HepMC Reference Manual

2.06.09

Generated by Doxygen 1.4.7

Tue Jun 5 15:50:47 2012

Contents

Chapter 1

HepMC Directory Hierarchy

1.1 HepMC Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 2

HepMC Namespace Index

2.1 HepMC Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

HepMC Hierarchical Index

3.1 HepMC Class Hierarchy

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

Chapter 4

HepMC Class Index

4.1 HepMC Class List

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

Chapter 5

HepMC File Index

5.1 HepMC File List

Here is a list of all files with brief descriptions:

Chapter 6

HepMC Page Index

6.1 HepMC Related Pages

Here is a list of all related documentation pages:

Chapter 7

HepMC Directory Documentation

7.1 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/examples/ Directory Reference

Directories

- directory **fio**
- **directory pythia8**

Files

- file **example_BuildEventFromScratch.cc**
- **file example_EventSelection.cc**
- **file example_UsingIterators.cc**
- **file example_VectorConversion.cc**
- **file examples/list_of_examples.cc**
- **file VectorConversion.h**

7.2 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/fio/ Directory Reference

Files

- file **HEPEVT_Wrapper.cc**
- **file HerwigWrapper.cc**
- **file IO_HEPEVT.cc**
- **file IO_HERWIG.cc**

7.3 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/examples/fio/ Directory Reference

Files

- file **example_MyHerwig.cc**
- **file example_MyPythia.cc**
- **file example_MyPythiaOnlyToHepMC.cc**
- **file example_PythiaStreamIO.cc**
- **file initPythia.cc**
- **file PythiaHelper.h**
- **file testHerwigCopies.cc**
- **file testPythiaCopies.cc**

7.4 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/HepMC/ Directory Reference

Files

- file **CompareGenEvent.h**
- **file enable_if.h**
- **file Flow.h**
- **file GenCrossSection.h**
- **file GenEvent.h**
- **file GenParticle.h**
- **file GenRanges.h**
- **file GenVertex.h**
- **file HeavyIon.h**
- **file HEPEVT_Wrapper.h**
- **file HepMCDefs.h**
- **file HerwigWrapper.h**
- **file IO_AsciiParticles.h**
- **file IO_BaseClass.h**
- **file IO_Exception.h**
- **file IO_GenEvent.h**
- **file IO_HEPEVT.h**
- **file IO_HERWIG.h**
- **file is_arithmetic.h**
- **file IteratorRange.h**
- **file PdfInfo.h**
- **file Polarization.h**
- **file PythiaWrapper.h**
- **file PythiaWrapper6_4.h**
- **file PythiaWrapper6_4_WIN32.h**
- **file SearchVector.h**
- **file SimpleVector.h**
- **file StreamHelpers.h**
- **file StreamInfo.h**
- **file TempParticleMap.h**
- **file Units.h**
- **file Version.h**
- **file WeightContainer.h**

7.5 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/examples/pythia8/ Directory Reference

Files

- file **main31.cc**
- **file main32.cc**

7.6 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/src/ Directory Reference

Files

- file **CompareGenEvent.cc**
- **file filterEvent.cc**
- **file Flow.cc**
- **file GenCrossSection.cc**
- **file GenEvent.cc**
- **file GenEventStreamIO.cc**
- **file GenParticle.cc**
- **file GenRanges.cc**
- **file GenVertex.cc**
- **file HeavyIon.cc**
- **file IO_AsciiParticles.cc**
- **file IO_GenEvent.cc**
- **file PdfInfo.cc**
- **file Polarization.cc**
- **file SearchVector.cc**
- **file StreamHelpers.cc**
- **file StreamInfo.cc**
- **file WeightContainer.cc**

7.7 /home/cepa01/garren/lcg/hepmc/HepMC-2.06.09/test/ Directory Reference

Files

- file **IsGoodEvent.h**
- **file test/list_of_examples.cc**
- **file testFlow.cc**
- **file testHepMCIteration.h**
- **file testHepMCMethods.cc**
- **file testHepMCMethods.h**
- **file testPolarization.cc**
- **file testPrintBug.cc**
- **file testSimpleVector.cc**
- **file testUnits.cc**
- **file testWeights.cc**

Chapter 8

HepMC Namespace Documentation

8.1 CLHEP Namespace Reference

8.1.1 Detailed Description

CLHEP (p. 23) Vector classes are used in one of the examples

8.2 detail Namespace Reference

8.2.1 Detailed Description

internal namespace

8.3 HepMC Namespace Reference

Classes

- class **Flow**
  The flow object.
- **class GenCrossSection**
  The GenCrossSection (p. 71) class stores the generated cross section.
- **class GenEvent**
  The GenEvent (p. 75) class is the core of HepMC (p. 25).
- **class GenParticle**
  The GenParticle (p. 113) class contains information about generated particles.
- **class GenEventVertexRange**
  GenEventVertexRange (p. 112) acts like a collection of vertices.
- **class ConstGenEventVertexRange**
  ConstGenEventVertexRange (p. 47) acts like a collection of vertices.
- **class GenEventParticleRange**
  GenEventParticleRange (p. 111) acts like a collection of particles.
- **class ConstGenEventParticleRange**
  ConstGenEventParticleRange (p. 45) acts like a collection of particles.
- **class GenVertexParticleRange**
  GenVertexParticleRange (p. 153) acts like a collection of particles.
- **class GenParticleProductionRange**
  GenParticleProductionRange (p. 126) acts like a collection of particles.
- **class ConstGenParticleProductionRange**
- **class GenParticleEndRange**
  GenParticleEndRange (p. 124) acts like a collection of particles.
- **class ConstGenParticleