GiNaC::function, GiNaC::numeric, GiNaC::power, GiNaC::mul, GiNaC::idx, GiNaC::pseries, GiNaC::constant, GiNaC::symbol, and GiNaC::integral.

References GiNaC::nops().

Referenced by GiNaC::Kronecker_dtau_kernel::get_numerical_value(), GiNaC::modular_form_kernel::is_numeric(), 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::structure< T, ComparisonPolicy >, GiNaC::ncmul, GiNaC::power, GiNaC::pseries, GiNaC::mul, GiNaC::add, and GiNaC::matrix.

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::pseries, and GiNaC::integral.

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::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::power, GiNaC::mul, GiNaC::relational, GiNaC::add, GiNaC::clifford, GiNaC::color, and GiNaC::integral.

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::tensepsilon, GiNaC::spinmetric, GiNaC::structure< T, ComparisonPolicy >, GiNaC::su3d, GiNaC::su3f, GiNaC::minkmetric, GiNaC::tensmetric, GiNaC::tensdelta, and GiNaC::matrix.

Referenced by GiNaC::indexed::eval().

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

c	print context object that describes the output formatting
level	value that is used to identify the precedence or indentation level for placing parentheses and formatting

Reimplemented in GiNaC::function, GiNaC::structure< T, ComparisonPolicy >, and GiNaC::fderivative.

References c.

Referenced by GiNaC::fderivative::print(), 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::function, GiNaC::indexed, GiNaC::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::numeric, GiNaC::expairseq, GiNaC::ncmul, GiNaC::relational, GiNaC::clifford, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::pseries, and GiNaC::integral.

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::function, GiNaC::tensepsilon, GiNaC::indexed, GiNaC::structure< T, ComparisonPolicy >, GiNaC::spinmetric, GiNaC::container< C >, GiNaC::minkmetric, GiNaC::numeric, GiNaC::tensmetric, GiNaC::expairseq, GiNaC::tensdelta, GiNaC::ncmul, GiNaC::relational, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::constant, GiNaC::idx, and GiNaC::symbol.

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::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, GiNaC::ELi_kernel, GiNaC::multiple_polylog_kernel, GiNaC::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::clifford, GiNaC::expairseq, GiNaC::relational, GiNaC::power, GiNaC::matrix, GiNaC::idx, GiNaC::integral, and GiNaC::pseries.

Referenced by GiNaC::matrix::eval_indexed(), GiNaC::tensmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::su3d::eval_indexed(), GiNaC::spinmetric::eval_indexed(), GiNaC::tensepsilon::eval_indexed(), 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::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, GiNaC::ELi_kernel, GiNaC::multiple_polylog_kernel, GiNaC::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::clifford, GiNaC::expairseq, GiNaC::relational, GiNaC::power, GiNaC::matrix, GiNaC::idx, GiNaC::integral, and GiNaC::pseries.

Referenced by GiNaC::matrix::eval_indexed(), GiNaC::tensmetric::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::su3d::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::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, GiNaC::ELi_kernel, GiNaC::multiple_polylog_kernel, GiNaC::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::clifford, GiNaC::matrix, and GiNaC::integral.

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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::power, and GiNaC::mul.

References GiNaC::ex::has(), GiNaC::match(), GiNaC::nops(), GiNaC::op(), and options.

Referenced by 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::structure< T, ComparisonPolicy >, GiNaC::expairseq, and GiNaC::wildcard.

References GiNaC::ex::match(), GiNaC::ex::nops(), GiNaC::nops(), GiNaC::ex::op(), and GiNaC::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::function, GiNaC::spinidx, GiNaC::structure< T, ComparisonPolicy >, GiNaC::varidx, GiNaC::fderivative, GiNaC::matrix, GiNaC::relational, GiNaC::idx, GiNaC::clifford, and GiNaC::color.

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::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::numeric, GiNaC::clifford, GiNaC::expairseq, GiNaC::power, GiNaC::relational, GiNaC::matrix, GiNaC::pseries, GiNaC::idx, 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(), and GiNaC::modular_form_kernel::is_numeric().

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::structure< T, ComparisonPolicy >, GiNaC::expairseq, GiNaC::relational, GiNaC::power, and GiNaC::idx.

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::numeric, GiNaC::symbol, GiNaC::power, GiNaC::mul, GiNaC::constant, and GiNaC::add.

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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::ncmul, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::pseries, and GiNaC::integral.

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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::ncmul, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::pseries, and GiNaC::integral.

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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::ncmul, GiNaC::power, GiNaC::mul, GiNaC::add, and GiNaC::pseries.

References GiNaC::_ex0, GiNaC::_ex1, and n.

Referenced by 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::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::indexed, GiNaC::mul, GiNaC::power, GiNaC::expairseq, GiNaC::add, GiNaC::ncmul, GiNaC::pseries, and GiNaC::integral.

References GiNaC::nops(), and options.

Referenced by 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

s	object(s) to sort in
distributed	recursive or distributed form (only used when s is a list)

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, and GiNaC::pseries.

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::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::indexed, GiNaC::numeric, GiNaC::power, GiNaC::ncmul, GiNaC::mul, GiNaC::fderivative, GiNaC::pseries, GiNaC::add, GiNaC::symbol, GiNaC::integral, GiNaC::constant, and GiNaC::idx.

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::function, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::structure< T, ComparisonPolicy >, GiNaC::power, GiNaC::fderivative, GiNaC::integral, GiNaC::integration_kernel, GiNaC::mul, GiNaC::add, GiNaC::pseries, 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::fderivative::series(), GiNaC::power::series(), and GiNaC::function::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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::power, GiNaC::mul, GiNaC::add, GiNaC::pseries, and GiNaC::symbol.

References GiNaC::_ex1, map(), GiNaC::subs_options::no_pattern, GiNaC::container< C >::nops(), nops(), GiNaC::container< C >::op(), GiNaC::ex::op(), GiNaC::replace_with_symbol(), and GiNaC::container< C >::subs().

Referenced by 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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::power, GiNaC::expairseq, 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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::power, GiNaC::expairseq, and GiNaC::symbol.

References GiNaC::replace_with_symbol().

6.8.3.38 integer_content()

```
numeric GiNaC::basic::integer_content ( ) const  [virtual]
```

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::mul, and GiNaC::add.

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::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::mul, and GiNaC::add.

6.8.3.40 max_coefficient()

```
numeric GiNaC::basic::max_coefficient ( ) const  [virtual]
```

Implementation ex::max_coefficient().

See also

heur_gcd

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::mul, and GiNaC::add.

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::structure< T, ComparisonPolicy >, GiNaC::indexed, GiNaC::mul, GiNaC::add, GiNaC::ncmul, and GiNaC::integral.

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

self	First indexed expression; its base object is ∗this
other	Second indexed expression

Returns

sum of self and other

See also

ex::simplify_indexed()

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, and GiNaC::matrix.

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

self	Indexed expression; its base object is ∗this
other	Numeric value

Returns

product of self and other

See also

ex::simplify_indexed()

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, and GiNaC::matrix.

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

self	Pointer to first indexed expression; its base object is ∗this
other	Pointer to second indexed expression
v	The complete vector of factors

Returns

true if the contraction was successful, false otherwise

See also

ex::simplify_indexed()

Reimplemented in GiNaC::structure< T, ComparisonPolicy >, GiNaC::tensepsilon, GiNaC::spinmetric, GiNaC::su3d, GiNaC::diracgamma, GiNaC::su3f, GiNaC::cliffordunit, GiNaC::su3t, GiNaC::tensmetric, GiNaC::tensdelta, and GiNaC::matrix.

6.8.3.45 return_type()

```
unsigned GiNaC::basic::return_type ( ) const  [virtual]
```

Reimplemented in GiNaC::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::tensepsilon, GiNaC::indexed, GiNaC::su3d, GiNaC::minkmetric, GiNaC::su3f, GiNaC::tensmetric, GiNaC::power, GiNaC::expairseq, GiNaC::ncmul, GiNaC::mul, GiNaC::matrix, GiNaC::add, GiNaC::relational, GiNaC::tensdelta, GiNaC::clifford, GiNaC::color, GiNaC::integral, GiNaC::tensor, and GiNaC::fail.

6.8.3.46 return_type_tinfo()

```
return_type_t GiNaC::basic::return_type_tinfo ( ) const  [virtual]
```

Reimplemented in GiNaC::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::indexed, GiNaC::power, GiNaC::ncmul, GiNaC::mul, GiNaC::add, GiNaC::relational, GiNaC::clifford, GiNaC::color, and GiNaC::integral.

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::function, GiNaC::spinidx, GiNaC::diracgammaR, GiNaC::diracgammaL, GiNaC::container< C >, GiNaC::diracgamma5, GiNaC::numeric, GiNaC::realsymbol, GiNaC::expairseq, GiNaC::power, GiNaC::mul, GiNaC::ncmul, GiNaC::add, GiNaC::matrix, GiNaC::pseries, GiNaC::integral, GiNaC::symbol, and GiNaC::constant.

6.8.3.48 real_part()

```
ex GiNaC::basic::real_part ( ) const  [virtual]
```

Reimplemented in GiNaC::function, GiNaC::container< C >, GiNaC::indexed, GiNaC::numeric, GiNaC::realsymbol, GiNaC::power, GiNaC::ncmul, GiNaC::add, GiNaC::matrix, GiNaC::mul, GiNaC::pseries, GiNaC::symbol, and GiNaC::constant.

Referenced by GiNaC::ncmul::real_part(), and GiNaC::function::real_part().

6.8.3.49 imag_part()

```
ex GiNaC::basic::imag_part ( ) const  [virtual]
```

Reimplemented in GiNaC::function, GiNaC::container< C >, GiNaC::indexed, GiNaC::numeric, GiNaC::realsymbol, GiNaC::power, GiNaC::ncmul, GiNaC::add, GiNaC::matrix, GiNaC::mul, GiNaC::pseries, GiNaC::symbol, and GiNaC::constant.

Referenced by GiNaC::ncmul::imag_part(), and GiNaC::function::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::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::container< C >, GiNaC::numeric, GiNaC::expairseq, GiNaC::fderivative, GiNaC::symbol, and GiNaC::constant.

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::function, GiNaC::structure< T, ComparisonPolicy >, GiNaC::numeric, GiNaC::expairseq, GiNaC::relational, GiNaC::symmetry, GiNaC::symbol, GiNaC::constant, GiNaC::idx, and GiNaC::wildcard.

References GiNaC::ex::gethash(), GiNaC::make_hash_seed(), 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.

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::function, GiNaC::spinidx, GiNaC::tensepsilon, GiNaC::indexed, GiNaC::container< C >, GiNaC::varidx, GiNaC::numeric, GiNaC::minkmetric, GiNaC::power, GiNaC::expairseq, GiNaC::fderivative, GiNaC::symmetry, GiNaC::matrix, GiNaC::pseries, GiNaC::relational, GiNaC::symbol, GiNaC::integral, GiNaC::constant, GiNaC::idx, GiNaC::clifford, GiNaC::color, 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::function, GiNaC::spinidx, GiNaC::tensepsilon, GiNaC::indexed, GiNaC::varidx, GiNaC::container< C >, GiNaC::numeric, GiNaC::minkmetric, GiNaC::power, GiNaC::expairseq, GiNaC::symmetry, GiNaC::fderivative, GiNaC::matrix, GiNaC::pseries, GiNaC::relational, GiNaC::symbol, GiNaC::integral, GiNaC::constant, GiNaC::idx, GiNaC::clifford, GiNaC::color, 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 m, GiNaC::match(), options, and GiNaC::ex::subs().

Referenced by GiNaC::symbol::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::numeric::subs(), and 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

s	symbol to differentiate in
nth	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().

Referenced by GiNaC::pseries::coeff().

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::power::coeff(), GiNaC::ncmul::coeff(), GiNaC::power::degree(), GiNaC::ncmul::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.

See also

basic::eval

Referenced by GiNaC::abs_conjugate(), GiNaC::abs_eval(), GiNaC::abs_evalf(), GiNaC::abs_expand(), GiNaC::abs_power(), GiNaC::abs_real_part(), GiNaC::acos_conjugate(), GiNaC::acos_eval(), GiNaC::acos_evalf(), GiNaC::acosh_conjugate(), GiNaC::acosh_eval(), GiNaC::acosh_evalf(), GiNaC::asin_conjugate(), GiNaC::asin_eval(), GiNaC::asin_evalf(), GiNaC::asinh_conjugate(), GiNaC::asinh_eval(), GiNaC::asinh_evalf(), GiNaC::atan2_eval(), GiNaC::atan_conjugate(), GiNaC::atan_eval(), GiNaC::atan_evalf(), GiNaC::atanh_conjugate(), GiNaC::atanh_eval(), GiNaC::atanh_evalf(), GiNaC::binomial_conjugate(), GiNaC::binomial_eval(), GiNaC::binomial_evalf(), GiNaC::binomial_real_part(), GiNaC::binomial_sym(), GiNaC::conjugate_expl_derivative(), GiNaC::cos_eval(), GiNaC::cos_evalf(), GiNaC::cosh_eval(), GiNaC::cosh_evalf(), GiNaC::csgn_power(), GiNaC::epsilon_tensor(), GiNaC::eta_imag_part(), GiNaC::integral::eval(), GiNaC::pseries::eval(), GiNaC::add::eval(), GiNaC::power::eval(), GiNaC::fderivative::eval(), GiNaC::numeric::eval(), GiNaC::structure< T, ComparisonPolicy >::eval(), GiNaC::function::eval(), GiNaC::matrix::eval_indexed(), GiNaC::tensmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::su3d::eval_indexed(), GiNaC::structure< T, ComparisonPolicy >::eval_indexed(), GiNaC::spinmetric::eval_indexed(), GiNaC::tensepsilon::eval_indexed(), GiNaC::function::evalf(), GiNaC::exp_eval(), GiNaC::exp_evalf(), GiNaC::exp_expand(), GiNaC::exp_power(), GiNaC::power::expand(), GiNaC::factorial_conjugate(), GiNaC::factorial_eval(), GiNaC::factorial_evalf(), GiNaC::factorial_real_part(), GiNaC::G2_eval(), GiNaC::G2_evalf(), GiNaC::G3_eval(), GiNaC::G3_evalf(), GiNaC::imag_part_expl_derivative(), GiNaC::iterated_integral2_eval(), GiNaC::iterated_integral3_eval(), GiNaC::iterated_integral_evalf_impl(), GiNaC::Li2_conjugate(), GiNaC::Li2_eval(), GiNaC::Li2_evalf(), GiNaC::log_conjugate(), GiNaC::log_eval(), GiNaC::log_evalf(), GiNaC::log_expand(), GiNaC::log_real_part(), GiNaC::lorentz_eps(), GiNaC::psi1_eval(), GiNaC::psi1_evalf(), GiNaC::psi2_eval(), GiNaC::psi2_evalf(), GiNaC::real_part_expl_derivative(), GiNaC::sin_eval(), GiNaC::sin_evalf(), GiNaC::sinh_eval(), GiNaC::sinh_evalf(), GiNaC::step_conjugate(), GiNaC::step_eval(), GiNaC::step_evalf(), GiNaC::step_real_part(), GiNaC::tan_eval(), GiNaC::tan_evalf(), GiNaC::tanh_eval(), GiNaC::tanh_evalf(), GiNaC::zeta1_eval(), GiNaC::zeta1_evalf(), GiNaC::zeta2_eval(), GiNaC::zeta2_evalf(), and GiNaC::zetaderiv_eval().

6.8.3.62 gethash()

```
unsigned GiNaC::basic::gethash ( ) const  [inline]
```

References calchash(), flags, GiNaC::status_flags::hash_calculated, and hashvalue.

Referenced by compare(), and is_equal().

6.8.3.63 setflag()

```
const basic& GiNaC::basic::setflag (
            unsigned f ) const  [inline]
```

Set some status_flags.

References flags.

Referenced by GiNaC::wildcard::calchash(), GiNaC::constant::calchash(), GiNaC::idx::calchash(), GiNaC::symbol::calchash(), GiNaC::symmetry::calchash(), GiNaC::relational::calchash(), GiNaC::numeric::calchash(), GiNaC::function::calchash(), GiNaC::constant::constant(), GiNaC::container< C >::container(), GiNaC::realsymbol::duplicate(), GiNaC::possymbol::duplicate(), duplicate(), GiNaC::power::eval(), GiNaC::ncmul::eval(), GiNaC::pseries::evalf(), GiNaC::integral::expand(), GiNaC::pseries::expand(), GiNaC::ncmul::expand(), GiNaC::add::expand(), GiNaC::power::expand(), GiNaC::function::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::power::expand_mul(), GiNaC::GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(), GiNaC::power::info(), GiNaC::matrix::matrix(), GiNaC::numeric::numeric(), GiNaC::print_func< print_dflt >(), GiNaC::wildcard::read_archive(), GiNaC::symbol::read_archive(), GiNaC::container< C >::read_archive(), GiNaC::numeric::read_archive(), GiNaC::reduced_matrix(), GiNaC::sub_matrix(), GiNaC::symbol::symbol(), GiNaC::symbolic_matrix(), GiNaC::symmetry::symmetry(), GiNaC::unit_matrix(), and GiNaC::wildcard::wildcard().

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::power::info(), GiNaC::idx::map(), map(), GiNaC::idx::replace_dim(), GiNaC::mul::smod(), GiNaC::add::split_ex_to_pair(), GiNaC::idx::subs(), 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::integral::let_op(), GiNaC::matrix::let_op(), GiNaC::clifford::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::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 Function Documentation

6.8.4.1 ex

```
friend class ex  [friend]
```

Referenced by GiNaC::matrix::charpoly(), GiNaC::indexed::eval(), GiNaC::function::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::idx::calchash(), GiNaC::relational::calchash(), GiNaC::symmetry::calchash(), GiNaC::function::calchash(), clearflag(), GiNaC::ex::construct_from_basic(), GiNaC::wildcard::do_print_tree(), GiNaC::constant::do_print_tree(), GiNaC::clifford::do_print_tree(), GiNaC::fderivative::do_print_tree(), GiNaC::symbol::do_print_tree(), GiNaC::symmetry::do_print_tree(), GiNaC::idx::do_print_tree(), GiNaC::pseries::do_print_tree(), GiNaC::varidx::do_print_tree(), GiNaC::numeric::do_print_tree(), GiNaC::indexed::do_print_tree(), GiNaC::spinidx::do_print_tree(), GiNaC::integral::eval(), GiNaC::add::eval(), GiNaC::power::eval(), GiNaC::ncmul::eval(), GiNaC::function::eval(), GiNaC::integral::eval_integ(), GiNaC::integral::expand(), gethash(), 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::wildcard::calchash(), GiNaC::constant::calchash(), GiNaC::idx::calchash(), GiNaC::symbol::calchash(), GiNaC::relational::calchash(), GiNaC::symmetry::calchash(), GiNaC::numeric::calchash(), GiNaC::function::calchash(), GiNaC::wildcard::do_print_tree(), GiNaC::constant::do_print_tree(), GiNaC::clifford::do_print_tree(), GiNaC::fderivative::do_print_tree(), GiNaC::symbol::do_print_tree(), GiNaC::symmetry::do_print_tree(), GiNaC::idx::do_print_tree(), GiNaC::pseries::do_print_tree(), GiNaC::varidx::do_print_tree(), GiNaC::numeric::do_print_tree(), GiNaC::indexed::do_print_tree(), GiNaC::spinidx::do_print_tree(), gethash(), operator=(), and GiNaC::function::print().

The documentation for this class was generated from the following files:

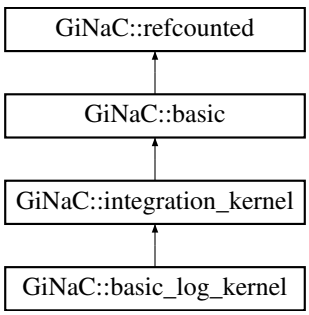
- basic.h
- basic.cpp
- normal.cpp
- pseries.cpp

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:



Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Additional Inherited Members

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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{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:

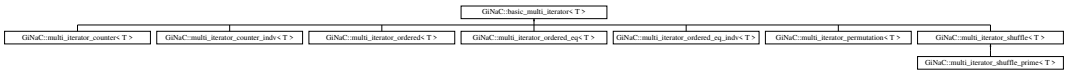
- integration_kernel.h
- integration_kernel.cpp

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
  $$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$
  .
- 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.

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

$$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$

.

Reimplemented in GiNaC::multi_iterator_shuffle_prime< T >, GiNaC::multi_iterator_shuffle< T >, GiNaC::multi_iterator_permutatio
GiNaC::multi_iterator_counter_indv< T >, GiNaC::multi_iterator_counter< T >, GiNaC::multi_iterator_ordered_eq_indv< T >,
GiNaC::multi_iterator_ordered_eq< T >, and GiNaC::multi_iterator_ordered< 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_shuffle< T >, GiNaC::multi_iterator_permutation< T >, GiNaC::multi_iterator_counter_indv
GiNaC::multi_iterator_counter< T >, GiNaC::multi_iterator_ordered_eq_indv< T >, GiNaC::multi_iterator_ordered_eq< T >,
and GiNaC::multi_iterator_ordered< T >.

6.10.4 Friends And Related Function 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().

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:

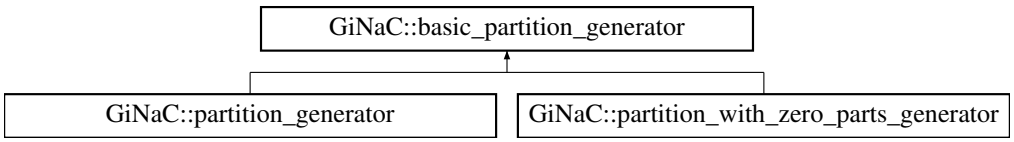
- utils_multi_iterator.h

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:



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 mpgen

```
mpartition2 GiNaC::basic_partition_generator::mpgen  [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>
```

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.

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 >::find(), GiNaC::function::print(), and GiNaC::basic::print_dispatch().

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 >::identify_parents().

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:

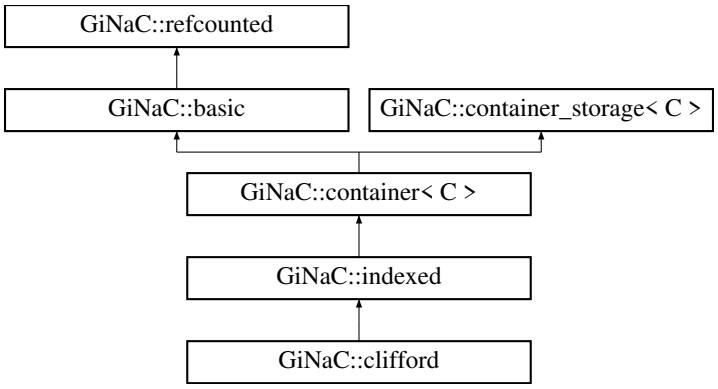
- class_info.h

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

- clifford (const ex &b, unsigned char rl=0)

  Construct object without any indices.
- clifford (const ex &b, const ex &mu, const ex &metr, unsigned char rl=0, int comm_sign=-1)

  Construct object with one Lorentz index.
- clifford (unsigned char rl, const ex &metr, int comm_sign, const exvector &v)
- clifford (unsigned char rl, const ex &metr, int comm_sign, exvector &&v)
- unsigned precedence () const override

  Return relative operator precedence (for parenthezing output).
- 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
- ex get_metric () const
- virtual ex get_metric (const ex &i, const ex &j, bool symmetrised=false) const
- bool same_metric (const ex &other) const

- int get_commutator_sign () const
- 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.

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_dflt (const print_dflt &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 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~i e~j +/- e~j e~i = B(i, j) + B(j, i)

Additional Inherited Members

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 commutate).

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(), GiNaC::symmetric2(), and GiNaC::transpose().

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::is_zero(), op(), GiNaC::ex::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.

References GiNaC::nops().

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(), GiNaC::op(), 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, metric, options, GiNaC::ex::subs(), and GiNaC::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(), read_archive(), and subs().

6.13.4.3 commutator_sign

```
int GiNaC::clifford::commutator_sign  [protected]
```

It is the sign in the definition e∼i e∼j +/- 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:

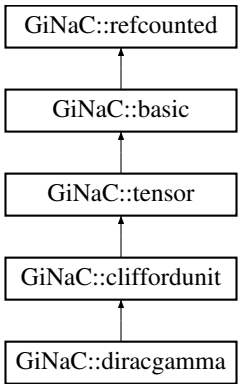
- clifford.h
- clifford.cpp

6.14 GiNaC::cliffordunit Class Reference

This class represents the Clifford algebra generators (units).

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::cliffordunit:



Public Member Functions

- bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const override
  Contraction of a Clifford unit with something else.

Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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:

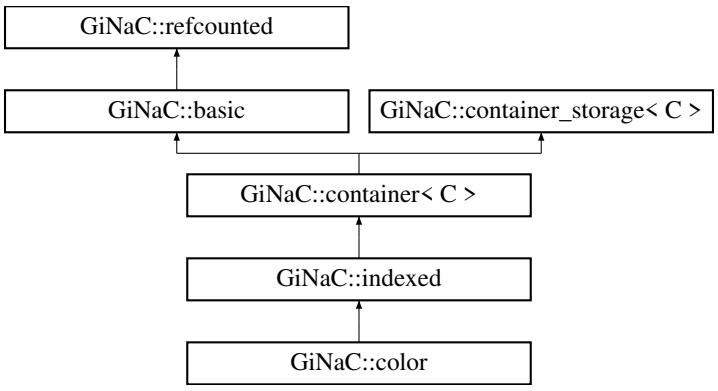
- clifford.h
- clifford.cpp

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:



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

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

Private Attributes

- unsigned char representation_label

  Representation label to distinguish independent color matrices coming from separated fermion lines.

Additional Inherited Members

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 commutate). 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().

6.15.2.2 color() [2/4]

```
GiNaC::color::color (
             const ex & b,
             const ex & i1,
             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(), and GiNaC::container_storage< C >::reserve().

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(), 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>
```

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::next, 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 atend, cmgen, current_updated, GiNaC::composition_generator::coolmulti::finished(), GiNaC↩ ::composition_generator::coolmulti::next_permutation(), 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 atend

```
bool GiNaC::composition_generator::atend  [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().

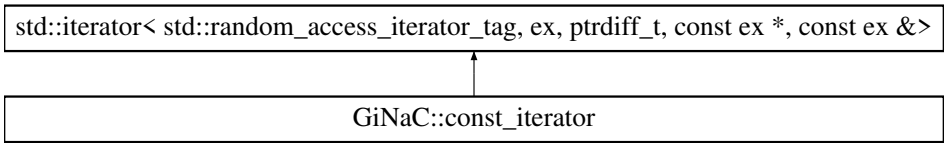
The documentation for this class was generated from the following file:

- utils.h

6.20 GiNaC::const_iterator Class Reference

`#include <ex.h>`

Inheritance diagram for GiNaC::const_iterator:



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 Constructor & Destructor Documentation

6.20.1.1 const_iterator() [1/2]

```
GiNaC::const_iterator::const_iterator ( )  [inline], [noexcept]
```

Referenced by operator+(), and operator-().

6.20.1.2 const_iterator() [2/2]

```
GiNaC::const_iterator::const_iterator (
             const ex & e_,
             size_t i_ )  [inline], [private], [noexcept]
```

6.20.2 Member Function Documentation

6.20.2.1 operator*()

```
ex GiNaC::const_iterator::operator* ( ) const  [inline]
```

References e, i, and GiNaC::ex::op().

6.20.2.2 operator->()

```
std::unique_ptr<ex> GiNaC::const_iterator::operator-> ( ) const  [inline]
```

References ex.

6.20.2.3 operator[]()

```
ex GiNaC::const_iterator::operator[] (
             difference_type n ) const  [inline]
```

References e, i, n, and GiNaC::ex::op().

6.20.2.4 operator++() [1/2]

```
const_iterator& GiNaC::const_iterator::operator++ ( )  [inline], [noexcept]
```

References i.

6.20.2.5 operator++() [2/2]

```
const_iterator GiNaC::const_iterator::operator++ (
            int  )  [inline], [noexcept]
```

References i.

6.20.2.6 operator+=()

```
const_iterator& GiNaC::const_iterator::operator+= (
            difference_type n )  [inline], [noexcept]
```

References i, and n.

6.20.2.7 operator+()

```
const_iterator GiNaC::const_iterator::operator+ (
            difference_type n ) const  [inline], [noexcept]
```

References const_iterator(), e, i, and n.

6.20.2.8 operator--() [1/2]

```
const_iterator& GiNaC::const_iterator::operator-- ( )  [inline], [noexcept]
```

References i.

6.20.2.9 operator--() [2/2]

```
const_iterator GiNaC::const_iterator::operator-- (
            int  )  [inline], [noexcept]
```

References i.

6.20.2.10 operator-=()

```
const_iterator& GiNaC::const_iterator::operator-= (
            difference_type n )  [inline], [noexcept]
```

References i, and n.

6.20.2.11 operator-()

```
const_iterator GiNaC::const_iterator::operator- (
            difference_type n ) const  [inline], [noexcept]
```

References const_iterator(), e, i, and n.

6.20.2.12 operator==()

```
bool GiNaC::const_iterator::operator== (
            const const_iterator & other ) const  [inline], [noexcept]
```

References GiNaC::are_ex_trivially_equal(), e, and i.

6.20.2.13 operator"!=()

```
bool GiNaC::const_iterator::operator!= (
            const const_iterator & other ) const  [inline], [noexcept]
```

6.20.2.14 operator<()

```
bool GiNaC::const_iterator::operator< (
            const const_iterator & other ) const  [inline], [noexcept]
```

References i.

6.20.2.15 operator>()

```
bool GiNaC::const_iterator::operator> (
            const const_iterator & other ) const  [inline], [noexcept]
```

6.20.2.16 operator<=()

```
bool GiNaC::const_iterator::operator<= (
            const const_iterator & other ) const  [inline], [noexcept]
```

6.20.2.17 operator>=()

```
bool GiNaC::const_iterator::operator>= (
            const const_iterator & other ) const  [inline], [noexcept]
```

6.20.3 Friends And Related Function Documentation

6.20.3.1 ex

```
friend class ex  [friend]
```

Referenced by operator->().

6.20.3.2 const_preorder_iterator

```
friend class const_preorder_iterator  [friend]
```

6.20.3.3 const_postorder_iterator

```
friend class const_postorder_iterator  [friend]
```

6.20.3.4 operator+

```
const_iterator operator+ (
            difference_type n,
            const const_iterator & it )  [friend]
```

6.20.3.5 operator-

```
difference_type operator- (
            const const_iterator & lhs,
            const const_iterator & rhs )  [friend]
```

6.20.4 Member Data Documentation

6.20.4.1 e

```
ex GiNaC::const_iterator::e  [protected]
```

Referenced by operator∗(), operator+(), operator-(), operator==(), and operator[ ]().

6.20.4.2 i

```
size_t GiNaC::const_iterator::i  [protected]
```

Referenced by operator∗(), operator+(), operator++(), operator+=(), operator-(), operator--(), operator-=(), operator<(), operator==(), and operator[ ]().

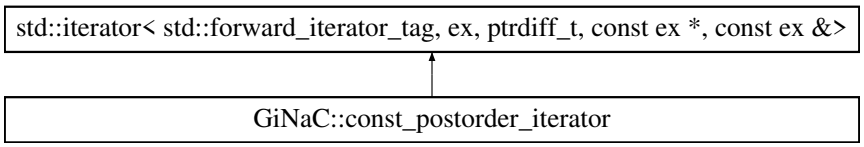
The documentation for this class was generated from the following file:

- ex.h

6.21 GiNaC::const_postorder_iterator Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::const_postorder_iterator:



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 Constructor & Destructor Documentation

6.21.1.1 const_postorder_iterator() [1/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator ( )  [inline], [noexcept]
```

6.21.1.2 const_postorder_iterator() [2/2]

```
GiNaC::const_postorder_iterator::const_postorder_iterator (
            const ex & e,
            size_t n )  [inline]
```

References descend(), n, and s.

6.21.2 Member Function Documentation

6.21.2.1 operator*()

```
reference GiNaC::const_postorder_iterator::operator* ( ) const  [inline]
```

References s.

6.21.2.2 operator->()

```
pointer GiNaC::const_postorder_iterator::operator-> ( ) const  [inline]
```

References s.

6.21.2.3 operator++() [1/2]

```
const_postorder_iterator& GiNaC::const_postorder_iterator::operator++ ( )  [inline]
```

References increment().

6.21.2.4 operator++() [2/2]

```
const_postorder_iterator GiNaC::const_postorder_iterator::operator++ (
            int  )  [inline]
```

References increment().

6.21.2.5 operator==()

```
bool GiNaC::const_postorder_iterator::operator== (
            const const_postorder_iterator & other ) const  [inline], [noexcept]
```

References s.

6.21.2.6 operator"!=()

```
bool GiNaC::const_postorder_iterator::operator!= (
            const const_postorder_iterator & other ) const  [inline], [noexcept]
```

6.21.2.7 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().

6.21.2.8 increment()

```
void GiNaC::const_postorder_iterator::increment ( )  [inline], [private]
```

References descend(), and s.

Referenced by operator++().

6.21.3 Member Data Documentation

6.21.3.1 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==().

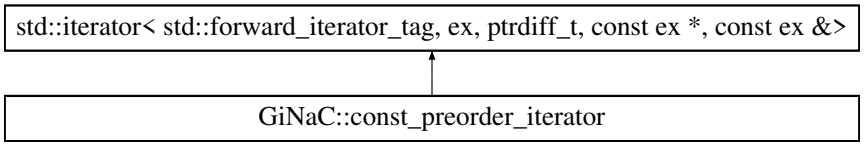
The documentation for this class was generated from the following file:

- ex.h

6.22 GiNaC::const_preorder_iterator Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::const_preorder_iterator:



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 Constructor & Destructor Documentation

6.22.1.1 const_preorder_iterator() [1/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator ( )  [inline], [noexcept]
```

6.22.1.2 const_preorder_iterator() [2/2]

```
GiNaC::const_preorder_iterator::const_preorder_iterator (
            const ex & e,
            size_t n )  [inline]
```

References n, and s.

6.22.2 Member Function Documentation

6.22.2.1 operator∗()

```
reference GiNaC::const_preorder_iterator::operator* ( ) const  [inline]
```

References s.

6.22.2.2 operator->()

```
pointer GiNaC::const_preorder_iterator::operator-> ( ) const  [inline]
```

References s.

6.22.2.3 operator++() [1/2]

```
const_preorder_iterator& GiNaC::const_preorder_iterator::operator++ ( )  [inline]
```

References increment().

6.22.2.4 operator++() [2/2]

```
const_preorder_iterator GiNaC::const_preorder_iterator::operator++ (
            int  )  [inline]
```

References increment().

6.22.2.5 operator==()

```
bool GiNaC::const_preorder_iterator::operator== (
            const const_preorder_iterator & other ) const  [inline], [noexcept]
```

References s.

6.22.2.6 operator"!=()

```
bool GiNaC::const_preorder_iterator::operator!= (
            const const_preorder_iterator & other ) const  [inline], [noexcept]
```

6.22.2.7 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++().

6.22.3 Member Data Documentation

6.22.3.1 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:

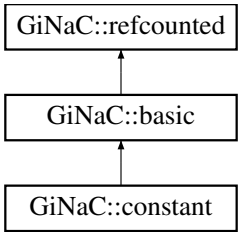
- ex.h

6.23 GiNaC::constant Class Reference

This class holds constants, symbols with specific numerical value.

`#include <constant.h>`

Inheritance diagram for GiNaC::constant:



Public Member Functions

- constant (const std::string &initname, evalffunctype 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.

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

Private Attributes

- std::string name

  printname of this constant
- std::string TeX_name

  LaTeX name.
- evalffunctype 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

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,
            evalffunctype 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(), 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(), 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:

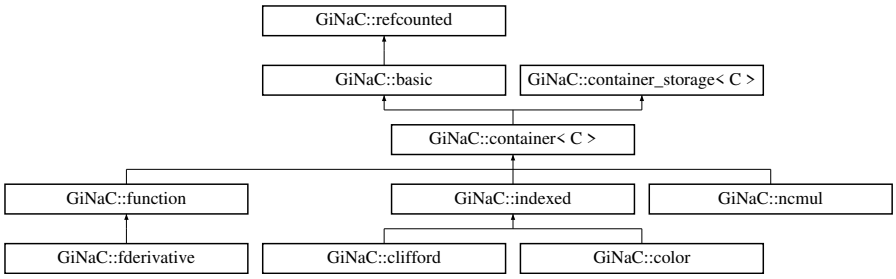
- constant.h
- constant.cpp

6.24 GiNaC::container< C > Class Template Reference

Wrapper template for making GiNaC classes out of STL containers.

```
#include <registrar.h>
```

Inheritance 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

Protected Types

- typedef container_storage< C >::STLT 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 subschildren (const exmap &m, unsigned options=0) const

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.

Private Member Functions

- void sort_ (std::random_access_iterator_tag)
- void sort_ (std::input_iterator_tag)
- void unique_ ()
- template<>
  void unique_ ()

  Specialization of container::unique_() for std::list.

Additional Inherited Members

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().

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(), 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()

```
bool GiNaC::lst::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::lst_to_matrix(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC::power::normal(), GiNaC::ex::normal(), GiNaC::basic::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::ncmul::expand(), GiNaC::mul::expand(), GiNaC::ncmul::get_free_indices(), GiNaC::H_evalf(), GiNaC::indexed::imag_part(), GiNaC::is_discriminant_of_quadratic_number_field(), GiNaC::kronecker_symbol(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC::power::normal(), GiNaC::ex::normal(), GiNaC::basic::normal(), GiNaC::ex::numer(), GiNaC::ex::numer_denom(), GiNaC::indexed::real_part(), 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.

Referenced by GiNaC::basic::normal().

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::function, GiNaC::indexed, and GiNaC::fderivative.

References GiNaC::container< C >::get_default_flags(), n, GiNaC::container_storage< C >::reserve(), GiNaC::container_storage< C >::seq, 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::function, GiNaC::indexed, and GiNaC::fderivative.

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::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(), 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(), 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::function, and GiNaC::fderivative.

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 STLT & 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 (
            STLT && 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::fderivative::do_print_csrc(), 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::ifactor(), GiNaC::Li_eval(), GiNaC::lsolve(), GiNaC::replace_with_symbol(), 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::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::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]

```
template<>
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:

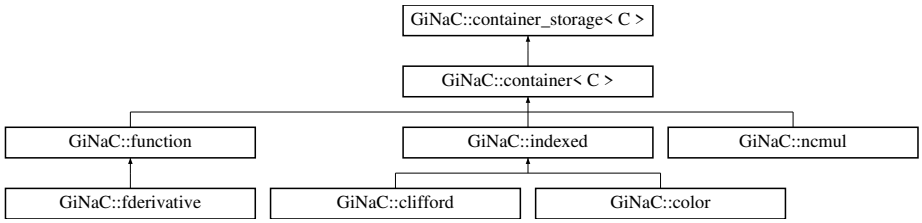
- container.h
- exprseq.h
- lst.h

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 ()
- template<>
  void reserve (size_t n)
- template<>
  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::color::eval_ncmul(), 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 (
            STLT & ,
            size_t  )  [inline], [static], [protected]
```

6.25.4.3 reserve() [3/4]

```
template<>
void GiNaC::container_storage< std::vector >::reserve (
            size_t n )  [inline], [protected]
```

References n.

6.25.4.4 reserve() [4/4]

```
template<>
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::ncmul::degree(), GiNaC::fderivative::derivative(), GiNaC::ncmul::derivative(), GiNaC::function::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::ncmul::eval(), GiNaC::indexed::eval(), GiNaC::function::eval(), GiNaC::function::eval_ncmul(), GiNaC::function::evalf(), GiNaC::ncmul::evalm(), GiNaC::ncmul::expand(), GiNaC::indexed::expand(), GiNaC::function::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::info(), GiNaC::function::info(), 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::indexed::read_archive(), GiNaC::function::read_archive(), GiNaC::function::real_part(), GiNaC::container< C >::rend(), GiNaC::ncmul::return_type(), GiNaC::function::return_type(), GiNaC::ncmul::return_type_tinfo(), GiNaC::function::return_type_tinfo(), GiNaC::function::series(), GiNaC::simplify_indexed(), 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

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::value.

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::value.

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().

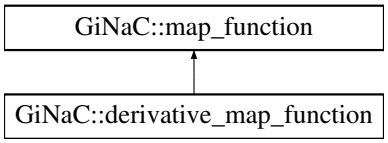
The documentation for this struct was generated from the following file:

- 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:



Public Member Functions

- derivative_map_function (const symbol &sym)
- ex operator() (const ex &e) override

Public Attributes

- const symbol & s

Additional Inherited Members

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	Gauss elimination. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$$ The determinant is then just the product of diagonal elements. Choose this algorithm only for purely numerical matrices.
divfree	Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$$ 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.
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	Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is *delayed* until it can be carried out without computing GCDs. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}) / m_{k-1,k-1}^{(k-1)}$$ (We have set $m_{-1,-1}^{(-1)} = 1$ 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 of from the lower right entry. This algorithm is rarely fast for computing determinants.

The documentation for this class was generated from the following file:

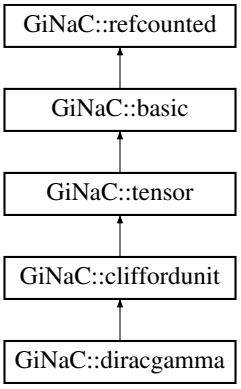
- flags.h

6.29 GiNaC::diracgamma Class Reference

This class represents the Dirac gamma Lorentz vector.

`#include <clifford.h>`

Inheritance diagram for GiNaC::diracgamma:



Public Member Functions

- bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const override

  Contraction of a gamma matrix with something else.

Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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:

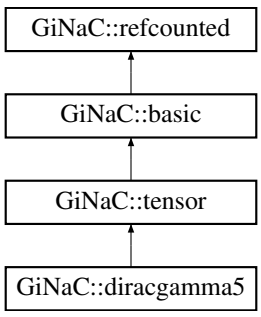
- clifford.h
- clifford.cpp

6.30 GiNaC::diracgamma5 Class Reference

This class represents the Dirac gamma5 object which anticommutates with all other gammas.

```
#include <clifford.h>
```

Inheritance 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

Private Member Functions

- ex conjugate () const override

Additional Inherited Members

6.30.1 Detailed Description

This class represents the Dirac gamma5 object which anticommutates 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:

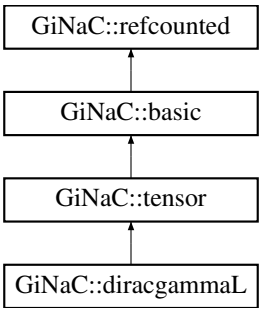
- clifford.h
- clifford.cpp

6.31 GiNaC::diracgammaL Class Reference

This class represents the Dirac gammaL object which behaves like 1/2 (1-gamma5).

`#include <clifford.h>`

Inheritance 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

Private Member Functions

- ex conjugate () const override

Additional Inherited Members

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:

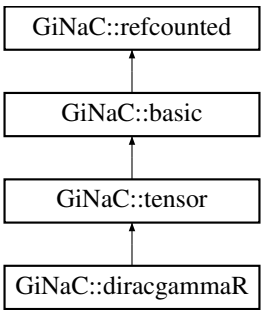
- clifford.h
- clifford.cpp

6.32 GiNaC::diracgammaR Class Reference

This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).

```
#include <clifford.h>
```

Inheritance 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

Private Member Functions

- ex conjugate () const override

Additional Inherited Members

6.32.1 Detailed Description

This class represents the Dirac gammaL object which behaves like 1/2 (1+gamma5).

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:

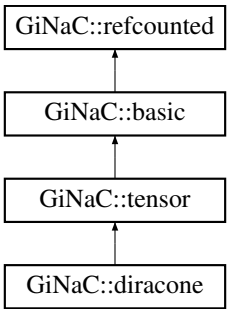
- clifford.h
- clifford.cpp

6.33 GiNaC::diracone Class Reference

This class represents the Clifford algebra unity element.

```
#include <clifford.h>
```

Inheritance diagram for GiNaC::diracone:



Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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:

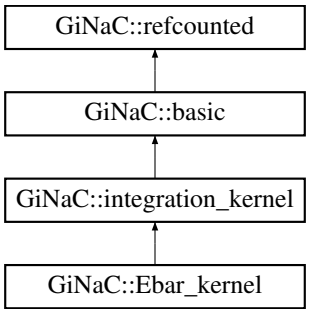
- utils.h

6.37 GiNaC::Ebar_kernel Class Reference

The Ebar-kernel.

`#include <integration_kernel.h>`

Inheritance diagram for GiNaC::Ebar_kernel:



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)

Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Protected Attributes

- ex n
- ex m
- ex x
- ex y

6.37.1 Detailed Description

The Ebar-kernel.

This class represents the differential one-form

$$\omega^{\overline{\mathrm{E}}}_{n;m}(x;y) = \overline{\mathrm{E}}_{n;m}(x;y;\bar{q})\frac{d\bar{q}}{\bar{q}}$$

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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{i-1}.

Reimplemented from GiNaC::integration_kernel.

References GiNaC::ex::evalf(), k, m, n, GiNaC::to_int(), 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:

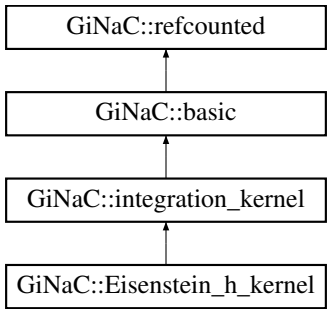
- integration_kernel.h
- integration_kernel.cpp

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:



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

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 Attributes

- ex k
- ex N
- ex r
- ex s
- ex C_norm

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::integration_kernel.

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, and s.

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:

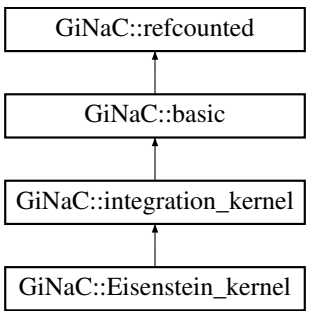
- integration_kernel.h
- integration_kernel.cpp

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:



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

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 Attributes

- ex k
- ex N
- ex a
- ex b
- ex K
- ex C_norm

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

$$\omega^{\text{Eisenstein}}_{k,N,a,b,K} = C_k E_{k,N,a,b,K}(\tau)\frac{d\bar{q}_N}{\bar{q}_N}$$

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::integration_kernel.

References 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>`

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:

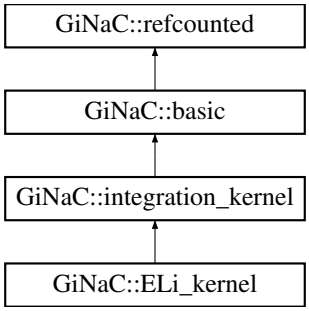
- utils.h

6.41 GiNaC::ELi_kernel Class Reference

The ELi-kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::ELi_kernel:



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)

Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Protected Attributes

- ex n
- ex m
- ex x
- ex y

6.41.1 Detailed Description

The ELi-kernel.

This class represents the differential one-form

$$\omega_{n;m}^{\mathrm{ELi}}(x;y) = \mathrm{ELi}_{n;m}(x;y;\bar{q})\frac{d\bar{q}}{\bar{q}}$$

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 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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{i-1}.

Reimplemented from GiNaC::integration_kernel.

References GiNaC::ex::evalf(), k, m, n, GiNaC::to_int(), 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 Template 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

template<>
struct std::equal_to< GiNaC::ex >

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

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.

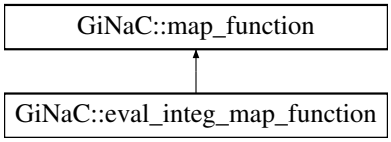
The documentation for this struct was generated from the following file:

- 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:



Public Member Functions

- ex operator() (const ex &e) override

Additional Inherited Members

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().

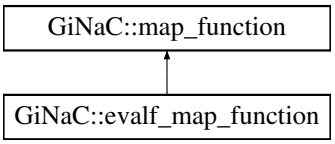
The documentation for this struct was generated from the following file:

- 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:



Public Member Functions

- ex operator() (const ex &e) override

Additional Inherited Members

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().

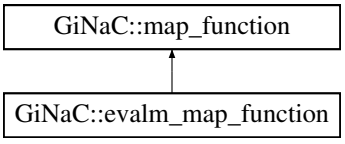
The documentation for this struct was generated from the following file:

- 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:



Public Member Functions

- ex operator() (const ex &e) override

Additional Inherited Members

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>
```

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 arount print to be called within a debugger.
- void dbgprinttree () const

  Little wrapper arount 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
- 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 makewriteable ()

  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 lst in the second argument. Undefined symbols and other parser errors will throw an exception.

References bp, GiNaC::status_flags::dynallocated, 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::dynallocated, and GINAC_ASSERT.

Referenced by GiNaC::ncmul::derivative(), GiNaC::ex_swap::operator()(), GiNaC::expair::swap(), GiNaC::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::canonicalize(), GiNaC::mul::derivative(), GiNaC::divide_in_z(), GiNaC::ncmul::evalm(), GiNaC::exp_expand(), GiNaC::mul::expandchildren(), GiNaC::G3↩
_eval(), GiNaC::G3_evalf(), GiNaC::gcd(), GiNaC::Li_eval(), GiNaC::Li_evalf(), GiNaC::Li_print_latex(), GiNa↩
C::log_expand(), GiNaC::rename_dummy_indices(), GiNaC::mul::series(), GiNaC::symmetrize(), and GiNaC↩
::symmetrize_cyclic().

6.48.3.3 end()

```
const_iterator GiNaC::ex::end ( ) const  [inline], [noexcept]
```

References nops().

Referenced by GiNaC::abs_expand(), GiNaC::canonicalize(), GiNaC::exp_expand(), GiNaC::G3_eval(), GiNaC::↩ G3_evalf(), GiNaC::Li_print_latex(), and GiNaC::log_expand().

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.

Referenced by GiNaC::eval().

6.48.3.9 evalf()

```
ex GiNaC::ex::evalf ( ) const  [inline]
```

References bp.

Referenced by GiNaC::adaptivesimpson(), GiNaC::EllipticE_evalf(), GiNaC::EllipticK_evalf(), GiNaC::integral::evalf(), GiNaC::constant::evalf(), GiNaC::power::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.

Referenced by GiNaC::add::evalm(), GiNaC::pseries::evalm(), GiNaC::power::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.

Referenced by GiNaC::integral::eval_ncmul(), GiNaC::relational::eval_ncmul(), and GiNaC::mul::eval_ncmul().

6.48.3.12 eval_integ()

```
ex GiNaC::ex::eval_integ ( ) const  [inline]
```

References bp.

Referenced by 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::relational::do_print(), GiNaC::mul::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::add::do_print_csrc(), GiNaC::power::do_print_csrc(), GiNaC::idx::do_print_csrc(), GiNaC::mul::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::relational::do_print_python_repr(), GiNaC::add::do_print_python_repr(), GiNaC::power::do_print_python_repr(), GiNaC::mul::do_print_python_repr(), GiNaC::pseries::do_print_python_repr(), GiNaC::clifford::do_print_tree(), GiNaC::idx::do_print_tree(), GiNaC::pseries::do_print_tree(), GiNaC::varidx::do_print_tree(), GiNaC::spinidx::do_print_tree(), GiNaC::basic::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::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 arount print to be called within a debugger.

References bp.

6.48.3.15 dbgprinttree()

```
void GiNaC::ex::dbgprinttree ( ) const
```

Little wrapper arount printtree to be called within a debugger.

References bp.

6.48.3.16 info()

```
bool GiNaC::ex::info (
            unsigned inf ) 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::asinh_conjugate(), GiNaC::asinh_eval(), GiNaC::atan2_eval(), GiNaC::atan_conjugate(), GiNaC::atan_eval(), GiNaC::atanh_eval(), GiNaC::beta_eval(), GiNaC::beta_series(), GiNaC::mul::can_be_further_expanded(), GiNaC::mul::can_make_flat(), GiNaC::pseries::conjugate(), GiNaC::power::conjugate(), 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::power::do_print_csrc(), GiNaC::idx::do_print_csrc(), GiNaC::eta_eval(), GiNaC::eta_evalf(), GiNaC::eta_series(), GiNaC::power::eval(), GiNaC::exp_eval(), GiNaC::exp_power(), GiNaC::power::expand(), 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::pseries::imag_part(), GiNaC::power::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_real_part(), GiNaC::log_series(), GiNaC::lsolve(), GiNaC::power::normal(), GiNaC::Order_imag_part(), GiNaC::matrix::pow(), GiNaC::prem(), GiNaC::psi1_eval(), GiNaC::psi1_series(), GiNaC::psi2_eval(), GiNaC::psi2_series(), GiNaC::quo(), GiNaC::pseries::real_part(), GiNaC::power::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::tryfactsubs(), 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::integral::expand(), GiNaC::mul::expand(), GiNaC::indexed::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::basic::match(), GiNaC::multiply_lcm(), GiNaC::nops(), postorder_begin(), preorder_begin(), GiNaC::product_to_exvector(), GiNaC::rename_dummy_indices_uniquely(), GiNaC::reposition_dummy_indices(), GiNaC::integral::series(), GiNaC::simplify_indexed(), 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.

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::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::power::eval(), GiNaC::indexed::eval(), GiNaC::integral::eval_integ(), GiNaC::exp_eval(), GiNaC::integral::expand(), GiNaC::mul::expand(), GiNaC::indexed::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::basic::match(), GiNaC::multiply_lcm(), GiNaC::power::normal(), normal(), GiNaC::basic::normal(), numer(), GiNaC::op(), GiNaC::op0_is_equal::operator()(), GiNaC::ex_base_is_less::operator()(), GiNaC::const_iterator::operator*(), GiNaC::const_iterator::operator[](), GiNaC::product_to_exvector(), GiNaC::replace_with_symbol(), GiNaC::reposition_dummy_indices(), GiNaC::clifford::same_metric(), GiNaC::matrix::scalar_mul_indexed(), GiNaC::simplify_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().

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(), and GiNaC::Eisenstein_h_kernel::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::rhs(), and GiNaC::Eisenstein_h_kernel::series().

6.48.3.26 conjugate()

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

References bp.

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(), GiNaC::integral::conjugate(), GiNaC::pseries::conjugate(), GiNaC::add::conjugate(), GiNaC::power::conjugate(), GiNaC::expair::conjugate(), GiNaC::conjugate(), GiNaC::conjugate_eval(), 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.

Referenced by GiNaC::abs_eval(), GiNaC::asinh_conjugate(), GiNaC::atan_conjugate(), GiNaC::conjugate_real_part(), GiNaC::mul::find_real_imag(), GiNaC::pseries::imag_part(), GiNaC::power::imag_part(), GiNaC::pseries::real_part(), GiNaC::add::real_part(), GiNaC::power::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.

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(), 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(), find(), GiNaC::find(), GiNaC::pseries::mul_series(), and GiNaC::replace_with_symbol().

6.48.3.31 match() [1/2]

```
bool GiNaC::ex::match (
            const ex & pattern ) const
```

Check whether expression matches a specified pattern.

References bp.

Referenced by find(), GiNaC::power::has(), 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, and options.

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::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::user_defined_kernel::is_numeric(), GiNaC::user_defined_kernel::Laurent_series(), GiNaC::lgamma_series(), GiNaC::Li2_series(), GiNaC::Li_series(), GiNaC::log_series(), GiNaC::power::normal(), normal(), numer(), numer_denom(), GiNaC::psi1_series(), GiNaC::psi2_series(), GiNaC::rename_dummy_indices(), GiNaC::rename_dummy_indices_uniquely(), GiNaC::tensor::replace_contr_index(), GiNaC::replace_with_symbol(), GiNaC::reposition_dummy_indices(), GiNaC::S_series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::basic::series(), GiNaC::step_series(), GiNaC::idx::subs(), GiNaC::pseries::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::basic::subs(), GiNaC::subs(), GiNaC::basic::subs_one_level(), GiNaC::container< C >::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, and GiNaC::info_flags::relation_equal.

6.48.3.36 map() [1/2]

```
ex GiNaC::ex::map (
            map_function & f ) const  [inline]
```

References bp.

Referenced by GiNaC::color_trace(), and GiNaC::expand_dummy_sum().

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::power::is_polynomial(), and GiNaC::is_polynomial().

6.48.3.43 degree()

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

References bp.

Referenced by GiNaC::basic::collect(), 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(), GiNaC::sqrfree_parfrac(), and GiNaC::sr_gcd().

6.48.3.44 ldegree()

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

References bp.

Referenced by GiNaC::gcd_pf_pow(), GiNaC::get_symbol_stats(), 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, and n.

Referenced by GiNaC::Bernoulli_polynomial(), GiNaC::pseries::coeff(), GiNaC::add::coeff(), GiNaC::mul::coeff(), GiNaC::coeff(), GiNaC::basic::collect(), GiNaC::pseries::convert_to_poly(), GiNaC::divide(), GiNaC::divide_in_z(), GiNaC::pseries::eval_integ(), GiNaC::pseries::evalm(), GiNaC::pseries::expand(), GiNaC::power::expand_add_2(), GiNaC::generalised_Bernoulli_number(), GiNaC::add::imag_part(), lcoeff(), GiNaC::lsolve(), 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::simplify_indexed(), GiNaC::sprem(), GiNaC::sqrfree_parfrac(), 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(), 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
```

References bp, 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::integral::expand(), GiNaC::pseries::expand(), GiNaC::power::expand(), GiNaC::mul::expand(), GiNaC::indexed::expand(), GiNaC::expand(), GiNaC::expand_dummy_sum(), GiNaC::ncmul::expandchildren(), GiNaC::mul::expandchildren(), GiNaC::mul::find_real_imag(), GiNaC::frac_cancel(), 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::mul::series(), GiNaC::power::series(), GiNaC::basic::series(), GiNaC::simplify_indexed(), GiNaC::sprem(), GiNaC::sqrfree_parfrac(), 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.

Referenced by GiNaC::matrix::charpoly(), 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

s	symbol by which the expression is derived
nth	order of derivative (default 1)

Returns

partial derivative as a new expression

References bp.

Referenced by GiNaC::abs_expl_derivative(), GiNaC::conjugate_expl_derivative(), GiNaC::integral::derivative(), GiNaC::power::derivative(), GiNaC::basic::diff(), GiNaC::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

r	expansion relation, lhs holds variable and rhs holds point
order	truncation order of series calculations
options	of class series_options

Returns

an expression holding a pseries object

References GiNaC::_ex0, bp, options, order, and r.

Referenced by GiNaC::Bernoulli_polynomial(), GiNaC::EllipticE_series(), GiNaC::EllipticK_series(), GiNaC::generalised_Bernoulli_number(), GiNaC::modular_form_kernel::is_numeric(), GiNaC::integration_kernel::Laurent_series(), GiNaC::Eisenstein_kernel::Laurent_series(), GiNaC::Eisenstein_h_kernel::Laurent_series(), GiNaC::modular_form_kernel::Laurent_series(), GiNaC::user_defined_kernel::Laurent_series(), GiNaC::Li2_series(), GiNaC::Li_series(), GiNaC::log_series(), GiNaC::S_series(), GiNaC::pseries::series(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::Eisenstein_kernel::series(), GiNaC::Eisenstein_h_kernel::series(), and GiNaC::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(), GiNaC::container< C >::op(), op(), and subs().

Referenced by GiNaC::matrix::determinant(), GiNaC::matrix::fraction_free_elimination(), GiNaC::matrix::gauss_elimination(), GiNaC::pseries::normal(), GiNaC::normal(), 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

repl	collects all temporary symbols and their replacements

Returns

rationalized expression

References bp.

Referenced by GiNaC::matrix::fraction_free_elimination(), GiNaC::power::to_polynomial(), GiNaC::expairseq::to_rational(), GiNaC::power::to_rational(), and GiNaC::to_rational().

6.48.3.54 to_polynomial()

```
ex GiNaC::ex::to_polynomial (
            exmap & repl ) const
```

References bp.

Referenced by GiNaC::find_common_factor(), GiNaC::expairseq::to_polynomial(), GiNaC::power::to_polynomial(), and GiNaC::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(), GiNaC::container< C >::op(), op(), and subs().

Referenced by 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(), GiNaC::container< C >::op(), op(), and subs().

Referenced by 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(), 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

x	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(), lcoeff(), GiNaC::info_flags::negative, and x.

Referenced by content(), GiNaC::frac_cancel(), primpart(), and unitcontprim().

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 Q[x].

The product of unit part, content part, and primitive part is the polynomial itself.

Parameters

x	main variable

Returns

content part

See also

ex::unit, ex::primpart, ex::unitcontprim

References GiNaC::_ex0, c, expand(), GiNaC::gcd(), info(), GiNaC::info_flags::integer, integer_content(), is_zero(), lcoeff(), 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.

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 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

x	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 Q[x] when the content part is already known.

This function is faster in computing the primitive part than the previous function.

Parameters

x	main variable
c	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 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.

Referenced by GiNaC::interpolate().

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.

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::integral::get_free_indices(), GiNaC::ncmul::get_free_indices(), GiNaC::add::get_free_indices(), GiNaC::mul::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

options	Simplification options (currently unused)

Returns

simplified expression

References GiNaC::simplify_indexed().

Referenced by 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

sp	Scalar products to be replaced automatically
options	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::expair::is_less(), GiNaC::expair_rest_is_less::operator()(), GiNaC::error_and_integral_is_less::operator()(), GiNaC::ex_is_less::operator()(), GiNaC::terminfo_is_less::operator()(), GiNaC::symminfo_is_less_by_symmterm::operator()(), GiNaC::symminfo_is_less_by_orig::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::atanh_eval(), GiNaC::atanh_series(), GiNaC::beta_eval(), GiNaC::pseries::coeff(), GiNaC::power::coeff(), GiNaC::add::combine_ex_with_coeff_to_pair(), GiNaC::mul::combine_pair_with_coeff_to_pair(), GiNaC::pseries::degree(), GiNaC::power::degree(), GiNaC::divide(), GiNaC::divide_in_z(), GiNaC::add::do_print_csrc(), GiNaC::power::do_print_csrc(), GiNaC::mul::do_print_csrc(), GiNaC::power::do_print_csrc_cl_N(), GiNaC::power::do_print_dflt(), GiNaC::power::do_print_latex(), GiNaC::epsilon_tensor(), 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::expair::is_canonical_numeric(), GiNaC::pseries::is_compatible_to(), GiNaC::idx::is_dummy_pair_same_type(), GiNaC::expair::is_equal(), is_zero(), GiNaC::pseries::ldegree(), GiNaC::power::ldegree(), GiNaC::Li2_eval(), GiNaC::Li2_series(), GiNaC::Li_eval(), GiNaC::log_eval(), GiNaC::lorentz_eps(), GiNaC::idx::match_same_type(), GiNaC::minimal_dim(), GiNaC::pseries::mul_series(), GiNaC::multiply_lcm(), GiNaC::power::normal(), GiNaC::idx_is_equal_ignore_dim::operator()(), GiNaC::ex_is_equal::operator()(), GiNaC::op0_is_equal::operator()(), GiNaC::spmapkey::operator==(), GiNaC::add::print_add(), GiNaC::mul::print_overall_coeff(), 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::simplify_indexed(), 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::add::eval(), GiNaC::power::eval(), GiNaC::indexed::eval(), GiNaC::function::eval(), GiNaC::pseries::evalm(), GiNaC::exp_eval(), GiNaC::pseries::expand(), GiNaC::mul::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::find_common_factor(), GiNaC::mul::find_real_imag(), GiNaC::frac_cancel(), 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::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::symbol::series(), GiNaC::power::series(), GiNaC::basic::series(), GiNaC::simplify_indexed(), GiNaC::simplify_indexed_product(), 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().

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().

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().

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::ncmul::eval(), GiNaC::exmul(), GiNaC::matrix::mul_scalar(), GiNaC::integral::return_type(), GiNaC::relational::return_type(), GiNaC::power::return_type(), and GiNaC::indexed::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::integral::return_type_tinfo(), GiNaC::relational::return_type_tinfo(), GiNaC::power::return_type_tinfo(), and GiNaC::indexed::return_type_tinfo().

6.48.3.81 gethash()

```
unsigned GiNaC::ex::gethash ( ) const  [inline]
```

References bp.

Referenced by GiNaC::idx::calchash(), GiNaC::relational::calchash(), GiNaC::basic::calchash(), and GiNaC::function::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(), and GINAC_ASSERT.

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(), 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 Function 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

e	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(), GiNaC::are_ex_trivially_equal(), coeff(), collect(), compare(), conjugate(), construct_from_basic(), dbgprint(), dbgprinttree(), degree(), denom(), diff(), eval(), eval_integ(), eval_ncmul(), evalf(), evalm(), ex(), GiNaC::ex_to(), expand(), get_free_indices(), gethash(), has(), GiNaC::archive_node::has_same_ex_as(), imag_part(), info(), integer_content(), GiNaC::is_a(), is_equal(), GiNaC::is_exactly_a(), is_polynomial(), ldegree(), let_op(), lhs(), makewriteable(), map(), match(), max_coefficient(), nops(), normal(), numer(), numer_denom(), op(), operator[](), print(), GiNaC::archive_node::printraw(), real_part(), return_type(), return_type_tinfo(), rhs(), series(), share(), smod(), 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>`

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::mul::expair_needs_further_processing().

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::add::combine_pair_with_coeff_to_pair(), GiNaC::mul::combine_pair_with_coeff_to_pair(), compare(), conjugate(), is_canonical_numeric(), is_equal(), is_less(), GiNaC::expair_rest_is_less::operator()(), print(), 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::add::combine_pair_with_coeff_to_pair(), GiNaC::mul::combine_pair_with_coeff_to_pair(), compare(), conjugate(), expair(), GiNaC::power::expand_mul(), is_canonical_numeric(), is_equal(), is_less(), print(), 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:

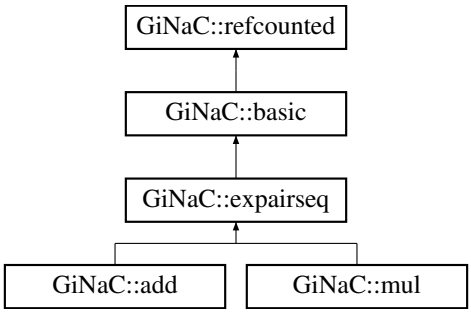
- expair.h

6.57 GiNaC::expairseq Class Reference

A sequence of class expair.

`#include <expairseq.h>`

Inheritance diagram for GiNaC::expairseq:



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 automatic non-interruptive term rewriting rules.
- 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.

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
- 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
- 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
- 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)
- void make_flat (const epvector &v, bool do_index_renaming=false)
- void canonicalize ()
- void combine_same_terms_sorted_seq ()
- bool is_canonical () const
- epvector expandchildren (unsigned options) const
- epvector evalchildren () const
- epvector subschildren (const exmap &m, unsigned options=0) const

Protected Attributes

- epvector seq
- ex overall_coeff

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 )
```

6.57.2.2 expairseq() [2/4]

```
GiNaC::expairseq::expairseq (
             const exvector & v )
```

6.57.2.3 expairseq() [3/4]

```
GiNaC::expairseq::expairseq (
             const epvector & v,
             const ex & oc,
             bool do_index_renaming = false )
```

6.57.2.4 expairseq() [4/4]

```
GiNaC::expairseq::expairseq (
             epvector && vp,
             const ex & oc,
             bool do_index_renaming = false )
```

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.

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.

6.57.3.3 nops()

```
size_t GiNaC::expairseq::nops ( ) const  [override], [virtual]
```

Number of operands/members.

Reimplemented from GiNaC::basic.

Referenced by GiNaC::algebraic_match_mul_with_mul(), GiNaC::add::conjugate(), GiNaC::add::do_print_python↩_repr(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::add::get_free_indices(), and Gi↩NaC::mul::get_free_indices().

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.

Referenced by GiNaC::algebraic_match_mul_with_mul(), GiNaC::add::conjugate(), GiNaC::add::do_print_python↩_repr(), GiNaC::add::get_free_indices(), GiNaC::mul::get_free_indices(), 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.

6.57.3.6 eval()

```
ex GiNaC::expairseq::eval ( ) const  [override], [virtual]
```

Perform automatic non-interruptive term rewriting rules.

Reimplemented from GiNaC::basic.

Reimplemented in GiNaC::mul.

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(), and GiNaC::ex::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(), and GiNaC::ex::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.

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.

6.57.3.11 conjugate()

```
ex GiNaC::expairseq::conjugate ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

Reimplemented in GiNaC::mul.

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.

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.

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.

6.57.3.15 return_type()

```
unsigned GiNaC::expairseq::return_type ( ) const  [override], [protected], [virtual]
```

Reimplemented from GiNaC::basic.

Reimplemented in GiNaC::mul.

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.

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.

6.57.3.18 thisexpairseq() [1/2]

```
virtual ex GiNaC::expairseq::thisexpairseq (
            const epvector & v,
            const ex & oc,
            bool do_index_renaming = false ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

Referenced by to_polynomial(), and to_rational().

6.57.3.19 thisexpairseq() [2/2]

```
virtual ex GiNaC::expairseq::thisexpairseq (
             epvector && vp,
             const ex & oc,
             bool do_index_renaming = false ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

6.57.3.20 printseq()

```
virtual void GiNaC::expairseq::printseq (
             const print_context & c,
             char delim,
             unsigned this_precedence,
             unsigned upper_precedence ) const  [protected], [virtual]
```

6.57.3.21 printpair()

```
virtual void GiNaC::expairseq::printpair (
             const print_context & c,
             const expair & p,
             unsigned upper_precedence ) const  [protected], [virtual]
```

6.57.3.22 split_ex_to_pair()

```
virtual expair GiNaC::expairseq::split_ex_to_pair (
             const ex & e ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

Referenced by to_polynomial(), and to_rational().

6.57.3.23 combine_ex_with_coeff_to_pair()

```
virtual expair GiNaC::expairseq::combine_ex_with_coeff_to_pair (
             const ex & e,
             const ex & c ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

6.57.3.24 combine_pair_with_coeff_to_pair()

```
virtual expair GiNaC::expairseq::combine_pair_with_coeff_to_pair (
             const expair & p,
             const ex & c ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

6.57.3.25 recombine_pair_to_ex()

```
virtual ex GiNaC::expairseq::recombine_pair_to_ex (
             const expair & p ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul, and GiNaC::add.

Referenced by to_polynomial(), and to_rational().

6.57.3.26 expair_needs_further_processing()

```
virtual bool GiNaC::expairseq::expair_needs_further_processing (
             epp it )  [protected], [virtual]
```

Reimplemented in GiNaC::mul.

6.57.3.27 default_overall_coeff()

```
virtual ex GiNaC::expairseq::default_overall_coeff ( ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul.

Referenced by to_polynomial(), and to_rational().

6.57.3.28 combine_overall_coeff() [1/2]

```
virtual void GiNaC::expairseq::combine_overall_coeff (
             const ex & c )  [protected], [virtual]
```

Reimplemented in GiNaC::mul.

6.57.3.29 combine_overall_coeff() [2/2]

```
virtual void GiNaC::expairseq::combine_overall_coeff (
            const ex & c1,
            const ex & c2 )  [protected], [virtual]
```

Reimplemented in GiNaC::mul.

6.57.3.30 can_make_flat()

```
virtual bool GiNaC::expairseq::can_make_flat (
            const expair & p ) const  [protected], [virtual]
```

Reimplemented in GiNaC::mul.

6.57.3.31 do_print()

```
void GiNaC::expairseq::do_print (
            const print_context & c,
            unsigned level ) const  [protected]
```

6.57.3.32 do_print_tree()

```
void GiNaC::expairseq::do_print_tree (
            const print_tree & c,
            unsigned level ) const  [protected]
```

6.57.3.33 construct_from_2_ex()

```
void GiNaC::expairseq::construct_from_2_ex (
            const ex & lh,
            const ex & rh )  [protected]
```

Referenced by GiNaC::add::add(), 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]
```

6.57.3.35 construct_from_expairseq_ex()

```
void GiNaC::expairseq::construct_from_expairseq_ex (
            const expairseq & s,
            const ex & e )  [protected]
```

6.57.3.36 construct_from_exvector()

```
void GiNaC::expairseq::construct_from_exvector (
            const exvector & v )  [protected]
```

Referenced by GiNaC::add::add(), 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]
```

Referenced by GiNaC::add::add(), 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]
```

6.57.3.39 make_flat() [1/2]

```
void GiNaC::expairseq::make_flat (
            const exvector & v )  [protected]
```

6.57.3.40 make_flat() [2/2]

```
void GiNaC::expairseq::make_flat (
            const epvector & v,
            bool do_index_renaming = false )  [protected]
```

6.57.3.41 canonicalize()

```
void GiNaC::expairseq::canonicalize ( )  [protected]
```

6.57.3.42 combine_same_terms_sorted_seq()

```
void GiNaC::expairseq::combine_same_terms_sorted_seq ( )  [protected]
```

6.57.3.43 is_canonical()

```
bool GiNaC::expairseq::is_canonical ( ) const  [protected]
```

Referenced by GiNaC::add::add(), and GiNaC::mul::mul().

6.57.3.44 expandchildren()

```
epvector GiNaC::expairseq::expandchildren (
            unsigned options ) const  [protected]
```

Referenced by GiNaC::add::expand().

6.57.3.45 evalchildren()

```
epvector GiNaC::expairseq::evalchildren ( ) const  [protected]
```

Referenced by GiNaC::add::eval().

6.57.3.46 subschildren()

```
epvector GiNaC::expairseq::subschildren (
            const exmap & m,
            unsigned options = 0 ) const  [protected]
```

6.57.4 Member Data Documentation

6.57.4.1 seq

```
epvector GiNaC::expairseq::seq  [protected]
```

Referenced by GiNaC::add::coeff(), GiNaC::add::degree(), GiNaC::add::derivative(), GiNaC::mul::do_print(), GiNaC::add::do_print_csrc(), GiNaC::mul::do_print_latex(), GiNaC::add::eval(), GiNaC::mul::eval(), GiNaC::add::eval_ncmul(), GiNaC::add::evalm(), GiNaC::power::expand(), GiNaC::mul::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::add::imag_part(), GiNaC::add::info(), GiNaC::add::integer_content(), GiNaC::mul::integer_content(), GiNaC::add::is_polynomial(), GiNaC::add::ldegree(), GiNaC::add::max_coefficient(), GiNaC::mul::max_coefficient(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC::add::print_add(), GiNaC::add::real_part(), GiNaC::add::return_type(), GiNaC::add::return_type_tinfo(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::add::smod(), GiNaC::mul::smod(), to_polynomial(), and to_rational().

6.57.4.2 overall_coeff

```
ex GiNaC::expairseq::overall_coeff  [protected]
```

Referenced by GiNaC::add::add(), GiNaC::add::coeff(), GiNaC::add::combine_ex_with_coeff_to_pair(), GiNaC::add::degree(), GiNaC::add::do_print_csrc(), GiNaC::mul::do_print_csrc(), GiNaC::add::eval(), GiNaC::mul::eval(), GiNaC::power::eval(), GiNaC::add::evalm(), GiNaC::add::expand(), GiNaC::power::expand(), GiNaC::mul::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::add::imag_part(), GiNaC::add::info(), GiNaC::add::integer_content(), GiNaC::mul::integer_content(), GiNaC::add::ldegree(), GiNaC::add::max_coefficient(), GiNaC::mul::max_coefficient(), GiNaC::mul::mul(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC::add::print_add(), GiNaC::mul::print_overall_coeff(), GiNaC::add::real_part(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::add::smod(), GiNaC::mul::smod(), GiNaC::add::split_ex_to_pair(), to_polynomial(), and to_rational().

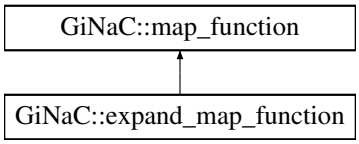
The documentation for this class was generated from the following files:

- expairseq.h
- 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:



Public Member Functions

- expand_map_function (unsigned o)
- ex operator() (const ex &e) override

Public Attributes

- unsigned options

Additional Inherited Members

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

expand_indexed	expands (a+b).i to a.i+b.i
expand_function_args	expands the arguments of functions
expand_rename_idx	used internally by mul::expand()
expand_transcendental	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

polynomial	factor only expressions that are polynomials
all	factor all polynomial subexpressions

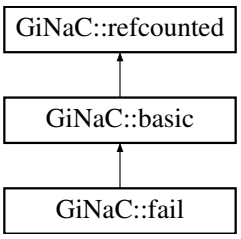
The documentation for this class was generated from the following file:

- flags.h

6.61 GiNaC::fail Class Reference

```
#include <fail.h>
```

Inheritance diagram for GiNaC::fail:



Protected Member Functions

- unsigned return_type () const override
- void do_print (const print_context &c, unsigned level) const

Additional Inherited Members

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:

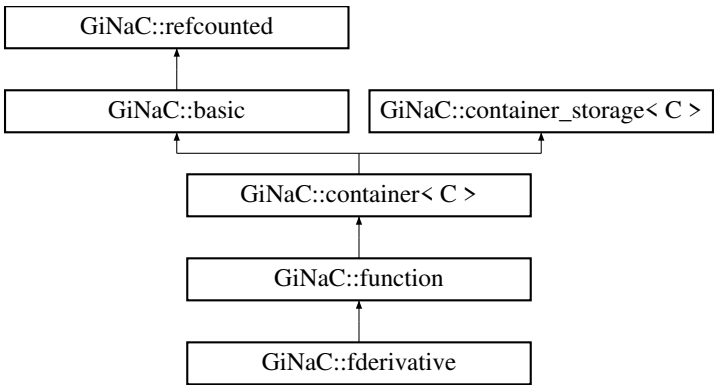
- fail.h

6.62 GiNaC::fderivative Class Reference

This class represents the (abstract) derivative of a symbolic function.

```
#include <fderivative.h>
```

Inheritance diagram for GiNaC::fderivative:



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.

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_csrc (const print_csrc &c, unsigned level) const
- void do_print_tree (const print_tree &c, unsigned level) const

Protected Attributes

- paramset parameter_set

  Set of parameter numbers with respect to which to take the derivative.

Additional Inherited Members

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

ser	Serial number of function
param	Number of parameter with respect to which to take the derivative
args	Arguments of derivative function

References parameter_set.

Referenced by derivative(), 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

ser	Serial number of function
params	Set of numbers of parameters with respect to which to take the derivative
args	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

c	print context object that describes the output formatting
level	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::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 fderivative(), GiNaC::ex::is_zero(), parameter_set, GiNaC::container_storage< C >::seq, and GiNa↩ C::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(), and thiscontainer().

The documentation for this class was generated from the following files:

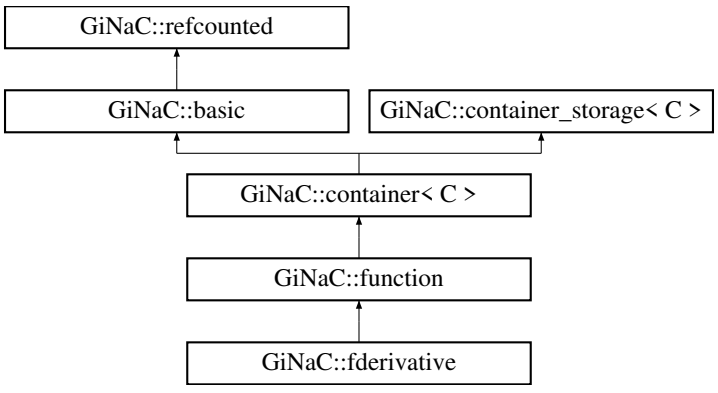
- fderivative.h
- fderivative.cpp

6.63 GiNaC::function Class Reference

The class function is used to implement builtin functions like sin, cos...

```
#include <function.h>
```

Inheritance 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.

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

Static Protected Member Functions

- static std::vector< function_options > & registered_functions ()

Protected Attributes

- unsigned serial

Friends

- class remember_table_entry

Additional Inherited Members

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 )
```

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

c	print context object that describes the output formatting
level	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(), GiNaC::function_options::print_dispatch_table, GiNaC::function_options::print_use_exvector_args, GiNaC::container< C >::printseq(), registered_functions(), GiNaC::container_storage< C >::seq, serial, and GiNaC::function_options::TeX_name.

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::expand(), 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, options, 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, GINAC_ASSERT, GiNaC::basic::hold(), 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::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 serial.

Referenced by eval().

6.63.3.10 thiscontainer() [2/2]

```
ex GiNaC::function::thiscontainer (
             exvector && v ) const  [override]
```

References 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_use_exvector_args, 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 & sym_lst )  [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_args, 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 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, GiNa↩
C::function_options::remember_strategy, 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 Function 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(), 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>
```

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)
- function_options & eval_func (eval_funcp_6 e)
- function_options & eval_func (eval_funcp_7 e)
- function_options & eval_func (eval_funcp_8 e)
- function_options & eval_func (eval_funcp_9 e)
- function_options & eval_func (eval_funcp_10 e)
- function_options & eval_func (eval_funcp_11 e)
- function_options & eval_func (eval_funcp_12 e)
- function_options & eval_func (eval_funcp_13 e)
- function_options & eval_func (eval_funcp_14 e)
- function_options & evalf_func (evalf_funcp_1 e)
- function_options & evalf_func (evalf_funcp_2 e)
- function_options & evalf_func (evalf_funcp_3 e)
- function_options & evalf_func (evalf_funcp_4 e)
- function_options & evalf_func (evalf_funcp_5 e)
- function_options & evalf_func (evalf_funcp_6 e)
- function_options & evalf_func (evalf_funcp_7 e)
- function_options & evalf_func (evalf_funcp_8 e)
- function_options & evalf_func (evalf_funcp_9 e)
- function_options & evalf_func (evalf_funcp_10 e)
- function_options & evalf_func (evalf_funcp_11 e)
- function_options & evalf_func (evalf_funcp_12 e)
- function_options & evalf_func (evalf_funcp_13 e)
- function_options & evalf_func (evalf_funcp_14 e)
- function_options & conjugate_func (conjugate_funcp_1 e)
- function_options & conjugate_func (conjugate_funcp_2 e)
- function_options & conjugate_func (conjugate_funcp_3 e)
- function_options & conjugate_func (conjugate_funcp_4 e)
- function_options & conjugate_func (conjugate_funcp_5 e)
- function_options & conjugate_func (conjugate_funcp_6 e)
- function_options & conjugate_func (conjugate_funcp_7 e)
- function_options & conjugate_func (conjugate_funcp_8 e)
- function_options & conjugate_func (conjugate_funcp_9 e)
- function_options & conjugate_func (conjugate_funcp_10 e)
- function_options & conjugate_func (conjugate_funcp_11 e)
- function_options & conjugate_func (conjugate_funcp_12 e)
- function_options & conjugate_func (conjugate_funcp_13 e)
- function_options & conjugate_func (conjugate_funcp_14 e)
- function_options & real_part_func (real_part_funcp_1 e)
- function_options & real_part_func (real_part_funcp_2 e)
- function_options & real_part_func (real_part_funcp_3 e)

- function_options & real_part_func (real_part_funcp_4 e)
- function_options & real_part_func (real_part_funcp_5 e)
- function_options & real_part_func (real_part_funcp_6 e)
- function_options & real_part_func (real_part_funcp_7 e)
- function_options & real_part_func (real_part_funcp_8 e)
- function_options & real_part_func (real_part_funcp_9 e)
- function_options & real_part_func (real_part_funcp_10 e)
- function_options & real_part_func (real_part_funcp_11 e)
- function_options & real_part_func (real_part_funcp_12 e)
- function_options & real_part_func (real_part_funcp_13 e)
- function_options & real_part_func (real_part_funcp_14 e)
- function_options & imag_part_func (imag_part_funcp_1 e)
- function_options & imag_part_func (imag_part_funcp_2 e)
- function_options & imag_part_func (imag_part_funcp_3 e)
- function_options & imag_part_func (imag_part_funcp_4 e)
- function_options & imag_part_func (imag_part_funcp_5 e)
- function_options & imag_part_func (imag_part_funcp_6 e)
- function_options & imag_part_func (imag_part_funcp_7 e)
- function_options & imag_part_func (imag_part_funcp_8 e)
- function_options & imag_part_func (imag_part_funcp_9 e)
- function_options & imag_part_func (imag_part_funcp_10 e)
- function_options & imag_part_func (imag_part_funcp_11 e)
- function_options & imag_part_func (imag_part_funcp_12 e)
- function_options & imag_part_func (imag_part_funcp_13 e)
- function_options & imag_part_func (imag_part_funcp_14 e)
- function_options & expand_func (expand_funcp_1 e)
- function_options & expand_func (expand_funcp_2 e)
- function_options & expand_func (expand_funcp_3 e)
- function_options & expand_func (expand_funcp_4 e)
- function_options & expand_func (expand_funcp_5 e)
- function_options & expand_func (expand_funcp_6 e)
- function_options & expand_func (expand_funcp_7 e)
- function_options & expand_func (expand_funcp_8 e)
- function_options & expand_func (expand_funcp_9 e)
- function_options & expand_func (expand_funcp_10 e)
- function_options & expand_func (expand_funcp_11 e)
- function_options & expand_func (expand_funcp_12 e)
- function_options & expand_func (expand_funcp_13 e)
- function_options & expand_func (expand_funcp_14 e)
- function_options & derivative_func (derivative_funcp_1 e)
- function_options & derivative_func (derivative_funcp_2 e)
- function_options & derivative_func (derivative_funcp_3 e)
- function_options & derivative_func (derivative_funcp_4 e)
- function_options & derivative_func (derivative_funcp_5 e)
- function_options & derivative_func (derivative_funcp_6 e)
- function_options & derivative_func (derivative_funcp_7 e)
- function_options & derivative_func (derivative_funcp_8 e)
- function_options & derivative_func (derivative_funcp_9 e)
- function_options & derivative_func (derivative_funcp_10 e)
- function_options & derivative_func (derivative_funcp_11 e)
- function_options & derivative_func (derivative_funcp_12 e)
- function_options & derivative_func (derivative_funcp_13 e)
- function_options & derivative_func (derivative_funcp_14 e)
- function_options & expl_derivative_func (expl_derivative_funcp_1 e)
- function_options & expl_derivative_func (expl_derivative_funcp_2 e)

- function_options & expl_derivative_func (expl_derivative_funcp_3 e)
- function_options & expl_derivative_func (expl_derivative_funcp_4 e)
- function_options & expl_derivative_func (expl_derivative_funcp_5 e)
- function_options & expl_derivative_func (expl_derivative_funcp_6 e)
- function_options & expl_derivative_func (expl_derivative_funcp_7 e)
- function_options & expl_derivative_func (expl_derivative_funcp_8 e)
- function_options & expl_derivative_func (expl_derivative_funcp_9 e)
- function_options & expl_derivative_func (expl_derivative_funcp_10 e)
- function_options & expl_derivative_func (expl_derivative_funcp_11 e)
- function_options & expl_derivative_func (expl_derivative_funcp_12 e)
- function_options & expl_derivative_func (expl_derivative_funcp_13 e)
- function_options & expl_derivative_func (expl_derivative_funcp_14 e)
- function_options & power_func (power_funcp_1 e)
- function_options & power_func (power_funcp_2 e)
- function_options & power_func (power_funcp_3 e)
- 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, 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, 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().

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, and TeX_name.

Referenced by function_options(), and initialize().

6.64.2.4 latex_name()

```
function_options & GiNaC::function_options::latex_name (
            std::string const & tn )
```

References 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().

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 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 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.

6.64.2.190 get_name()

```
std::string GiNaC::function_options::get_name ( ) const  [inline]
```

References name.

6.64.2.191 get_nparams()

```
unsigned GiNaC::function_options::get_nparams ( ) const  [inline]
```

References nparams.

6.64.2.192 has_derivative()

```
bool GiNaC::function_options::has_derivative ( ) const  [inline], [protected]
```

References derivative_f.

6.64.2.193 has_power()

```
bool GiNaC::function_options::has_power ( ) const  [inline], [protected]
```

References power_f.

6.64.2.194 test_and_set_nparams()

```
void GiNaC::function_options::test_and_set_nparams (
            unsigned n )  [protected]
```

References n, name, and nparams.

Referenced by conjugate_func(), derivative_func(), eval_func(), evalf_func(), expand_func(), expl_derivative_func(), imag_part_func(), info_func(), power_func(), print_func(), real_part_func(), and series_func().

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().

6.64.3 Friends And Related Function 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(), and initialize().

6.64.4.5 evalf_f

`evalf_funcp GiNaC::function_options::evalf_f [protected]`

Referenced by GiNaC::function::evalf(), 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(), 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(), 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(), and initialize().

6.64.4.9 expand_f

`expand_funcp GiNaC::function_options::expand_f [protected]`

Referenced by GiNaC::function::expand(), expand_func(), and initialize().

6.64.4.10 derivative_f

derivative_funcp GiNaC::function_options::derivative_f [protected]

Referenced by 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(), and initialize().

6.64.4.12 power_f

power_funcp GiNaC::function_options::power_f [protected]

Referenced by has_power(), initialize(), GiNaC::function::power(), and power_func().

6.64.4.13 series_f

series_funcp GiNaC::function_options::series_f [protected]

Referenced by initialize(), GiNaC::function::series(), 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(), 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

no_heur_gcd	Usually GiNaC 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 GiNaC to NOT use the heuristic algorithm.
no_part_factored	GiNaC 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 *really* know what are you doing!
use_sr_gcd	By default GiNaC uses modular GCD algorithm. Typically it's much faster than PRS (pseudo remainder sequence) algorithm. This flag forces GiNaC 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

algebraic	enable algebraic matching

The documentation for this class was generated from the following file:

- flags.h

6.71 std::hash< GiNaC::ex > Struct Template 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

template<>
struct std::hash< GiNaC::ex >

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:

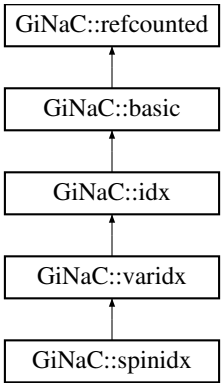
- ex.h

6.72 GiNaC::idx Class Reference

This class holds one index of an indexed object.

```
#include <idx.h>
```

Inheritance diagram for GiNaC::idx:



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.

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 Attributes

- ex value

  Expression that constitutes the index (numeric or symbolic name)
- ex dim

  Dimension of space (can be symbolic or numeric)

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

v	Value of index (numeric or symbolic)
dim	Dimension of index space (numeric or symbolic)

Returns

newly constructed index

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_calculated, 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::status_flags::hash_calculated, m, options, GiNaC::subs_options::really_subs_idx, GiNaC::ex::subs(), and value.

Referenced by GiNaC::spinmetric::contract_with().

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.

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.

Reimplemented in GiNaC::spinidx, and GiNaC::varidx.

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::spinidx, and GiNaC::varidx.

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::spinidx, and GiNaC::varidx.

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::spinidx, and GiNaC::varidx.

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:

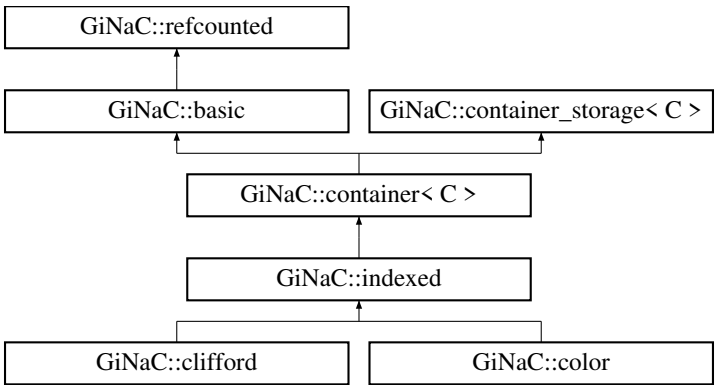
- indexed.cpp

6.74 GiNaC::indexed Class Reference

This class holds an indexed expression.

```
#include <indexed.h>
```

Inheritance diagram for GiNaC::indexed:



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 &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.

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 Attributes

- ex symtree

  Index symmetry (tree of symmetry objects)

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

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

b	Base expression

Returns

newly constructed indexed object

Referenced by GiNaC::clifford::get_metric(), 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

b	Base expression
i1	The index

Returns

newly constructed indexed object

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

b	Base expression
i1	First index
i2	Second index

Returns

newly constructed indexed object

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

b	Base expression
i1	First index
i2	Second index
i3	Third index

Returns

newly constructed indexed object

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

b	Base expression
i1	First index
i2	Second index
i3	Third index
i4	Fourth index

Returns

newly constructed indexed object

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

b	Base expression
symm	Symmetry of indices
i1	First index
i2	Second index

Returns

newly constructed indexed object

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

b	Base expression
symm	Symmetry of indices
i1	First index
i2	Second index
i3	Third index

Returns

newly constructed indexed object

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

b	Base expression
symm	Symmetry of indices
i1	First index
i2	Second index
i3	Third index
i4	Fourth index

Returns

newly constructed indexed object

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

b	Base expression
iv	Vector of indices

Returns

newly constructed indexed object

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

b	Base expression
symm	Symmetry of indices
iv	Vector of indices

Returns

newly constructed indexed object

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::eval_indexed(), GiNaC::basic::ex, GINAC_ASSE↩RT, 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 GiNaC::symmetry::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(), 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, 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 GiNaC::symmetry::validate().

6.74.4 Friends And Related Function 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

e	Object to work on
variant_dummy_indices	The set of indices that might need repositioning (will be changed by this function)
moved_indices	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(), 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:

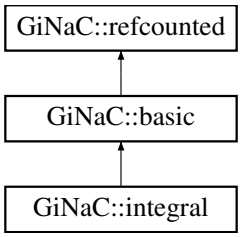
- flags.h

6.76 GiNaC::integral Class Reference

Symbolic integral.

`#include <integral.h>`

Inheritance diagram for GiNaC::integral:



Public Member Functions

- integral (const ex &x_, const ex &a_, const ex &b_, const ex &f_)
- unsigned precedence () const override

  Return relative operator precedence (for parenthezing output).
- ex eval () const override

  Perform automatic non-interruptive term rewriting rules.
- ex evalf () 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.

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

Private Attributes

- ex x
- ex a
- ex b
- ex f

Additional Inherited Members

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, 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(), 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().

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, GiNa↩C::ex::series(), GiNaC::ex::subs(), GiNaC::to_int(), and x.

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:

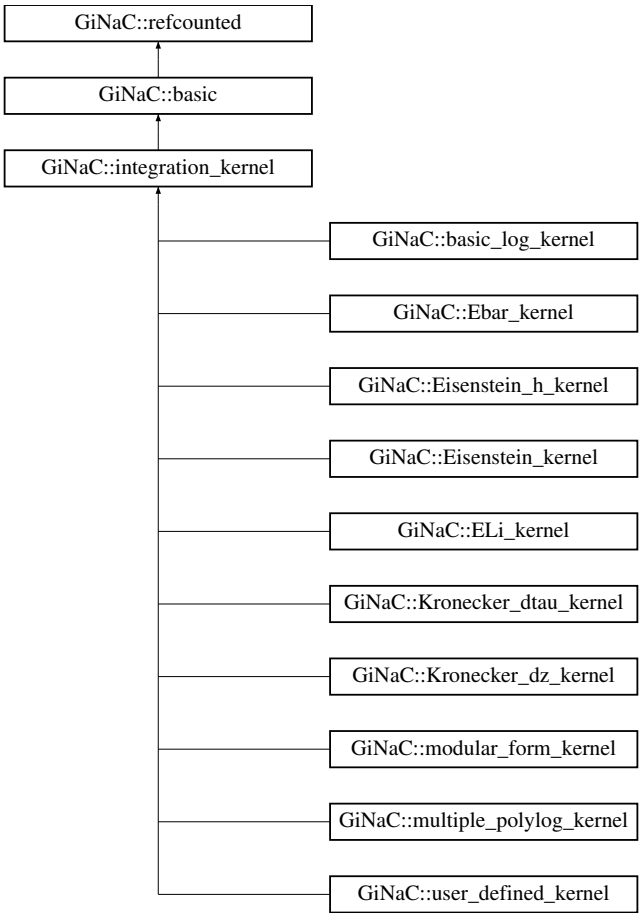
- integral.h
- indexed.cpp
- integral.cpp
- pseries.cpp

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:



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.

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 λ^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 Attributes

- int cache_step_size
- std::vector< cln::cl_N > series_vec

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, GiNaC::Eisenstein_h_kernel, and GiNaC::Eisenstein_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::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, GiNaC::ELi_kernel, and GiNaC::multiple_polylog_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 $O(x^o rder)$.

Reimplemented in GiNaC::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, and GiNaC::Eisenstein_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::modular_form_kernel, GiNaC::Eisenstein_h_kernel, GiNaC::Eisenstein_kernel, GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, and GiNaC::ELi_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::user_defined_kernel, GiNaC::modular_form_kernel, GiNaC::Eisenstein_h_kernel, and GiNaC::Eisenstein_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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{i-1}.

Reimplemented in GiNaC::Kronecker_dz_kernel, GiNaC::Kronecker_dtau_kernel, GiNaC::Ebar_kernel, GiNaC::ELi_kernel, GiNaC::multiple_polylog_kernel, and GiNaC::basic_log_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(), GiNa↩ C::Kronecker_dtau_kernel::get_numerical_value(), GiNaC::Kronecker_dz_kernel::get_numerical_value(), GiNa↩ C::Eisenstein_kernel::get_numerical_value(), GiNaC::Eisenstein_h_kernel::get_numerical_value(), and GiNaC↩ ::modular_form_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:

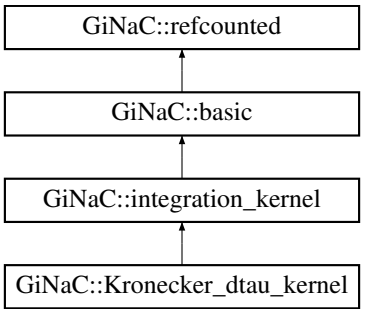
- inifcns.h
- inifcns_elliptic.cpp

6.82 GiNaC::Kronecker_dtau_kernel Class Reference

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

`#include <integration_kernel.h>`

Inheritance 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^(n)(z,K*tau), where tau is given by qbar.*

Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Protected Attributes

- ex n
- ex z
- ex K
- ex C_norm

6.82.1 Detailed Description

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

This class represents the differential one-form

$$\omega_{n,K}^{\mathrm{Kronecker},\tau}(z_j) = \frac{C_n K(n-1)}{(2\pi i)^n} g^{(n)}(z_j, K\tau)\frac{d\bar{q}}{\bar{q}}$$

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∗tau), where tau is given by qbar.

Reimplemented from GiNaC::integration_kernel.

References C_norm, GiNaC::ex::evalf(), GiNaC::basic::evalf(), GiNaC::exp(), GiNaC::Ebar_kernel::get_numerical_value(), GiNaC::integration_kernel::get_numerical_value_impl(), GiNaC::I, 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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{i-1}.

Reimplemented from GiNaC::integration_kernel.

References GiNaC::bernoulli(), C_norm, GiNaC::ex::evalf(), GiNaC::basic::evalf(), GiNaC::exp(), GiNaC::factorial(), GiNaC::I, K, n, GiNaC::Pi, GiNaC::numeric::to_int(), GiNaC::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:

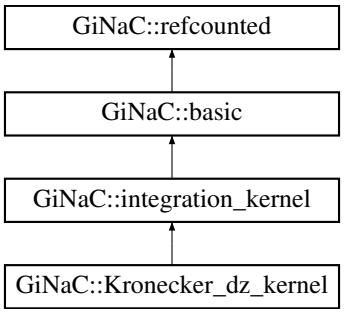
- integration_kernel.h
- integration_kernel.cpp

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:



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).

Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Protected Attributes

- ex n
- ex z_j
- ex tau
- ex K
- ex C_norm

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

$$\omega_{n,K}^{\mathrm{Kronecker},z}(z_j, \tau) = C_n(2\pi i)^{2-n} g^{(n-1)}(z - z_j, K\tau)dz$$

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::I, 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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{i-1}.

Reimplemented from GiNaC::integration_kernel.

References GiNaC::bernoulli(), C_norm, GiNaC::ex::evalf(), GiNaC::basic::evalf(), GiNaC::exp(), GiNaC::factorial(), GiNaC::Ebar_kernel::get_numerical_value(), GiNaC::I, GiNaC::is_even(), GiNaC::is_zero(), 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]
```

References GiNaC::lhs(), and GiNaC::rhs().

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 27.4.2.1.6 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().

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(), 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(), 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:

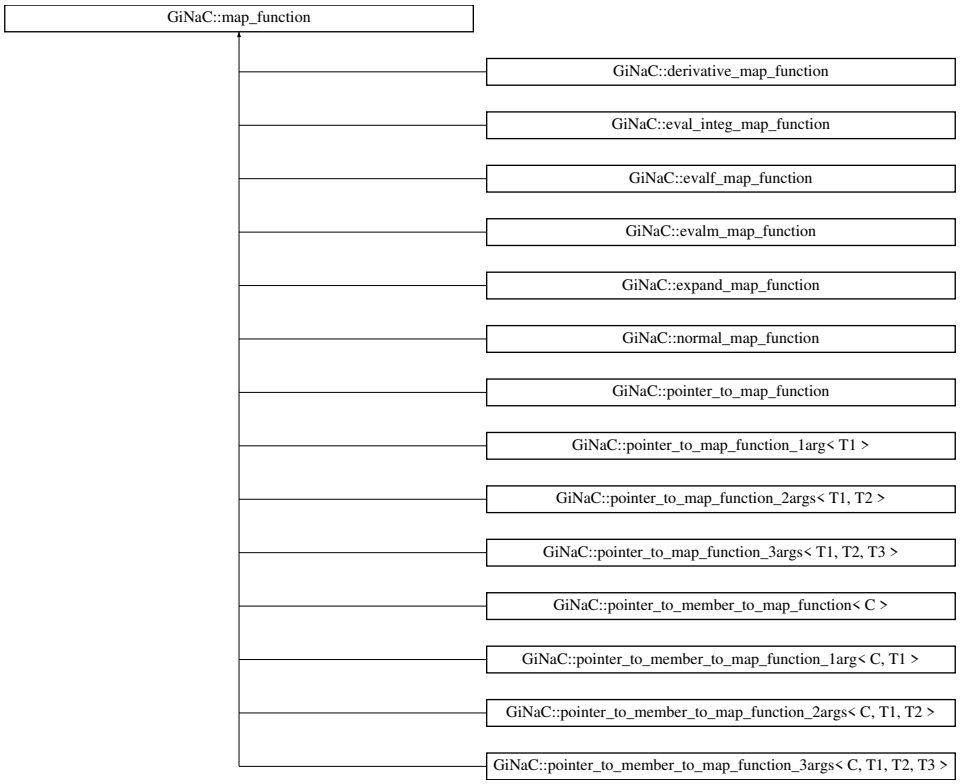
- expairseq.h

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::normal_map_function, GiNaC::pointer_to_member_to_map_function_3args< C, T1, T2, T3 >, GiNaC::pointer_to_member_to_map_function_2args< C, T1, T2 >, GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >, GiNaC::pointer_to_member_to_map_function< C >, GiNaC::pointer_to_map_function_3args< T1, T2, T3 >, GiNaC::pointer_to_map_function_2args< T1, T2 >, GiNaC::pointer_to_map_function_1arg< T1 >, GiNaC::pointer_to_map_functio GiNaC::expand_map_function, GiNaC::derivative_map_function, GiNaC::eval_integ_map_function, GiNaC::evalm_map_function, and GiNaC::evalf_map_function.

The documentation for this struct was generated from the following file:

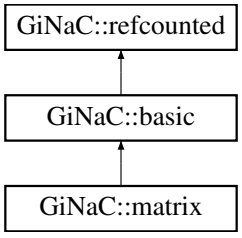
- basic.h

6.89 GiNaC::matrix Class Reference

Symbolic matrices.

```
#include <matrix.h>
```

Inheritance diagram for GiNaC::matrix:



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, i\col).
- ex & let_op (size_t i) override

  returns writable matrix entry at position (i/col, i\col).
- 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 x n matrix and a m x 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.

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 Attributes

- unsigned row

  number of rows
- unsigned col

  number of columns
- exvector m

  representation (cols indexed first)

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

r	number of rows
c	number of cols

References GiNaC::status_flags::not_shareable, and GiNaC::basic::setflag().

Referenced by add(), conjugate(), determinant(), imag_part(), 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 add_indexed(), let_op(), op(), and scalar_mul_indexed().

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 add_indexed(), and scalar_mul_indexed().

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, and row.

Referenced by fraction_free_elimination().

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, GiNaC::to_int(), 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 GINAC_ASSERT, GiNaC::ex::is_equal(), nops(), GiNaC::ex::nops(), op(), and GiNaC::ex::op().

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(), nops(), op(), 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, GINAC_ASSERT, GiNaC::is_dummy_pair(), mul(), 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, 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

logic_error	(incompatible matrices)

References col, m, matrix(), and row.

Referenced by GiNaC::add::evalm().

6.89.3.20 sub()

```
matrix GiNaC::matrix::sub (
            const matrix & other ) const
```

Difference of matrices.

Exceptions

logic_error	(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

logic_error	(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::numeric::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

ro	row of element
co	column of element

Exceptions

range_error	(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

ro	row of element
co	column of element

Exceptions

range_error	(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 x n matrix, producing a new n x m matrix object that represents the transposed.

References c, cols(), m, matrix(), r, and rows().

Referenced by contract_with().

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 [[a/(a-b),1],[b/(a-b),1]] is returned as unity. (In this respect, it behaves like MapleV and unlike Mathematica.)

Parameters

algo	allows to chose an algorithm

Returns

the determinant as a new expression

Exceptions

logic_error	(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, determinant_minor(), 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::info_flags::numeric, GiNaC::permutation_sign(), r, GiNaC::info_flags::rational_function, and row.

Referenced by charpoly().

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 [[a/(a-b),x],[y,b/(b-a)]] is recognized to be unity.

Returns

the sum of diagonal elements

Exceptions

logic_error	(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

logic_error	(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.

Referenced by pow().

6.89.3.33 inverse() [2/2]

```
matrix GiNaC::matrix::inverse (
            unsigned algo ) const
```

Inverse of this matrix.

Parameters

algo	selects the algorithm (one of solve_algo)

Returns

the inverted matrix

Exceptions

logic_error	(matrix not square)
runtime_error	(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 x n matrix and a m x p right hand side by applying an elimination scheme to the augmented matrix.

Parameters

vars	n x p matrix, all elements must be symbols
rhs	m x p matrix
algo	selects the solving algorithm

Returns

n x p solution matrix

Exceptions

logic_error	(incompatible matrices)
invalid_argument	(1st argument must be matrix of symbols)
runtime_error	(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.

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.

Referenced by determinant().

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(), 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

det	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

det	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

det	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_ASSERT, GiNaC::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

ro	is the row from where to begin
co	is the column to be inspected
symbolic	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(), archive(), charpoly(), conjugate(), determinant(), echelon_form(), eval_indexed(), imag_↩ part(), inverse(), markowitz_elimination(), match_same_type(), matrix(), mul(), mul_scalar(), nops(), 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(), archive(), charpoly(), cols(), conjugate(), determinant(), do_print_latex(), echelon_form(), eval_indexed(), imag_part(), inverse(), markowitz_elimination(), match_same_type(), matrix(), mul(), mul_scalar(), nops(), 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(), mul(), mul_scalar(), op(), 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:

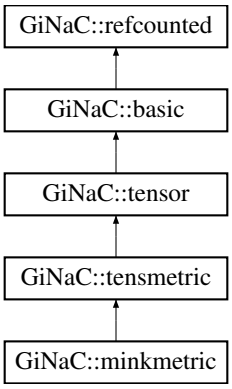
- matrix.h
- matrix.cpp

6.90 GiNaC::minkmetric Class Reference

This class represents a Minkowski metric tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::minkmetric:



Public Member Functions

- minkmetric (bool pos_sig)

  Construct Lorentz metric tensor with given signature.
- bool info (unsigned inf) 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.

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

Private Attributes

- bool pos_sig

  If true, the metric is diag(-1,1,1...).

Additional Inherited Members

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 diag(1,-1,-1,...) or diag(-1,1,1,...).

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 inf ) const  [override], [virtual]
```

Information about the object.

See also

class info_flags

Reimplemented from GiNaC::tensmetric.

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::tensmetric.

References GiNaC::_ex0, GiNaC::_ex1, GiNaC::_ex_1, GiNaC::idx::get_value(), GINAC_ASSERT, GiNaC::info_↩ flags::nonnegint, GiNaC::basic::nops(), GiNaC::basic::op(), pos_sig, and GiNaC::to_int().

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::tensmetric.

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 diag(-1,1,1...).

Otherwise it is diag(1,-1,-1,...).

Referenced by archive(), eval_indexed(), and read_archive().

The documentation for this class was generated from the following files:

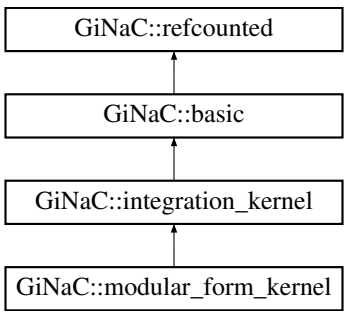
- tensor.h
- tensor.cpp

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:



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

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 Attributes

- ex k
- ex P
- ex C_norm

6.91.1 Detailed Description

A kernel corresponding to a polynomial in Eisenstein series.

This class represents the differential one-form

$$\omega^{\mathrm{modular}}(P_k(\eta_{k_1}^{(1)},\ldots,\eta_{k_r}^{(r)})) = C_k P_k(\eta_{k_1}^{(1)},\ldots,\eta_{k_r}^{(r)})\frac{d\bar{q}_N}{\bar{q}_N}.$$

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 order, P, qbar, and r.

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::basic::evalf(), GiNaC::ex::info(), k, GiNaC::info_flags::nonnegint, GiNaC::info_flags::numeric, P, qbar, GiNaC::ex::series(), GiNaC::series_to_poly(), and GiNaC::basic::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, P, 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().

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(), is_numeric(), Laurent_series(), 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:

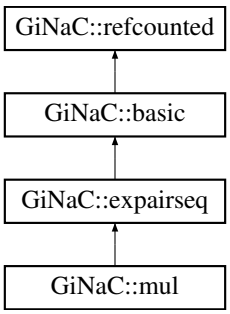
- utils.h

6.93 GiNaC::mul Class Reference

Product of expressions.

`#include <mul.h>`

Inheritance diagram for GiNaC::mul:



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

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
- ex thisexpairseq (epvector &&vp, const ex &oc, bool do_index_renaming=false) const override
- expair split_ex_to_pair (const ex &e) const override
- 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
- bool expair_needs_further_processing (epp 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.

Static Protected Member Functions

- static bool can_be_further_expanded (const ex &e)

Friends

- class add
- class ncmul
- class power

Additional Inherited Members

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().

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_csrc(), do_print_latex(), and print_overall_coeff().

6.93.3.2 info()

```
bool GiNaC::mul::info (
            unsigned inf ) const  [override], [virtual]
```

Information about the object.

See also

class info_flags

Reimplemented from GiNaC::expairseq.

References GiNaC::_num1_p, GiNaC::factor(), GiNaC::numeric::info(), GiNaC::ex::info(), GiNaC::is_negative(), GiNaC::is_positive(), and GiNaC::real().

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.

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.

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.

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(), GiNaC::coeff(), and n.

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::algebraic_match_mul_with_mul(), GiNaC::has(), GiNaC::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, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- ∗(...,x;0) -> 0
- ∗(+(x1,x2,...);c) -> ∗(+(∗(x1,c),∗(x2,c),...))
- ∗(x;1) -> x
- ∗(;c) -> c

Reimplemented from GiNaC::expairseq.

References GiNaC::_ex0, GiNaC::_ex1, GiNaC::_num1_p, GiNaC::_num_1_p, c, GiNaC::basic::clearflag(), GiNaC::coeff(), GiNaC::add::combine_pair_with_coeff_to_pair(), GINAC_ASSERT, GiNaC::numeric::is_integer(), GiNaC::numeric::is_pos_integer(), last, likely, GiNaC::numeric::mul(), GiNaC::expairseq::overall_coeff, GiNaC::expairseq::seq, and unlikely.

6.93.3.9 evalf()

```
ex GiNaC::mul::evalf ( ) const  [override], [virtual]
```

Evaluate object numerically.

Reimplemented from GiNaC::basic.

6.93.3.10 real_part()

```
ex GiNaC::mul::real_part ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

6.93.3.11 imag_part()

```
ex GiNaC::mul::imag_part ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

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, and m.

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::begin(), GiNaC::ex::coeff(), GiNaC::ex::expand(), GiNaC::factor(), GINAC_↩ ASSERT, GiNaC::ex::info(), GiNaC::info_flags::integer, GiNaC::ex::ldegree(), GiNaC::pseries::mul_const(), GiNa↩ C::pseries::mul_series(), GiNaC::expairseq::op(), options, order, GiNaC::expairseq::overall_coeff, r, recombine_↩ pair_to_ex(), GiNaC::expairseq::seq, GiNaC::ex::series(), and GiNaC::to_int().

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, 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::begin(), GiNaC::basic::diff(), GiNaC::pow(), 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::ex::eval_ncmul().

6.93.3.22 return_type()

```
unsigned GiNaC::mul::return_type ( ) const  [override], [protected], [virtual]
```

Reimplemented from GiNaC::expairseq.

6.93.3.23 return_type_tinfo()

```
return_type_t GiNaC::mul::return_type_tinfo ( ) const  [override], [protected], [virtual]
```

Reimplemented from GiNaC::basic.

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]
```

Reimplemented from GiNaC::expairseq.

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]
```

Reimplemented from GiNaC::expairseq.

References GiNaC::_ex1, GiNaC::power::basis, and GiNaC::power::exponent.

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, and GiNaC::pow().

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(), and GiNaC::expair::rest.

6.93.3.29 recombine_pair_to_ex()

```
ex GiNaC::mul::recombine_pair_to_ex (
            const expair & p ) const  [override], [protected], [virtual]
```

Reimplemented from GiNaC::expairseq.

References GiNaC::_ex1, GiNaC::expair::coeff, GiNaC::ex::is_equal(), and GiNaC::expair::rest.

Referenced by do_print(), do_print_latex(), integer_content(), max_coefficient(), normal(), series(), and smod().

6.93.3.30 expair_needs_further_processing()

```
bool GiNaC::mul::expair_needs_further_processing (
            epp it )  [override], [protected], [virtual]
```

Reimplemented from GiNaC::expairseq.

References GiNaC::_ex1, and GiNaC::expair::is_equal().

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, and GINAC_ASSERT.

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.

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, and GiNaC::ex::info().

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, GiNaC::ex::expand(), factors, GiNaC::get_all_dummy_indices_safely(), GiNaC::ex::is_equal(), GiNaC::ex::is_zero(), n, GiNaC::ex::nops(), GiNaC::container< C >::op(), GiNaC::ex::op(), options, GiNaC::expairseq::overall_coeff, GiNaC::rename_dummy_indices_uniquely(), and GiNaC::expairseq::seq.

6.93.3.36 algebraic_subs_mul()

```
ex GiNaC::mul::algebraic_subs_mul (
            const exmap & m,
            unsigned options ) const
```

References GiNaC::algebraic_match_mul_with_mul(), m, GiNaC::nops(), GiNaC::op(), options, GiNaC::pow(), GiNaC::ex::subs(), 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(), and GiNaC::ex::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(), 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::ex::is_equal(), GiNaC::expairseq::overall_coeff, precedence(), and GiNaC::ex::print().

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::nops(), GiNaC::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(), and GiNaC::ex::op().

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 safe 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::begin(), GiNaC::ex::expand(), GiNaC::factor(), last, and options.

6.93.4 Friends And Related Function 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]
```

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

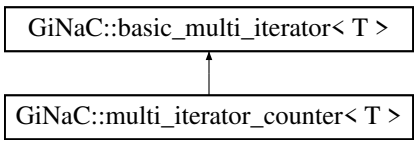
The class multi_iterator_counter defines a multi_iterator $(i_1, i_2, ..., i_k)$, such that

$$B \leq i_j < N$$

.

`#include <utils_multi_iterator.h>`

Inheritance 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
  $$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$
  .
- 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.

Friends

- template<class TT >
  std::ostream & operator<< (std::ostream &os, const multi_iterator_counter< TT > &v)

Additional Inherited Members

6.94.1 Detailed Description

template<class T>
class GiNaC::multi_iterator_counter< T >

The class multi_iterator_counter defines a multi_iterator $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$

.

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 Function 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

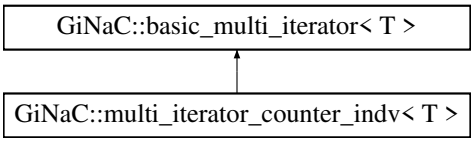
The class multi_iterator_counter_indv defines a multi_iterator $(i_1, i_2, ..., i_k)$, such that

$$B \leq i_j < N_j$$

.

```
#include <utils_multi_iterator.h>
```

Inheritance 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
  $$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$
  .
- 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.

Protected Attributes

- std::vector< T > Nv

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 $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, n_3, ..., n_k) = (B, B, ..., B)$$

.

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 Function 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 $(i_1, i_2, ..., i_k)$, such that

$$B \leq i_j < N$$

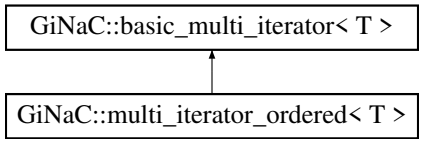
and

$$i_j < i_{j+1}.$$

It is assumed that $k > 0$ and $N - B \geq k$.

```
#include <utils_multi_iterator.h>
```

Inheritance 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

$$(n_1, n_2, n_3, ..., n_k) = (B+0, B+1, B+2, ..., B+k-1)$$

  .
- 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.

Friends

- template<class TT >
  std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered< TT > &v)

Additional Inherited Members

6.96.1 Detailed Description

template<class T>
class GiNaC::multi_iterator_ordered< T >

The class multi_iterator_ordered defines a multi_iterator $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, n_3, ..., n_k) = (B+0, B+1, B+2, ..., B+k-1)$$

.

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 Function 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 $(i_1, i_2, ..., i_k)$, such that

$$B \leq i_j < N$$

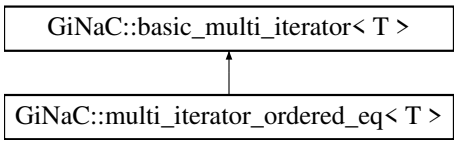
and

$$i_j \leq i_{j+1}.$$

It is assumed that $k > 0$.

```
#include <utils_multi_iterator.h>
```

Inheritance 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

  $$(n_1, n_2, ..., n_k) = (B, B, ..., B)$$

  .
- 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.

Friends

- template<class TT >
  std::ostream & operator<< (std::ostream &os, const multi_iterator_ordered_eq< TT > &v)

Additional Inherited Members

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 $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, ..., n_k) = (B, B, ..., B)$$

.

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 Function 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 $(i_1, i_2, ..., i_k)$, such that

$$B \leq i_j < N_j$$

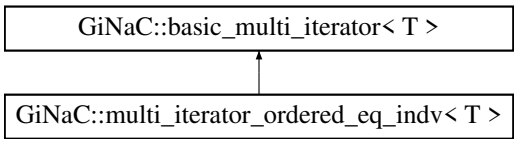
and

$$i_j \leq i_{j+1}.$$

.

```
#include <utils_multi_iterator.h>
```

Inheritance 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
  $$(n_1, n_2, n_3, ..., n_k) = (B, B, B, ..., B)$$
  .
- 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.

Protected Attributes

- std::vector< T > Nv

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 $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, n_3, ..., n_k) = (B, B, B, ..., B)$$

.

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 Function 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 $(i_1, i_2, ..., i_k)$, for which

$$B \leq i_j < N$$

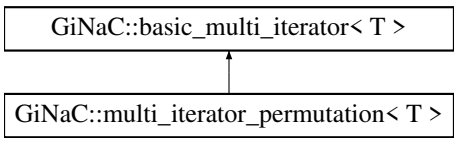
and

$$i_i \neq i_j$$

In particular, if $N - B = k$, multi_iterator_permutation loops over all permutations of k elements.

```
#include <utils_multi_iterator.h>
```

Inheritance 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

  $$(n_1, n_2, n_3, ..., n_k) = (B + 0, B + 1, B + 2, ..., B + k - 1)$$

  .
- 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

  $$(-1)^{n_{inv}},$$

  where n_{inv} is the number of inversions, e.g.

Friends

- template<class TT >
  std::ostream & operator<< (std::ostream &os, const multi_iterator_permutation< TT > &v)

Additional Inherited Members

6.99.1 Detailed Description

template<class T>
class GiNaC::multi_iterator_permutation< T >

The class multi_iterator_permutation defines a multi_iterator $(i_1, i_2, ..., i_k)$, 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

$$(n_1, n_2, n_3, ..., n_k) = (B+0, B+1, B+2, ..., B+k-1)$$

.

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 Function 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:

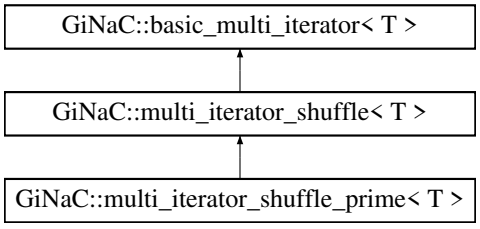
- utils_multi_iterator.h

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 >:



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.

Protected Attributes

- size_t N_internal
- std::vector< size_t > v_internal
- std::vector< T > v_orig

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 >::v, 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.

6.100.4 Friends And Related Function 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:

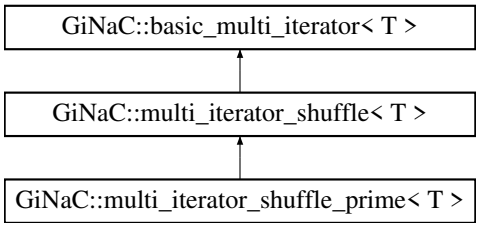
- utils_multi_iterator.h

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 >:



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.

Friends

- template<class TT >
  std::ostream & operator<< (std::ostream &os, const multi_iterator_shuffle_prime< TT > &v)

Additional Inherited Members

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 Function 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:

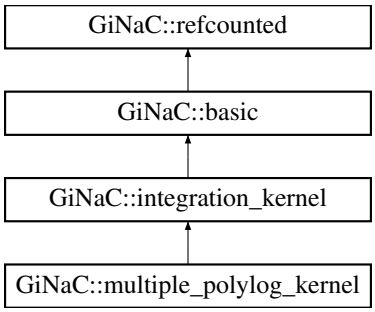
- utils_multi_iterator.h

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:



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.

Protected Member Functions

- cln::cl_N series_coeff_impl (int i) const override

  For $\omega = d\lambda$ only the coefficient of λ^0 is non-zero.
- void do_print (const print_context &c, unsigned level) const

Protected Attributes

- ex z

6.102.1 Detailed Description

The integration kernel for multiple polylogarithms.

This class represents the differential one-form

$$\omega^{\mathrm{mpl}}(z) = \frac{d\lambda}{\lambda - z}$$

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 λ^0 is non-zero.

The i-th coefficient corresponds to the power λ^{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:

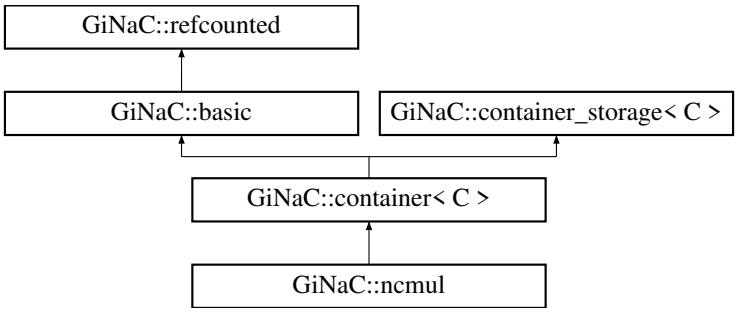
- integration_kernel.h
- integration_kernel.cpp

6.103 GiNaC::ncmul Class Reference

Non-commutative product of expressions.

`#include <ncmul.h>`

Inheritance diagram for GiNaC::ncmul:



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

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_csrc (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

Friends

- class power
- ex reeval_ncmul (const exvector &v)
- ex hold_ncmul (const exvector &v)

Additional Inherited Members

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::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(), GiNaC::ex::return_type(), GiNaC::container_storage< C >::seq, and GiNaC::basic::setflag().

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::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, factors, GiNaC::ex::nops(), GiNaC::ex::op(), and GiNaC::ex::return_type().

Referenced by eval().

6.103.3.21 append_factors()

```
void GiNaC::ncmul::append_factors (
             exvector & v,
             const ex & e ) const  [protected]
```

References GiNaC::return_types::commutative, GiNaC::ex::nops(), GiNaC::ex::op(), and GiNaC::ex::return_type().

Referenced by 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 Function 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]
```

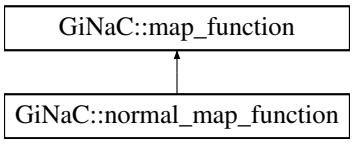
The documentation for this class was generated from the following files:

- 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:



Public Member Functions

- ex operator() (const ex &e) override

Additional Inherited Members

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:

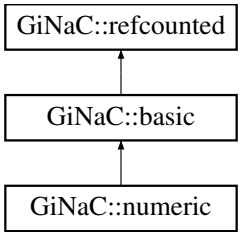
- normal.cpp

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:



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.

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 Attributes

- cln::cl_N 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=(), 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

overflow_error	(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(), GiNaC::mul::info(), 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) -> true. But this is consistent, since we also would like to have (-2+I).has(2) -> 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 1.00000000000000000000 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().

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().

6.105.3.17 conjugate()

```
ex GiNaC::numeric::conjugate ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

References GiNaC::conjugate(), 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::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-(), and GiNaC::operator-=().

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::power::expand_add_2(), GiNaC::lcm(), GiNaC↩
::operator∗(), GiNaC::operator∗=(), GiNaC::operator-(), and GiNaC::matrix::pow().

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

overflow_error	(division by zero)

References numeric(), and value.

Referenced by GiNaC::power::eval(), GiNaC::multinomial_coefficient(), GiNaC::operator/(), GiNaC::operator/=(), and GiNaC::basic::series().

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

overflow_error	(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().

Referenced by 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.

csgn(x)==0 for x==0, csgn(x)==1 for Re(x)>0 or Re(x)=0 and Im(x)>0, csgn(x)==-1 for Re(x)<0 or Re(x)=0 and 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::cos_eval(), GiNaC::power::eval(), GiNaC::exp_eval(), has(), is_equal_same_type(), 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::cosh_eval(), GiNaC::csgn_eval(), GiNaC::csgn_series(), GiNaC::power::eval(), GiNaC::fsolve(), GiNaC::gcd(), has(), info(), GiNaC::iquo(), GiNaC::irem(), GiNaC::matrix::pivot(), GiNaC::matrix::pow(), GiNaC::pseries::power_const(), GiNaC::sinh_eval(), GiNaC::step_eval(), GiNaC::step_series(), GiNaC::tanh_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(), 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::irem(), GiNaC::lcm(), GiNaC::mod(), normal(), GiNaC::psi1_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(), and info().

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(), numer(), 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+b∗I, 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

invalid_argument	(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

invalid_argument	(complex inequality)

References is_real(), and value.

6.105.3.63 operator>()

```
bool GiNaC::numeric::operator> (
            const numeric & other ) const
```

Numerical comparison: greater.

Exceptions

invalid_argument	(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

invalid_argument	(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::eval(), GiNaC::power::expand(), 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::irem(), GiNaC::lcm(), GiNaC::mod(), 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(), GiNaC::lcm(), 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(), GiNaC::lcm(), 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 >= 0 such that -2^n <= x < 2^n. If x > 0, this is the unique n > 0 such that 2^(n-1) <= 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(), 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:

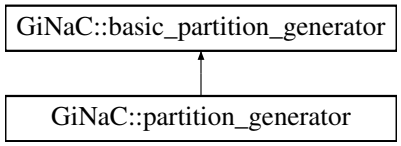
- ex.h

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:



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

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_partition().

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:

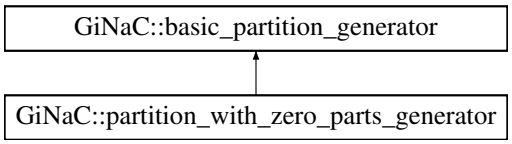
- utils.h

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:



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

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().

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().

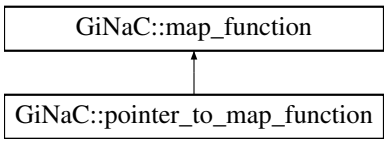
The documentation for this class was generated from the following file:

- utils.h

6.109 GiNaC::pointer_to_map_function Class Reference

```
#include <ex.h>
```

Inheritance diagram for GiNaC::pointer_to_map_function:



Public Member Functions

- pointer_to_map_function (ex x(const ex &))
- ex operator() (const ex &e) override

Protected Attributes

- ex(∗ ptr)(const ex &)

Additional Inherited Members

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()().

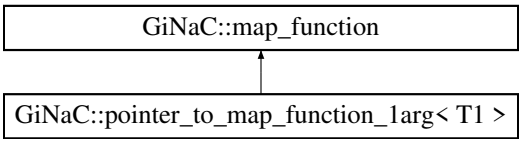
The documentation for this class was generated from the following file:

- 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 >:



Public Member Functions

- pointer_to_map_function_1arg (ex x(const ex &, T1), T1 a1)
- ex operator() (const ex &e) override

Protected Attributes

- ex(∗ ptr)(const ex &, T1)
- T1 arg1

Additional Inherited Members

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()().

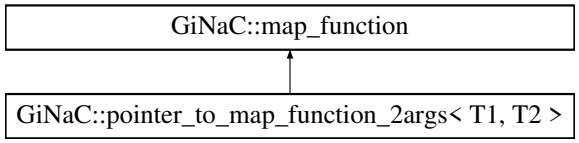
The documentation for this class was generated from the following file:

- 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 >:



Public Member Functions

- pointer_to_map_function_2args (ex x(const ex &, T1, T2), T1 a1, T2 a2)
- ex operator() (const ex &e) override

Protected Attributes

- ex(∗ ptr)(const ex &, T1, T2)
- T1 arg1
- T2 arg2

Additional Inherited Members

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()().

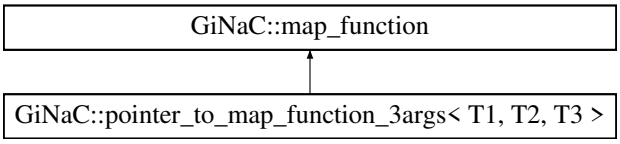
The documentation for this class was generated from the following file:

- 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 >:



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

Protected Attributes

- ex(∗ ptr)(const ex &, T1, T2, T3)
- T1 arg1
- T2 arg2
- T3 arg3

Additional Inherited Members

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 >::arg2, 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()().

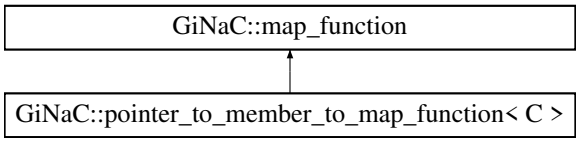
The documentation for this class was generated from the following file:

- 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 >:



Public Member Functions

- pointer_to_member_to_map_function (ex(C::∗member)(const ex &), C &obj)
- ex operator() (const ex &e) override

Protected Attributes

- ex(C::∗ ptr)(const ex &)
- C & c

Additional Inherited Members

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()().

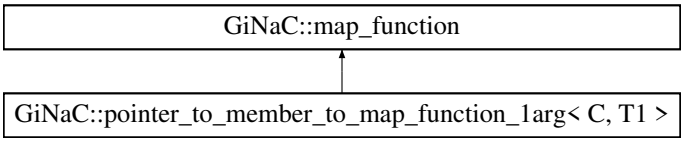
The documentation for this class was generated from the following file:

- 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 >:



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

Protected Attributes

- ex(C::∗ ptr)(const ex &, T1)
- C & c
- T1 arg1

Additional Inherited Members

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::*)(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< C, T1 >::c.

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_1arg< 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_1arg< 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_1arg< C, T1 >::arg1  [protected]
```

Referenced by GiNaC::pointer_to_member_to_map_function_1arg< C, T1 >::operator()().

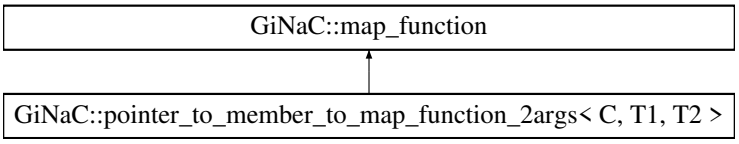
The documentation for this class was generated from the following file:

- 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 >:



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

Protected Attributes

- ex(C::* ptr)(const ex &, T1, T2)
- C & c
- T1 arg1
- T2 arg2

Additional Inherited Members

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 >::arg2, 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()().

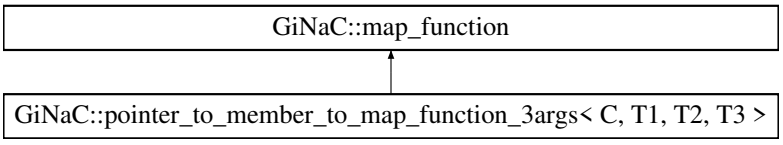
The documentation for this class was generated from the following file:

- 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 >:



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

Protected Attributes

- ex(C::* ptr)(const ex &, T1, T2, T3)
- C & c
- T1 arg1
- T2 arg2
- T3 arg3

Additional Inherited Members

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 >::c.

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:

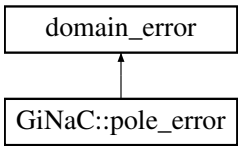
- ex.h

6.117 GiNaC::pole_error Class Reference

Exception class thrown when a singularity is encountered.

`#include <numeric.h>`

Inheritance diagram for GiNaC::pole_error:



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:

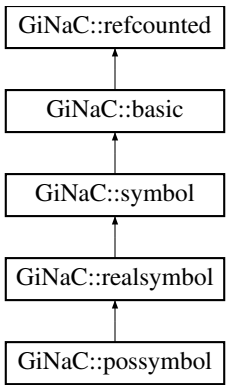
- numeric.h
- utils.cpp

6.118 GiNaC::possymbol Class Reference

Specialization of symbol to real positive domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::possymbol:



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.

Additional Inherited Members

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::realsymbol.

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::realsymbol.

References GiNaC::status_flags::dynallocated, possymbol(), and GiNaC::basic::setflag().

The documentation for this class was generated from the following files:

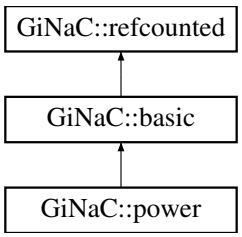
- symbol.h
- symbol.cpp

6.119 GiNaC::power Class Reference

This class holds a two-component object, a basis and and exponent representing exponentiation.

```
#include <power.h>
```

Inheritance 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) override

  Read (a.k.a.

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

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

Friends

- class mul

6.119.1 Detailed Description

This class holds a two-component object, a basis and and 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(), expand_add(), expand_add_2(), and subs().

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 inf ) 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::ex::is_equal(), GiNaC::basic::is_↩ equal(), GiNaC::is_integer(), and GiNaC::to_int().

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::ex::is_equal(), GiNaC::basic::is_equal(), GiNaC::is_↩ integer(), GiNaC::ex::ldegree(), and GiNaC::to_int().

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::ex::is_equal(), GiNaC::basic::is_equal(), GiNaC::is_integer(), n, and GiNaC::to_int().

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, x1, x2,... stand for a symbolic variables of type ex and c, c1, c2... stand for such expressions that contain a plain number.

- ^(x,0) -> 1 (also handles ^(0,0))
- ^(x,1) -> x
- ^(0,c) -> 0 or exception (depending on the real part of c)
- ^(1,x) -> 1
- ^(c1,c2) -> *(c1^n,c1^(c2-n)) (so that 0<(c2-n)<1, try to evaluate roots, possibly in numerator and denominator of c1)
- ^(^(x,c1),c2) -> ^(x,c1*c2) if x is positive and c1 is real.
- ^(^(x,c1),c2) -> ^(x,c1*c2) (c2 integer or -1 < c1 <= 1 or (c1=-1 and c2>0), case c1=1 should not happen, see below!)
- ^(*(x,y,z),c) -> *(x^c,y^c,z^c) (if c integer)
- ^(*(x,c1),c2) -> ^(x,c2)*c1^c2 (c1>0)
- ^(*(x,c1),c2) -> ^(-x,c2)*c1^c2 (c1<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::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::numeric::is_equal(), GiNaC::ex::is_equal(), GiNaC::numeric::is_integer(), GiNaC::is_negative(), GiNaC::numeric::is_pos_integer(), GiNaC::numeric::is_positive(), GiNaC::numeric::is_rational(), GiNaC::numeric::is_real(), GiNaC::numeric::is_zero(), GiNaC::ex::is_zero(), GiNaC::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::numeric::real(), GiNaC::info_flags::real, GiNaC::ex::return_type(), 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::ldegree(), GiNaC::log(), GiNaC::info_flags::negint, GiNaC::subs_options::no_pattern, options, order, r, GiNaC::info_flags::rational_function, GiNaC::ex::series(), GiNaC::basic::series(), GiNaC::ex::subs(), and GiNaC::to_int().

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(), power(), GiNaC::ex::subs(), GiNaC::basic::subs_one_level(), and Gi↩ NaC::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↩ ::container< C >::op(), GiNaC::ex::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(), GiNaC::ex::to_polynomial(), and GiNaC::ex::to_rational().

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, Gi↩NaC::ex::imag_part(), GiNaC::ex::info(), GiNaC::info_flags::integer, GiNaC::ex::is_equal(), GiNaC::log(), n, GiNa↩C::info_flags::nonnegative, GiNaC::numer(), GiNaC::pow(), GiNaC::ex::real_part(), and GiNaC::to_long().

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(), GiNaC::sin(), and GiNaC::to_long().

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::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, GiNa↩
C::ex::is_equal(), GiNaC::ex::print(), GiNaC::print_sym_pow(), and GiNaC::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::composition_generator::next(), GiNaC::expairseq::nops(), options, GiNaC::expairseq::overall_coeff, GiNaC::pow(), power(), r, GiNaC::expairseq::seq, GiNaC::basic::setflag(), 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, mul, GiNaC::numeric::mul(), GiNaC::numeric::mul_dyn(), GiNaC::expairseq::nops(), options, GiNaC::expairseq::overall_coeff, power(), r, GiNaC::expairseq::seq, and GiNaC::basic::setflag().

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 Function 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:

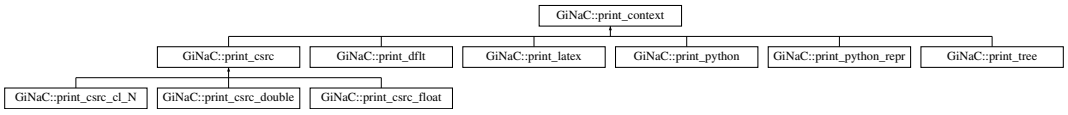
- power.h
- normal.cpp
- power.cpp
- pseries.cpp

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::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:

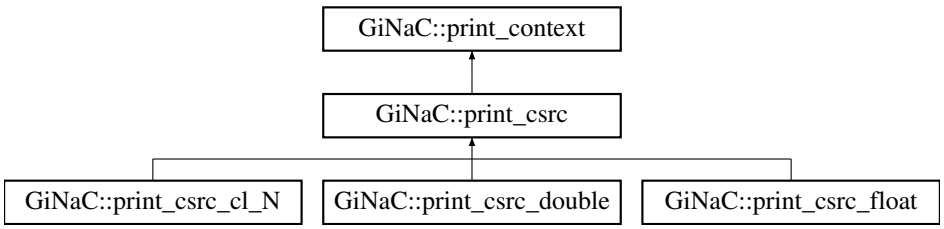
- print.h

6.122 GiNaC::print_csrc Class Reference

Base context for C source output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print_csrc:



Public Member Functions

- print_csrc (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

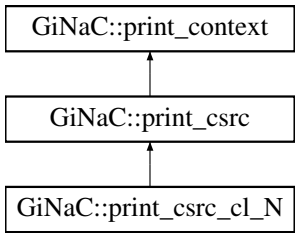
- print.h
- print.cpp

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:



Public Member Functions

- print_csrc_cl_N (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

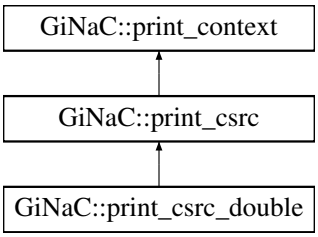
- print.h
- print.cpp

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:



Public Member Functions

- print_csrc_double (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

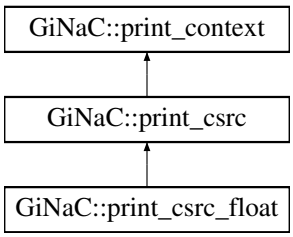
- print.h
- print.cpp

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:



Public Member Functions

- print_csrc_float (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

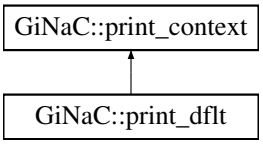
- print.h
- print.cpp

6.126 GiNaC::print_dflt Class Reference

Context for default (ginsh-parsable) output.

`#include <print.h>`

Inheritance diagram for GiNaC::print_dflt:



Public Member Functions

- print_dflt (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

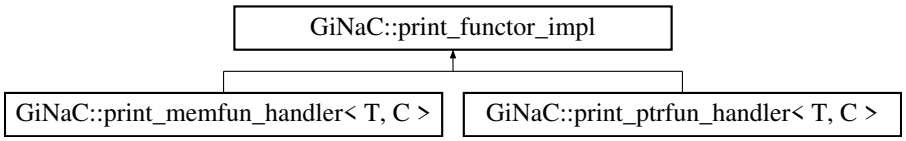
- print.h

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_memfun_handler< T, C >, and GiNaC::print_ptrfun_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_memfun_handler< T, C >, and GiNaC::print_ptrfun_handler< T, C >.

The documentation for this class was generated from the following file:

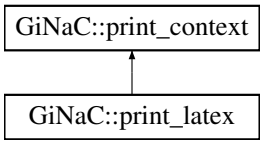
- print.h

6.129 GiNaC::print_latex Class Reference

Context for latex-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print_latex:



Public Member Functions

- print_latex (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

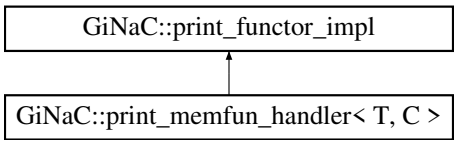
- print.h
- print.cpp

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 >:



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

Private Attributes

- F f

6.130.1 Detailed Description

template<class T, class C>
class GiNaC::print_memfun_handler< T, C >

print_functor 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_functor_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_functor_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

print_index_dimensions	print the dimensions of indices

The documentation for this class was generated from the following file:

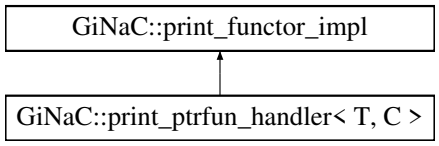
- print.h

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 >:



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

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:

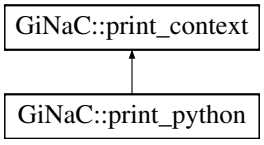
- print.h

6.133 GiNaC::print_python Class Reference

Context for python pretty-print output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print_python:



Public Member Functions

- print_python (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

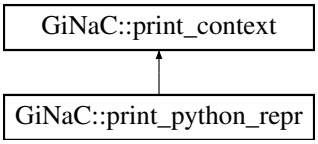
- print.h
- print.cpp

6.134 GiNaC::print_python_repr Class Reference

Context for python-parsable output.

```
#include <print.h>
```

Inheritance diagram for GiNaC::print_python_repr:



Public Member Functions

- print_python_repr (std::ostream &, unsigned options=0)

Additional Inherited Members

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:

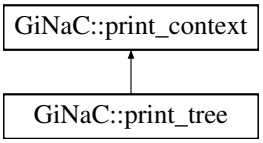
- print.h
- print.cpp

6.135 GiNaC::print_tree Class Reference

Context for tree-like output for debugging.

`#include <print.h>`

Inheritance diagram for GiNaC::print_tree:



Public Member Functions

- print_tree (unsigned d)
- print_tree (std::ostream &, unsigned options=0, unsigned d=4)

Public Attributes

- const unsigned delta_indent

  size of indentation step

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:

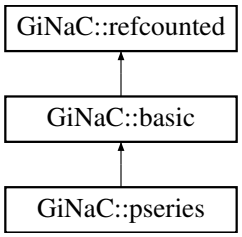
- archive.h

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:



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 ith 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.

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 Attributes

- epvector seq

  Vector of {coefficient, power} pairs.
- ex var

  Series variable (holds a symbol)
- ex point

  Expansion point.

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_*	expansion variable and point (must hold a relational)
*ops_*	vector of {coefficient, power} pairs (coefficient must not be zero)

Returns

newly constructed pseries

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 coeff(), GiNaC::ex::is_equal(), seq, and var.

Referenced by mul_series(), 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 + Order((1-x)^3) has ldegree(x) 1, not 0.

Reimplemented from GiNaC::basic.

References coeff(), 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+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(), GiNaC::basic::compare(), convert_to_poly(), GINAC_ASSERT, GiNaC::ex::is_equal(), n, seq, and var.

Referenced by degree(), ldegree(), 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, GiNaC::basic::setflag(), 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, seq, GiNaC::ex::series(), GiNaC::to_int(), 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, GiNaC::basic::setflag(), 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::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

no_order	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

other	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, GiNaC::to_int(), 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

other	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

other	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(), ldegree(), point, pseries(), seq, GiNaC::to_int(), 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

p	power to compute
deg	truncation order of series calculation

References GiNaC::_ex0, GiNaC::_ex1, c, coeff(), GiNaC::is_integer(), GiNaC::numeric::is_negative(), GiNaC::is_order_function(), is_zero(), GiNaC::numeric::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::ex::print(), GiNaC::basic::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(), 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(), 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(), 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>
```

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 makewriteable() 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.

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.

6.141.2.3 ptr() [3/3]

```
template<class T>
GiNaC::ptr< T >::ptr (
            const ptr< T > & other )  [inline], [noexcept]
```

6.141.2.4 ~ptr()

```
template<class T>
GiNaC::ptr< T >::~ptr ( )  [inline]
```

6.141.3 Member Function Documentation

6.141.3.1 operator=()

```
template<class T>
ptr& GiNaC::ptr< T >::operator= (
            const ptr< T > & other )  [inline]
```

6.141.3.2 operator*()

```
template<class T>
T& GiNaC::ptr< T >::operator* ( ) const  [inline], [noexcept]
```

6.141.3.3 operator->()

```
template<class T>
T* GiNaC::ptr< T >::operator-> ( ) const  [inline], [noexcept]
```

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.

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.

6.141.3.6 operator==()

```
template<class T>
template<class U >
bool GiNaC::ptr< T >::operator== (
            const ptr< U > & rhs ) const  [inline], [noexcept]
```

6.141.3.7 operator"!=()

```
template<class T>
template<class U >
bool GiNaC::ptr< T >::operator!= (
            const ptr< U > & rhs ) const  [inline], [noexcept]
```

6.141.4 Friends And Related Function 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< GiNaC::basic >::operator!=(), and GiNaC::ptr< GiNaC::basic >::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< GiNaC::basic >::makewritable(), GiNaC::ptr< GiNaC::basic >::operator!=(), GiNaC::ptr< GiNaC::basic >::operator*(), GiNaC::ptr< GiNaC::basic >::operator->(), GiNaC::ptr< GiNaC::basic >::operator=(), GiNaC::ptr< GiNaC::basic >::operator==(), GiNaC::ptr< GiNaC::basic >::ptr(), GiNaC::ptr< GiNaC::basic >::swap(), and GiNaC::ptr< GiNaC::basic >::~ptr().

The documentation for this class was generated from the following file:

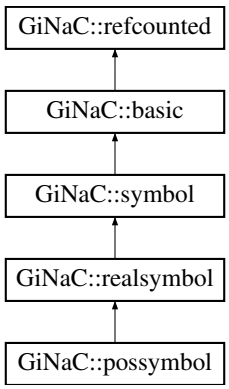
- ptr.h

6.142 GiNaC::realsymbol Class Reference

Specialization of symbol to real domain.

```
#include <symbol.h>
```

Inheritance diagram for GiNaC::realsymbol:



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.

Additional Inherited Members

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.

Reimplemented in GiNaC::possymbol.

References GiNaC::domain::real.

6.142.3.2 conjugate()

```
ex GiNaC::realsymbol::conjugate ( ) const  [inline], [override], [virtual]
```

Reimplemented from GiNaC::symbol.

6.142.3.3 real_part()

```
ex GiNaC::realsymbol::real_part ( ) const  [inline], [override], [virtual]
```

Reimplemented from GiNaC::symbol.

6.142.3.4 imag_part()

```
ex GiNaC::realsymbol::imag_part ( ) const  [inline], [override], [virtual]
```

Reimplemented from GiNaC::symbol.

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.

Reimplemented in GiNaC::possymbol.

References GiNaC::status_flags::dynallocated, realsymbol(), and GiNaC::basic::setflag().

The documentation for this class was generated from the following files:

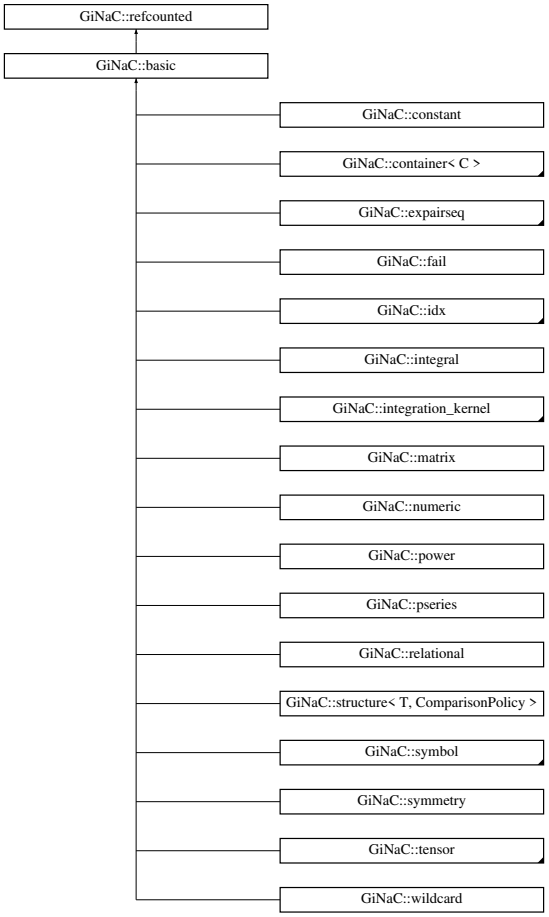
- symbol.h
- symbol.cpp

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_functor > & 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_functor &f)
- void set_print_func (unsigned id, const print_functor &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_functor > 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_functor>& 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().

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:

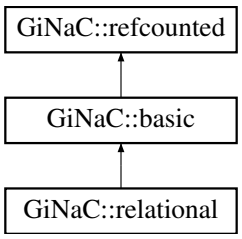
- registrar.h

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:



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 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

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 Attributes

- ex lh
- ex rh
- operators o

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

equal	
not_equal	
less	
less_or_equal	
greater	
greater_or_equal	

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 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.

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 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.

6.145.5.10 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.

6.145.5.11 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.

6.145.5.12 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.

6.145.5.13 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().

6.145.5.14 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.

6.145.5.15 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.

6.145.5.16 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().

6.145.5.17 rhs()

```
ex GiNaC::relational::rhs ( ) const  [inline]
```

References rh.

Referenced by GiNaC::atan_series(), GiNaC::atanh_series(), GiNaC::Li2_series(), and GiNaC::log_series().

6.145.5.18 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!().

6.145.5.19 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) == false does not imply (a>=b) == 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::is_zero(), less, less_or_equal, lh, make_safe_bool(), not_equal, o, and rh.

6.145.5.20 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(), 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(), 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(), 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 undefinitely.
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:

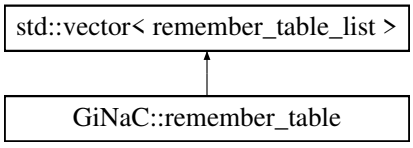
- flags.h

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:



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_ASSERT, and table_size.

6.147.3.2 add_entry()

```
void GiNaC::remember_table::add_entry (
             function const & f,
             ex const & result )
```

References 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_table().

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(), 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>`

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_ASSERT, hashvalue, last_access, seq, and successful_hits.

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]
```

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 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:

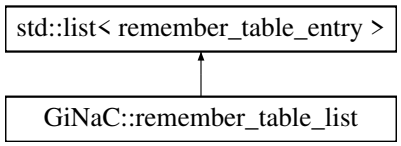
- remember.h
- remember.cpp

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:



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_lfu, GiNaC::remember_strategies::delete_lru, 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==(), and GiNaC::basic::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

- spmap 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 spm.

Referenced by GiNaC::simplify_indexed_product().

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 spm.

Referenced by GiNaC::simplify_indexed_product().

6.153.2.7 debugprint()

```
void GiNaC::scalar_products::debugprint ( ) const
```

References k, and spm.

6.153.3 Member Data Documentation

6.153.3.1 spm

```
spmap GiNaC::scalar_products::spm  [protected]
```

Referenced by 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

suppress_branchcut	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	Gauss elimination. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = m_{i,j}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)} / m_{k,k}^{(k)}$$ This algorithm is well-suited for numerical matrices but generally suffers from the expensive division (and computation of GCDs) at each step.

Enumerator

divfree	Division-free elimination. This is a modification of Gauss elimination where the division by the pivot element is not carried out. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)}$$ This algorithm is only there for the purpose of cross-checks. It suffers from exponential intermediate expression swell. Use it only for small systems.
bareiss	Bareiss fraction-free elimination. This is a modification of Gauss elimination where the division by the pivot element is *delayed* until it can be carried out without computing GCDs. If $m_{i,j}^{(0)}$ are the entries of the original matrix, then the matrix is transformed into triangular form by applying the rules $$m_{i,j}^{(k+1)} = (m_{i,j}^{(k)} m_{k,k}^{(k)} - m_{i,k}^{(k)} m_{k,j}^{(k)})/m_{k-1,k-1}^{(k-1)}$$ (We have set $m_{-1,-1}^{(-1)} = 1$ 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.
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:

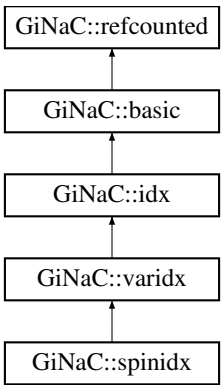
- flags.h

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:



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.

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 Attributes

- bool dotted

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

v	Value of index (numeric or symbolic)
dim	Dimension of index space (numeric or symbolic)
covariant	Make covariant index (default is contravariant)
dotted	Make covariant dotted (default is undotted)

Returns

newly constructed index

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::varidx.

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::varidx.

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::varidx.

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::varidx.

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_calculated.

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:

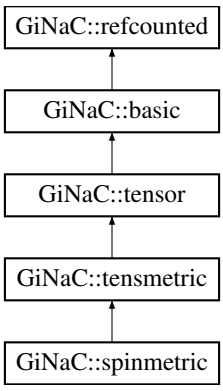
- idx.h
- idx.cpp

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:



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 metric tensor.
- bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const override

  Contraction of an indexed spinor metric with something else.

Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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::tensmetric.

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::tensmetric.

References GiNaC::_ex0, GiNaC::_ex1, GiNaC::_ex_1, GiNaC::idx::get_value(), GINAC_ASSERT, GiNaC::basic::hold(), GiNaC::info_flags::nonnegint, GiNaC::basic::nops(), GiNaC::basic::op(), and GiNaC::to_int().

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::tensmetric.

References GiNaC::_ex1, GiNaC::_ex2, GiNaC::_ex_2, GiNaC::delta_tensor(), GINAC_ASSERT, GiNaC::is_dummy_pair(), GiNaC::idx::is_symbolic(), and GiNaC::idx::subs().

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>`

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.

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

dynallocated	heap-allocated (i.e. created by new if we want to be clever and bypass the stack, **See also** ex::construct_from_basic())
evaluated	.eval() has already done its job
expanded	.expand(0) has already done its job (other expand() options ignore this flag)
hash_calculated	.calchash() has already done its job
not_shareable	don't share instances of this object between different expressions unless explicitly asked to (used by ex::compare())

The documentation for this class was generated from the following file:

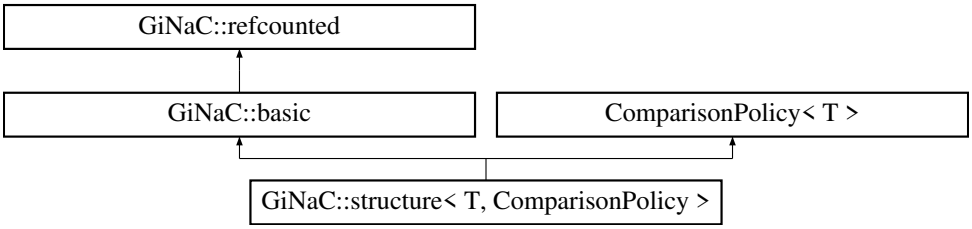
- flags.h

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 >:



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

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.

Static Private Member Functions

- static const char * get_class_name ()

Private Attributes

- T obj

Additional Inherited Members

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.

References GiNaC::evalm().

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

c	print context object that describes the output formatting
level	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.

References GiNaC::op().

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 GiNaC::has(), and 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.

References GiNaC::match().

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, options, and GiNaC::subs().

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.

References GiNaC::degree().

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.

References GiNaC::ldegree().

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 GiNaC::coeff(), and 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 GiNaC::expand(), and 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

s	object(s) to sort in
distributed	recursive or distributed form (only used when s is a list)

Reimplemented from GiNaC::basic.

References GiNaC::collect().

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, r, and GiNaC::series().

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.

References GiNaC::normal().

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.

References GiNaC::to_rational().

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.

References GiNaC::to_polynomial().

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

self	First indexed expression; its base object is ∗this
other	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

self	Indexed expression; its base object is ∗this
other	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

self	Pointer to first indexed expression; its base object is ∗this
other	Pointer to second indexed expression
v	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.

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 >::is_equal_same_type(), and GiNaC::structure< T, ComparisonPolicy >::operator->().

The documentation for this class was generated from the following file:

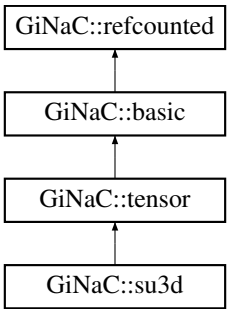
- structure.h

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:



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.

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

Additional Inherited Members

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(), std::swap(), and GiNaC::to_int().

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:

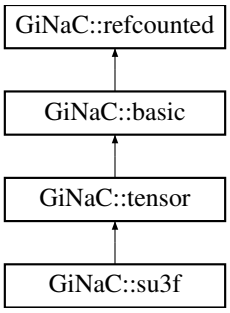
- color.h
- color.cpp

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:



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.

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

Additional Inherited Members

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(), std::swap(), and GiNaC::to_int().

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::I, 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:

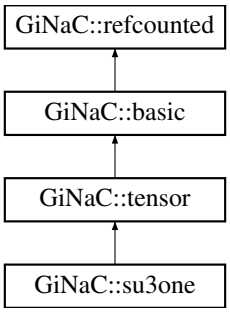
- color.h
- color.cpp

6.163 GiNaC::su3one Class Reference

This class represents the su(3) unity element.

`#include <color.h>`

Inheritance diagram for GiNaC::su3one:



Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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:

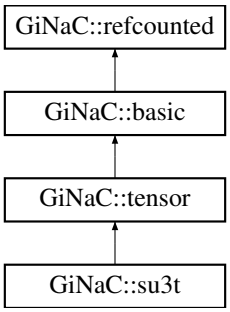
- color.h

6.164 GiNaC::su3t Class Reference

This class represents an su(3) generator.

```
#include <color.h>
```

Inheritance diagram for GiNaC::su3t:



Public Member Functions

- bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const override

  Contraction of generator with something else.

Protected Member Functions

- void do_print (const print_context &c, unsigned level) const
- void do_print_latex (const print_latex &c, unsigned level) const

Additional Inherited Members

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 expairseq::subschildren()
pattern_is_not_product	used internally by expairseq::subschildren()
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".

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:

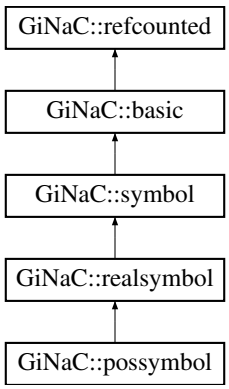
- normal.cpp

6.169 GiNaC::symbol Class Reference

Basic CAS symbol.

`#include <symbol.h>`

Inheritance diagram for GiNaC::symbol:



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

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 Attributes

- unsigned serial

  unique serial number for comparison
- std::string name

  printname of this symbol
- std::string TeX_name

  LaTeX name of this symbol.

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, and r.

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.

Reimplemented in GiNaC::realsymbol.

6.169.3.10 real_part()

```
ex GiNaC::symbol::real_part ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

Reimplemented in GiNaC::realsymbol.

6.169.3.11 imag_part()

```
ex GiNaC::symbol::imag_part ( ) const  [override], [virtual]
```

Reimplemented from GiNaC::basic.

Reimplemented in GiNaC::realsymbol.

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 & sym_lst )  [override], [virtual]
```

Read (a.k.a.

Read object from archive_node.

deserialize) object from archive.

Reimplemented from GiNaC::basic.

References 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::possymbol, and GiNaC::realsymbol.

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_Te↩X_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 s.

Referenced by 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:

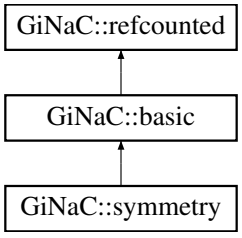
- inifcns.cpp

6.171 GiNaC::symmetry Class Reference

This class describes the symmetry of a group of indices.

```
#include <symmetry.h>
```

Inheritance diagram for GiNaC::symmetry:



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.

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

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

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 & sym_lst )  [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_cycl(), GiNaC::sy_none(), 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.

Referenced by GiNaC::indexed::read_archive(), and GiNaC::indexed::validate().

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 Function 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

v	Start of expression vector
symm	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.

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:

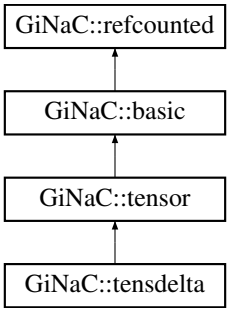
- indexed.cpp

6.175 GiNaC::tensdelta Class Reference

This class represents the delta tensor.

```
#include <tensor.h>
```

Inheritance diagram for GiNaC::tensdelta:



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.

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

Additional Inherited Members

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::to_int().

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::tensor.

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:

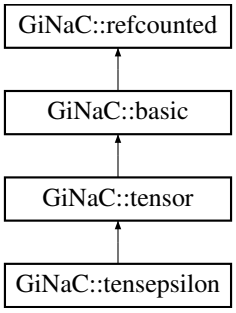
- tensor.h
- tensor.cpp

6.176 GiNaC::tensepsilon Class Reference

This class represents the totally antisymmetric epsilon tensor.

`#include <tensor.h>`

Inheritance diagram for GiNaC::tensepsilon:



Public Member Functions

- tensepsilon (bool minkowski, bool pos_sig)
- bool info (unsigned inf) 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.

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

Private Attributes

- bool minkowski

  If true, tensor is in Minkowski-type space.
- bool pos_sig

  If true, the metric is assumed to be diag(-1,1,1...).

Additional Inherited Members

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, GiNaC::to_int(), 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_ASSERT, GiNaC::lorentz_g(), GiNaC::metric_tensor(), minkowski, and pos_sig.

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::tensor.

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 diag(-1,1,1...).

Otherwise it is diag(1,-1,-1,...). 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:

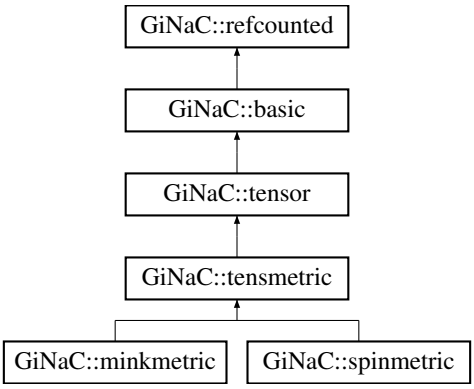
- tensor.h
- tensor.cpp

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:



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 metric tensor.
- bool contract_with (exvector::iterator self, exvector::iterator other, exvector &v) const override

  Contraction of an indexed metric tensor with something else.

Protected Member Functions

- unsigned return_type () const override
- void do_print (const print_context &c, unsigned level) const

Additional Inherited Members

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.

Reimplemented in GiNaC::spinmetric, and GiNaC::minkmetric.

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.

Reimplemented in GiNaC::spinmetric, and GiNaC::minkmetric.

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.

Reimplemented in GiNaC::spinmetric.

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::tensor.

Reimplemented in GiNaC::minkmetric.

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:

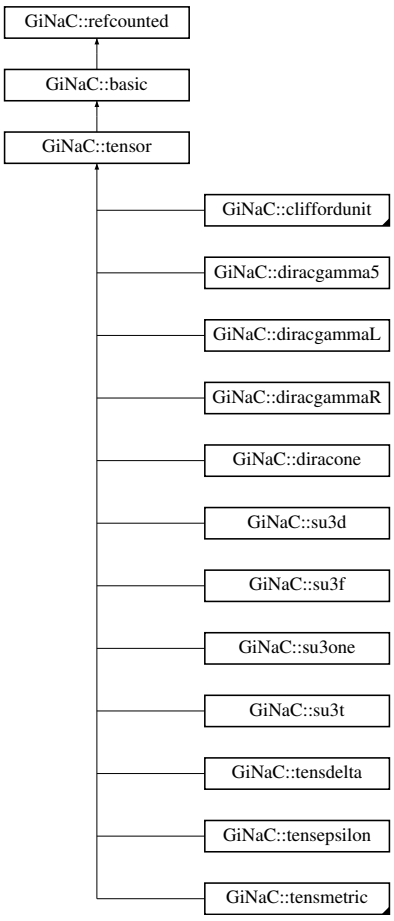
- tensor.h
- tensor.cpp

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:



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.

Protected Member Functions

- unsigned return_type () const override

Additional Inherited Members

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.

Reimplemented in GiNaC::tensepsilon, GiNaC::minkmetric, GiNaC::tensmetric, and GiNaC::tensdelta.

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.

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

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:

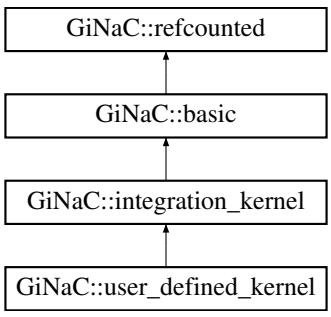
- archive.h
- archive.cpp

6.183 GiNaC::user_defined_kernel Class Reference

A user-defined integration kernel.

```
#include <integration_kernel.h>
```

Inheritance diagram for GiNaC::user_defined_kernel:



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.

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 Attributes

- ex f
- ex x

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:

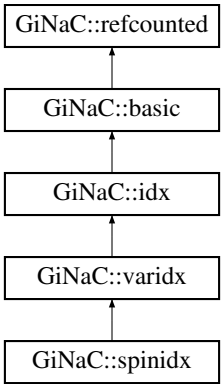
- integration_kernel.h
- integration_kernel.cpp

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:



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.

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 Attributes

- bool covariant

  x.mu, default is contravariant: x∼mu

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

v	Value of index (numeric or symbolic)
dim	Dimension of index space (numeric or symbolic)
covariant	Make covariant index (default is contravariant)

Returns

newly constructed index

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.

Reimplemented in GiNaC::spinidx.

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.

Reimplemented in GiNaC::spinidx.

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.

Reimplemented in GiNaC::spinidx.

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.

Reimplemented in GiNaC::spinidx.

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∼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:

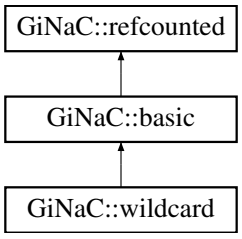
- basic.h

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:



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 &syms) override
  Read (a.k.a.
- unsigned get_label () const

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

Private Attributes

- unsigned label

  Label used to distinguish different wildcards.

Additional Inherited Members

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(zeta1_evalf).
                                eval_func(zeta1_eval).
                                derivative_func(zeta1_deriv).
                                print_func<print_latex>(zeta1_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

- 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

- 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

- 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

- 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::symmetry::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::canonicalize(), GiNaC::clifford::clifford(), GiNaC::pseries::coeff(), GiNaC::add::combine_ex_with_coeff_to_pair(), GiNaC::mul::combine_ex_with_coeff_to_pair(), GiNaC::mul::combine_overall_coeff(), GiNaC::add::combine_pair_with_coeff_to_pair(), GiNaC::mul::combine_pair_with_coeff_to_pair(), GiNaC::function::conjugate(), GiNaC::ex::construct_from_basic(), GiNaC::matrix::contract_with(), GiNaC::tensdelta::contract_with(), GiNaC::tensmetric::contract_with(), GiNaC::su3t::contract_with(), GiNaC::su3f::contract_with(), GiNaC::su3d::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::mul::eval(), GiNaC::power::eval(), GiNaC::ncmul::eval(), GiNaC::indexed::eval(), GiNaC::function::eval(), GiNaC::matrix::eval_indexed(), GiNaC::tensdelta::eval_indexed(), GiNaC::tensmetric::eval_indexed(), GiNaC::minkmetric::eval_indexed(), GiNaC::su3f::eval_indexed(), GiNaC::su3d::eval_indexed(), GiNaC::spinmetric::eval_indexed(), GiNaC::tensepsilon::eval_indexed(), GiNaC::function::evalf(), GiNaC::ex::ex(), GiNaC::ex_to(), GiNaC::exp_deriv(), GiNaC::expair::expair(), GiNaC::indexed::expand(), GiNaC::function::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::symbol::is_equal_same_type(), GiNaC::fderivative::is_equal_same_type(), GiNaC::numeric::is_equal_same_type(), GiNaC::container< C >::is_equal_same_type(), GiNaC::structure< T, ComparisonPolicy >::is_equal_same_type(), GiNaC::function::is_equal_same_type(), GiNaC::matrix::let_op(), GiNaC::clifford::let_op(), GiNaC::container< C >::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::lsolve(), GiNaC::ex::makewriteable(), GiNaC::matrix::markowitz_elimination(), GiNaC::color::match_same_type(), GiNaC::clifford::match_same_type(), GiNaC::idx::match_same_type(), GiNaC::relational::match_same_type(), GiNaC::matrix::match_same_type(), GiNaC::fderivative::match_same_type(),

GiNaC::varidx::match_same_type(), GiNaC::spinidx::match_same_type(), GiNaC::function::match_same_type(), GiNaC::add::max_coefficient(), GiNaC::mul::max_coefficient(), GiNaC::mul::mul(), GiNaC::add::normal(), GiNaC::ex::normal(), GiNaC::ex::numer(), GiNaC::ex::numer_denom(), GiNaC::integral::op(), GiNaC::idx::op(), GiNaC::matrix::op(), GiNaC::power::op(), GiNaC::relational::op(), GiNaC::clifford::op(), GiNaC::container< C >::op(), GiNaC::sy_is_less::operator()(), GiNaC::sy_swap::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< GiNaC::basic >::ptr(), GiNaC::matrix::rank(), GiNaC::function::real_part(), GiNaC::rename_dummy_indices(), GiNaC::relational::return_type(), GiNaC::ncmul::return_type(), GiNaC::function::return_type(), GiNaC::relational::return_type_tinfo(), GiNaC::function::return_type_tinfo(), GiNaC::S_deriv(), GiNaC::matrix::scalar_mul_indexed(), GiNaC::symbol::series(), GiNaC::pseries::series(), GiNaC::mul::series(), GiNaC::function::series(), GiNaC::simplify_indexed_product(), GiNaC::sin_deriv(), GiNaC::sinh_deriv(), GiNaC::add::smod(), GiNaC::mul::smod(), 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

- 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 <cstddef>
#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

- 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

- 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

- 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~mu -> (1, mu) and a\ -> (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~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 &rll, 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)
- static ex GiNaC::get_clifford_comp (const ex &e, const ex &c, bool root=true)

  Auxiliary structure to define a function for striping 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 anticommutates 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

- 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∼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 &rll, 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)
- 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

- 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_RE↩ GISTERED_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+I∗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 &rll)

  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]; \
    }
```

Referenced by GiNaC::permute_free_index_to_front().

7.11.2.2 CMPINDICES

```
#define CMPINDICES(
              A,
              B,
              C ) ((v[0] == (A)) && (v[1] == (B)) && (v[2] == (C)))
```

Referenced by GiNaC::su3f::eval_indexed(), and GiNaC::su3d::eval_indexed().

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

- 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+I∗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 &rll)

  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(), and GiNaC::mul::eval().

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

- 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)) const ant
- GiNaC::GINAC_BIND_UNARCHIVER (constant)

Variables

- const constant GiNaC::Pi ("Pi", PiEvalf, "\i", domain::positive)

  Pi.
- const constant GiNaC::Euler ("Euler", EulerEvalf, "\amma_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

- 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

- 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

- 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

- 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

- GiNaC
- GiNaC::internal
- 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

- 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

- 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

- 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

- 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>
```

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

- 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

- 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

- 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 <algorithm>
#include <limits>
#include <list>
#include <vector>
#include <stack>
#include <cln/cln.h>
```

Namespaces

- 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::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(), GiNa↩ C::symbol::series(), GiNaC::pseries::series(), GiNaC::add::series(), GiNaC::integration_kernel::series(), GiNaC↩ ::mul::series(), GiNaC::fderivative::series(), GiNaC::integral::series(), GiNaC::power::series(), GiNaC::structure< T, ComparisonPolicy >::series(), GiNaC::ex::series(), GiNaC::basic::series(), GiNaC::Eisenstein_kernel::series(), GiNaC::modular_form_kernel::series(), GiNaC::function::series(), GiNaC::series(), GiNaC::refcounted::set_↩ refcount(), GiNaC::simplify_indexed_product(), GiNaC::matrix::solve(), GiNaC::sprem(), GiNaC::sr_gcd(), Gi↩ NaC::numeric::step(), GiNaC::sub_matrix(), GiNaC::matrix::subs(), GiNaC::ex::subs(), 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::add::combine_ex_with_coeff_to_pair(), GiNaC::mul::combine_ex_with_coeff_to_pair(), GiNaC::mul::combine_overall_coeff(), GiNaC::add::combine_pair_with_coeff_to_pair(), GiNaC::mul::combine_pair_with_coeff_to_pair(), 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::wildcard::do_print(), GiNaC::constant::do_print(), GiNaC::integral::do_print(), GiNaC::relational::do_print(), GiNaC::fderivative::do_print(), GiNaC::symbol::do_print(), GiNaC::ncmul::do_print(), GiNaC::add::do_print(), GiNaC::integration_kernel::do_print(), GiNaC::symmetry::do_print(), GiNaC::mul::do_print(), GiNaC::idx::do_print(), GiNaC::pseries::do_print(), GiNaC::matrix::do_print(), GiNaC::basic_log_kernel::do_print(), GiNaC::varidx::do_print(), GiNaC::multiple_polylog_kernel::do_print(), GiNaC::numeric::do_print(), GiNaC::indexed::do_print(), GiNaC::spinidx::do_print(), GiNaC::ELi_kernel::do_print(), GiNaC::container< C >::do_print(), GiNaC::Ebar_kernel::do_print(), GiNaC::basic::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::fderivative::do_print_csrc(), GiNaC::ncmul::do_print_csrc(), GiNaC::add::do_print_csrc(), GiNaC::power::do_print_csrc(), GiNaC::idx::do_print_csrc(), GiNaC::mul::do_print_csrc(), GiNaC::numeric::do_print_csrc(), GiNaC::power::do_print_csrc_cl_N(), GiNaC::numeric::do_print_csrc_cl_N(), GiNaC::clifford::do_print_dflt(), GiNaC::power::do_print_dflt(), GiNaC::constant::do_print_latex(), GiNaC::integral::do_print_latex(), GiNaC::fderivative::do_print_latex(), GiNaC::clifford::do_print_latex(), GiNaC::symbol::do_print_latex(), GiNaC::add::do_print_latex(), GiNaC::power::do_print_latex(), GiNaC::mul::do_print_latex(), GiNaC::idx::do_print_latex(), GiNaC::pseries::do_print_latex(), GiNaC::matrix::do_print_latex(), GiNaC::numeric::do_print_latex(), GiNaC::indexed::do_print_latex(), GiNaC::spinidx::do_print_latex(), GiNaC::power::do_print_python(), GiNaC::pseries::do_print_python(), GiNaC::container< C >::do_print_python(), GiNaC::wildcard::do_print_python_repr(), GiNaC::constant::do_print_python_repr(), GiNaC::relational::do_print_python_repr(), GiNaC::symbol::do_print_python_repr(), GiNaC::add::do_print_python_repr(), GiNaC::power::do_print_python_repr(), GiNaC::mul::do_print_python_repr(), GiNaC::matrix::do_print_python_repr(), GiNaC::pseries::do_print_python_repr(), GiNaC::numeric::do_print_python_repr(), GiNaC::container< C >::do_print_python_repr(), GiNaC::basic::do_print_python_repr(), GiNaC::wildcard::do_print_tree(), GiNaC::constant::do_print_tree(), GiNaC::clifford::do_print_tree(), GiNaC::fderivative::do_print_tree(), GiNaC::symbol::do_print_tree(), GiNaC::symmetry::do_print_tree(), GiNaC::idx::do_print_tree(), GiNaC::pseries::do_print_tree(), GiNaC::varidx::do_print_tree(), GiNaC::numeric::do_print_tree(), GiNaC::indexed::do_print_tree(), GiNaC::spinidx::do_print_tree(), GiNaC::container< C >::do_print_tree(), GiNaC::basic::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::lsolve(), GiNaC::matrix::markowitz_elimination(), GiNaC::matrix::matrix(), GiNaC::matrix::mul(), GiNaC::matrix::mul_scalar(), GiNaC::error_and_integral_is_less::operator()(), GiNaC::print_ptrfun_handler< T, C >::operator()(), GiNaC::print_memfun_handler< T, C >::operator()(), GiNaC::print_functor::operator()(), GiNaC::matrix::pivot(), GiNaC::pseries::power_const(), GiNaC::ex::primpart(), GiNaC::fderivative::print(), GiNaC::expair::print(), GiNaC::ex::print(), GiNaC::structure< T, ComparisonPolicy >::print(), GiNaC::basic::print(), GiNaC::function::print(), GiNaC::add::print_add(), 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::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::matrix::subs(), GiNaC::clifford::subs(), 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::power::expand(), GiNaC::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::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::reduced_matrix(), GiNaC::reposition_dummy_indices(), GiNaC::resultant(), GiNaC::rows(), GiNaC::S_eval(), GiNaC::smod(), GiNaC::sub_matrix(), GiNaC::symbol::subs(), GiNaC::idx::subs(), GiNaC::pseries::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::numeric::subs(), GiNaC::container< C >::subs(), GiNaC::ex::subs(), GiNaC::structure< T, ComparisonPolicy >::subs(), GiNaC::basic::subs(), GiNaC::subs(), GiNaC::basic::subs_one_level(), GiNaC::container< C >::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::ex::subs(), and GiNaC::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::wildcard::archive(), GiNaC::color::archive(), GiNaC::clifford::archive(), GiNaC::constant::archive(), GiNaC::idx::archive(), GiNaC::integral::archive(), GiNaC::symbol::archive(), GiNaC::relational::archive(), GiNaC::pseries::archive(), GiNaC::matrix::archive(), GiNaC::fderivative::archive(), GiNaC::symmetry::archive(), GiNaC::power::archive(), GiNaC::minkmetric::archive(), GiNaC::numeric::archive(), GiNaC::varidx::archive(), GiNaC::container< C >::archive(), GiNaC::indexed::archive(), GiNaC::tensepsilon::archive(), GiNaC::spinidx::archive(), GiNaC::basic::archive(), GiNaC::archive::archive(), GiNaC::function::archive(), GiNaC::bernoulli(), GiNaC::binomial(), GiNaC::pseries::coeff(), GiNaC::add::coeff(), GiNaC::mul::coeff(), GiNaC::power::coeff(), GiNaC::ncmul::coeff(), GiNaC::numeric::coeff(), GiNaC::structure< T, ComparisonPolicy >::coeff(), GiNaC::ex::coeff(), GiNaC::basic::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::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::pseries::normal(), GiNaC::add::normal(), GiNaC::mul::normal(), GiNaC::const_iterator::operator+(), GiNaC::const_iterator::operator+=(), GiNaC::const_iterator::operator-(), GiNaC::const_iterator::operator-=(), GiNaC::operator<<(), GiNaC::operator>>(), GiNaC::const_iterator::operator[ ](), GiNaC::primitive_dirichlet_character(), GiNaC::psi2_deriv(), GiNaC::psi2_eval(), GiNaC::psi2_evalf(), GiNaC::psi2_series(), GiNaC::wildcard::read_archive(), GiNaC::color::read_archive(), GiNaC::clifford::read_archive(), GiNaC::constant::read_archive(), GiNaC::idx::read_archive(), GiNaC::integral::read_archive(), GiNaC::symbol::read_archive(), GiNaC::relational::read_archive(), GiNaC::pseries::read_archive(), GiNaC::fderivative::read_archive(), GiNaC::matrix::read_archive(), GiNaC::symmetry::read_archive(), GiNaC::power::read_archive(), GiNaC::minkmetric::read_archive(), GiNaC::numeric::read_archive(), GiNaC::container< C >::read_archive(), GiNaC::varidx::read_archive(), GiNaC::indexed::read_archive(), GiNaC::tensepsilon::read_archive(), GiNaC::spinidx::read_archive(), GiNaC::function::read_archive(), GiNaC::power::real_part(), 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::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::canonicalize(), GiNaC::cyclic_permutation(), GiNaC::mul::eval(), GiNaC::power::expand_add_2(), GiNaC::mul::expandchildren(), GiNaC::find_free_and_dummy(), GiNaC::permutation_sign(), GiNaC::shaker_sort(), 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::mpartition2::next_partition(), GiNaC::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::reposition_dummy_indices(), GiNaC::resultant(), GiNaC::ELi_kernel::series_coeff_impl(), GiNaC::Ebar_kernel::series_coeff_impl(), GiNaC::simplify_indexed(), and GiNaC::sqrfree_parfrac().

7.28.3.13 poly

```
upoly poly
```

Referenced by GiNaC::matrix::charpoly(), and GiNaC::factor().

7.28.3.14 x

```
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::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::atan_conjugate(), GiNaC::atan_deriv(), GiNaC::atan_eval(), GiNaC::atan_evalf(), GiNaC::atanh(), GiNaC::atanh_conjugate(), GiNaC::atanh_deriv(), GiNaC::atanh_eval(), GiNaC::atanh_evalf(), 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::matrix::conjugate(), GiNaC::mul::conjugate(), GiNaC::container< C >::conjugate(), 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_power(), GiNaC::exp_real_part(), GiNaC::factorial_conjugate(), GiNaC::factorial_eval(), GiNaC::factorial_evalf(), GiNaC::factorial_print_dflt_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::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 >(), GiNa↩
C::is_zero(), GiNaC::isqrt(), 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::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_↩
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_real_part(), GiNa↩
C::Order_series(), GiNaC::pow(), GiNaC::prem(), 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(), GiNa↩
C::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(), Gi↩
NaC::sinh_real_part(), GiNaC::sprem(), GiNaC::sqrfree(), GiNaC::sqrfree_parfrac(), GiNaC::sqrfree_yun(), GiNa↩
C::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_conjugate(), GiNaC↩
::tgamma_deriv(), GiNaC::tgamma_eval(), GiNaC::tgamma_evalf(), GiNaC::to_double(), GiNaC::to_int(), GiNaC↩
::to_long(), 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

```
exset syms
```

Referenced by GiNaC::lsolve().

7.28.3.18 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::integral::expand(), GiNaC::pseries::expand(), GiNaC::ncmul::expand(), GiNaC::add::expand(), GiNaC::power::expand(), GiNaC::mul::expand(), GiNaC::indexed::expand(), GiNaC::structure< T, ComparisonPolicy >::expand(), GiNaC::ex::expand(), GiNaC::basic::expand(), GiNaC::function::expand(), GiNaC::expand(), GiNaC::power::expand_add(), GiNaC::power::expand_add_2(), GiNaC::power::expand_mul(), GiNaC::ncmul::expandchildren(), GiNaC::mul::expandchildren(), GiNaC::factor(), GiNaC::gcd(), GiNaC::mul::has(), GiNaC::power::has(), GiNaC::ex::has(), GiNaC::structure< T, ComparisonPolicy >::has(), GiNaC::basic::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::psi1_series(), GiNaC::psi2_series(), GiNaC::S_series(), GiNaC::pseries::series(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::integral::series(), GiNaC::fderivative::series(), GiNaC::power::series(), GiNaC::structure< T, ComparisonPolicy >::series(), GiNaC::ex::series(), GiNaC::function::series(), GiNaC::series(), GiNaC::set_print_context(), GiNaC::set_print_func(), GiNaC::set_print_options(), GiNaC::simplify_indexed(), GiNaC::step_series(), GiNaC::symbol::subs(), GiNaC::idx::subs(), GiNaC::matrix::subs(), GiNaC::pseries::subs(), GiNaC::relational::subs(), GiNaC::power::subs(), GiNaC::clifford::subs(), GiNaC::numeric::subs(), GiNaC::container< C >::subs(), GiNaC::ex::subs(), GiNaC::structure< T, ComparisonPolicy >::subs(), GiNaC::basic::subs(), GiNaC::subs(), GiNaC::basic::subs_one_level(), GiNaC::container< C >::subschildren(), GiNaC::tan_series(), GiNaC::tanh_series(), and GiNaC::tgamma_series().

7.29 factor.h File Reference

Polynomial factorization.

Namespaces

- 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

- 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

- 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

- 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

- 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

- 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

- 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...

Namespaces

- 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 &)
- typedef ex(∗ GiNaC::series_funcp_2) (const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_2) (const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_2) (const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::real_part_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_3) (const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_3) (const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_3) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::series_funcp_3) (const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_3) (const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_3) (const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::real_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_4) (const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_4) (const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_4) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::series_funcp_4) (const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_4) (const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_4) (const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::real_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)

- typedef ex(∗ GiNaC::series_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_5) (const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::real_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::series_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_6) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::real_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::series_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_7) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)

- typedef ex(∗ GiNaC::real_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::imag_part_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::expand_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- 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 &)
- typedef ex(∗ GiNaC::power_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- 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)
- 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 &)
- typedef bool(∗ GiNaC::info_funcp_8) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- typedef ex(∗ GiNaC::eval_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::evalf_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- typedef ex(∗ GiNaC::conjugate_funcp_9) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &)
- 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 &)
- 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 &)
- 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)
- 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)
- 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 &)
- 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 &)
- 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)
- 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 &)
- 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)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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)
- 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)

- 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 &)
- 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 &)
- 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 relational &, int, unsigned)
- 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 print_context &)
- typedef bool(∗ GiNaC::info_funcp_10) (const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, const ex &, unsigned)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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 &, unsigned)
- 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 &, unsigned)
- 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 symbol &)
- 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 &)
- 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 relational &, int, unsigned)
- 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 print_context &)
- 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 &, unsigned)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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 &)
- 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 &, unsigned)
- 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 &, unsigned)
- 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 &)
- 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 &)
- 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 relational &, int, unsigned)
- 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 print_context &)

- 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)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &, unsigned)
- 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 &, const ex &, unsigned)
- 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 &)
- 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 &, const ex &)
- 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 ex &, const relational &, int, unsigned)
- 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 ex &, const print_context &)
- 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 &, const ex &, unsigned)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &, const ex &)
- 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 &)
- 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)
- 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)
- 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 &)
- 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 &)
- 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)
- 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 &)
- 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)
- typedef ex(∗ GiNaC::eval_funcp_exvector) (const exvector &)

- typedef ex(∗ GiNaC::evalf_funcp_exvector) (const exvector &)
- typedef ex(∗ GiNaC::conjugate_funcp_exvector) (const exvector &)
- typedef ex(∗ GiNaC::real_part_funcp_exvector) (const exvector &)
- typedef ex(∗ GiNaC::imag_part_funcp_exvector) (const exvector &)
- typedef ex(∗ GiNaC::expand_funcp_exvector) (const exvector &, unsigned)
- typedef ex(∗ GiNaC::derivative_funcp_exvector) (const exvector &, unsigned)
- typedef ex(∗ GiNaC::expl_derivative_funcp_exvector) (const exvector &, const symbol &)
- typedef ex(∗ GiNaC::power_funcp_exvector) (const exvector &, const ex &)
- typedef ex(∗ GiNaC::series_funcp_exvector) (const exvector &, const relational &, int, unsigned)
- typedef void(∗ GiNaC::print_funcp_exvector) (const exvector &, const print_context &)
- typedef bool(∗ GiNaC::info_funcp_exvector) (const exvector &, unsigned)

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

- 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

- 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

- 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

- 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

- 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 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 dimensionality 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

- 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

- 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_float). 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("\ta"). 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("\athrm{Li}_2"))
- static ex GiNaC::Li3_eval (const ex &x)
- GiNaC::REGISTER_FUNCTION (Li3, eval_func(Li3_eval). latex_name("\athrm{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("\eta^\rime"))
- 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_expl_derivative (const ex &arg, const symbol &s)
- GiNaC::REGISTER_FUNCTION (Order, eval_func(Order_eval). series_func(Order_series). latex_↩ name("\athcal{O}"). expl_derivative_func(Order_expl_derivative). conjugate_func(Order_conjugate). real_↩ part_func(Order_real_part). imag_part_func(Order_imag_part))
- 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

- 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

- 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

- 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("\og \amma"))
- 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("\amma"))
- 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("\athrm{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(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

- 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 Li(n,x) multiple polylogarithm Li(lst{m_1,...,m_k},lst{x_1,...,x_k}) G(lst{a_1,...,a_k},y) or G(lst{a_1,...,a_k},lst{s_1,...,s_k},y) Nielsen's generalized polylogarithm S(n,p,x) harmonic polylogarithm H(m,x) or H(lst{m_1,...,m_k},x) multiple zeta value zeta(m) or zeta(lst{m_1,...,m_k}) alternating Euler sum zeta(m,s) or zeta(lst{m_1,...,m_k},lst{s_1,...,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.M.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 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

- 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)
- GiNaC::REGISTER_FUNCTION (exp, eval_func(exp_eval). evalf_func(exp_evalf). 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("\xp"))
- 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)
- GiNaC::REGISTER_FUNCTION (log, eval_func(log_eval). evalf_func(log_evalf). 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("\n"))
- 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)
- GiNaC::REGISTER_FUNCTION (sin, eval_func(sin_eval). evalf_func(sin_evalf). derivative_func(sin_deriv). real_part_func(sin_real_part). imag_part_func(sin_imag_part). conjugate_func(sin_conjugate). latex_↩ name("\in"))
- 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). 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("\os"))
- 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). 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("\an"))
- 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)
- GiNaC::REGISTER_FUNCTION (asin, eval_func(asin_eval). evalf_func(asin_evalf). derivative_func(asin_↩ deriv). conjugate_func(asin_conjugate). latex_name("\rcsin"))
- 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). derivative_↩ func(acos_deriv). conjugate_func(acos_conjugate). latex_name("\rccos"))
- 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)

- GiNaC::REGISTER_FUNCTION (atan, eval_func(atan_eval). evalf_func(atan_evalf). derivative_func(atan_deriv). series_func(atan_series). conjugate_func(atan_conjugate). latex_name("\rctan"))
- 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)
- GiNaC::REGISTER_FUNCTION (atan2, eval_func(atan2_eval). 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). derivative_func(sinh_deriv). real_part_func(sinh_real_part). imag_part_func(sinh_imag_part). conjugate_func(sinh_conjugate). latex_name("\inh"))
- 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). derivative_func(cosh_deriv). real_part_func(cosh_real_part). imag_part_func(cosh_imag_part). conjugate_func(cosh_conjugate). latex_name("\osh"))
- 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). 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("\anh"))
- 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). 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). 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). 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

- 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::subsvalue (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

- 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

- 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 ...
- 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::integration_kernel::Laurent_series(), GiNa↩C::Eisenstein_kernel::Laurent_series(), GiNaC::Eisenstein_h_kernel::Laurent_series(), GiNaC::modular_form_↩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::symbol::series(), GiNaC::pseries::series(), GiNaC::add::series(), GiNaC::mul::series(), GiNaC::integration_kernel::series(), GiNa↩C::integral::series(), GiNaC::fderivative::series(), GiNaC::power::series(), GiNaC::structure< T, ComparisonPolicy >::series(), GiNaC::ex::series(), GiNaC::basic::series(), GiNaC::Eisenstein_kernel::series(), GiNaC::Eisenstein_↩h_kernel::series(), GiNaC::modular_form_kernel::series(), GiNaC::function::series(), GiNaC::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::Bernoulli_polynomial(), GiNaC::generalised_Bernoulli_number(), GiNaC::integration_↩kernel::Laurent_series(), GiNaC::Eisenstein_kernel::Laurent_series(), GiNaC::Eisenstein_h_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 <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 τ (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

- 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 ...
- 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 lst.cpp File Reference

Implementation of GiNaC's lst.

```
#include "lst.h"
#include "archive.h"
```

Namespaces

- GiNaC

7.54.1 Detailed Description

Implementation of GiNaC's lst.

7.55 lst.h File Reference

Definition of GiNaC's lst.

```
#include "container.h"
#include <list>
```

Namespaces

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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 degree(n, x) < 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 e1,e2 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

- 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 degree(n, x) < 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

- 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_idecoded_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

- 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

- 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

- 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 (basis^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

- 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 (basis^exponent).

7.69 power.h File Reference

Interface to GiNaC's symbolic exponentiation (basis^exponent).

```
#include "basic.h"
#include "ex.h"
#include "archive.h"
```

Classes

- class GiNaC::power

  This class holds a two-component object, a basis and and exponent representing exponentiation.

Namespaces

- 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 (basis^exponent).

7.70 print.cpp File Reference

Implementation of helper classes for expression output.

```
#include "print.h"
#include <iostream>
```

Namespaces

- 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

- 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, supername)

  Macro for inclusion in the declaration of a print_context class.
- #define GINAC_IMPLEMENT_PRINT_CONTEXT(classname, supername)

  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,
              supername )
```

Value:

```
GINAC_DECLARE_PRINT_CONTEXT_COMMON(classname) \
    typedef supername 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,
              supername )
```

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, #supername,
      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

- 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

- 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

- GiNaC
- 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

- 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

- class GiNaC::container< C >

  Wrapper template for making GiNaC classes out of STL containers.
- 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

- 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, supername)

  Primary macro for inclusion in the declaration of each registered class.
- #define GINAC_DECLARE_REGISTERED_CLASS(classname, supername)

  Macro for inclusion in the declaration of each registered class.
- #define GINAC_IMPLEMENT_REGISTERED_CLASS(classname, supername) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supername, typeid(classname)));

  Macro for inclusion in the implementation of each registered class.
- #define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT(classname, supername, options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supername, typeid(classname)).options);

  Macro for inclusion in the implementation of each registered class.
- #define GINAC_IMPLEMENT_REGISTERED_CLASS_OPT_T(classname, supername, options) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC::registered_class_options(#classname, #supername, 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_REGISTER↩ ED_CLASS.

7.76.2.2 GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS

```
#define GINAC_DECLARE_REGISTERED_CLASS_NO_CTORS(
              classname,
              supername )
```

Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
    typedef supername 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,
              supername )
```

Value:

```
GINAC_DECLARE_REGISTERED_CLASS_COMMON(classname) \
    template<class B, typename... Args> friend B & dynallocate(Args &&... args); \
    typedef supername 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,
              supername ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNa
#supername, 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,
              supername,
              options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC:
#supername, 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,
              supername,
              options ) GiNaC::registered_class_info classname::reg_info = GiNaC::registered_class_info(GiNaC:
#supername, 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

- 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

- 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

- 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

- 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.
- class GiNaC::structure< T, ComparisonPolicy >

  Wrapper template for making GiNaC classes out of C++ structures.

Namespaces

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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!/(p1!*p2!*...*pk!) where n = p1+p2+...+pk, 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_120
- const numeric * GiNaC::_num_60_p
- const ex GiNaC::_ex_60 = _ex_60
- const numeric * GiNaC::_num_48_p
- const ex GiNaC::_ex_48 = _ex_48
- const numeric * GiNaC::_num_30_p
- const ex GiNaC::_ex_30 = _ex_30
- const numeric * GiNaC::_num_25_p
- const ex GiNaC::_ex_25 = _ex_25
- const numeric * GiNaC::_num_24_p
- const ex GiNaC::_ex_24 = _ex_24
- const numeric * GiNaC::_num_20_p
- const ex GiNaC::_ex_20 = _ex_20
- const numeric * GiNaC::_num_18_p
- const ex GiNaC::_ex_18 = _ex_18
- const numeric * GiNaC::_num_15_p
- const ex GiNaC::_ex_15 = _ex_15

- const numeric * GiNaC::_num_12_p
- const ex GiNaC::_ex_12 = _ex_12
- const numeric * GiNaC::_num_11_p
- const ex GiNaC::_ex_11 = _ex_11
- const numeric * GiNaC::_num_10_p
- const ex GiNaC::_ex_10 = _ex_10
- const numeric * GiNaC::_num_9_p
- const ex GiNaC::_ex_9 = _ex_9
- const numeric * GiNaC::_num_8_p
- const ex GiNaC::_ex_8 = _ex_8
- const numeric * GiNaC::_num_7_p
- const ex GiNaC::_ex_7 = _ex_7
- const numeric * GiNaC::_num_6_p
- const ex GiNaC::_ex_6 = _ex_6
- const numeric * GiNaC::_num_5_p
- const ex GiNaC::_ex_5 = _ex_5
- const numeric * GiNaC::_num_4_p
- const ex GiNaC::_ex_4 = _ex_4
- const numeric * GiNaC::_num_3_p
- const ex GiNaC::_ex_3 = _ex_3
- const numeric * GiNaC::_num_2_p
- const ex GiNaC::_ex_2 = _ex_2
- const numeric * GiNaC::_num_1_p
- const ex GiNaC::_ex_1 = _ex_1
- const numeric * GiNaC::_num_1_2_p
- const ex GiNaC::_ex_1_2 = _ex_1_2
- const numeric * GiNaC::_num_1_3_p
- const ex GiNaC::_ex_1_3 = _ex_1_3
- const numeric * GiNaC::_num_1_4_p
- const ex GiNaC::_ex_1_4 = _ex_1_4
- const numeric * GiNaC::_num0_p
- const ex GiNaC::_ex0 = _ex0
- const numeric * GiNaC::_num1_4_p
- const ex GiNaC::_ex1_4 = _ex1_4
- const numeric * GiNaC::_num1_3_p
- const ex GiNaC::_ex1_3 = _ex1_3
- const numeric * GiNaC::_num1_2_p
- const ex GiNaC::_ex1_2 = _ex1_2
- const numeric * GiNaC::_num1_p
- const ex GiNaC::_ex1 = _ex1
- const numeric * GiNaC::_num2_p
- const ex GiNaC::_ex2 = _ex2
- const numeric * GiNaC::_num3_p
- const ex GiNaC::_ex3 = _ex3
- const numeric * GiNaC::_num4_p
- const ex GiNaC::_ex4 = _ex4
- const numeric * GiNaC::_num5_p
- const ex GiNaC::_ex5 = _ex5
- const numeric * GiNaC::_num6_p
- const ex GiNaC::_ex6 = _ex6
- const numeric * GiNaC::_num7_p
- const ex GiNaC::_ex7 = _ex7
- const numeric * GiNaC::_num8_p
- const ex GiNaC::_ex8 = _ex8
- const numeric * GiNaC::_num9_p

- const ex GiNaC::_ex9 = _ex9
- const numeric * GiNaC::_num10_p
- const ex GiNaC::_ex10 = _ex10
- const numeric * GiNaC::_num11_p
- const ex GiNaC::_ex11 = _ex11
- const numeric * GiNaC::_num12_p
- const ex GiNaC::_ex12 = _ex12
- const numeric * GiNaC::_num15_p
- const ex GiNaC::_ex15 = _ex15
- const numeric * GiNaC::_num18_p
- const ex GiNaC::_ex18 = _ex18
- const numeric * GiNaC::_num20_p
- const ex GiNaC::_ex20 = _ex20
- const numeric * GiNaC::_num24_p
- const ex GiNaC::_ex24 = _ex24
- const numeric * GiNaC::_num25_p
- const ex GiNaC::_ex25 = _ex25
- const numeric * GiNaC::_num30_p
- const ex GiNaC::_ex30 = _ex30
- const numeric * GiNaC::_num48_p
- const ex GiNaC::_ex48 = _ex48
- const numeric * GiNaC::_num60_p
- const ex GiNaC::_ex60 = _ex60
- const numeric * GiNaC::_num120_p
- const ex GiNaC::_ex120 = _ex120

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

- 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!/(p1!*p2!*...*pk!) where n = p1+p2+...+pk, 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 $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N$$

  and

  $$i_j < i_{j+1}.$$

  It is assumed that $k > 0$ *and* $N - B \geq k$*.*
- class GiNaC::multi_iterator_ordered_eq< T >

  The class multi_iterator_ordered_eq defines a multi_iterator $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N$$

  and

  $$i_j \leq i_{j+1}.$$

  It is assumed that $k > 0$*.*
- class GiNaC::multi_iterator_ordered_eq_indv< T >

  The class multi_iterator_ordered_eq_indv defines a multi_iterator $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N_j$$

  and

  $$i_j \leq i_{j+1}.$$

  .

- class GiNaC::multi_iterator_counter< T >

  The class multi_iterator_counter defines a multi_iterator $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N$$

  .
- class GiNaC::multi_iterator_counter_indv< T >

  The class multi_iterator_counter_indv defines a multi_iterator $(i_1, i_2, ..., i_k)$*, such that*

  $$B \leq i_j < N_j$$

  .
- class GiNaC::multi_iterator_permutation< T >

  The class multi_iterator_permutation defines a multi_iterator $(i_1, i_2, ..., i_k)$*, 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.*
- 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

- 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

- GiNaC

Macros

- #define GINACLIB_MAJOR_VERSION 1
- #define GINACLIB_MINOR_VERSION 8
- #define GINACLIB_MICRO_VERSION 0
- #define GINAC_LT_CURRENT 11
- #define GINAC_LT_REVISION 0
- #define GINAC_LT_AGE 0
- #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 0
```

7.91.2.4 GINAC_LT_CURRENT

```
#define GINAC_LT_CURRENT 11
```

7.91.2.5 GINAC_LT_REVISION

```
#define GINAC_LT_REVISION 0
```

7.91.2.6 GINAC_LT_AGE

```
#define GINAC_LT_AGE 0
```

7.91.2.7 GINACLIB_ARCHIVE_VERSION

```
#define GINACLIB_ARCHIVE_VERSION 3
```

Referenced by GiNaC::operator<<(), and GiNaC::operator>>().

7.91.2.8 GINACLIB_ARCHIVE_AGE

```
#define GINACLIB_ARCHIVE_AGE 3
```

Referenced by GiNaC::operator>>().

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

- 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

- 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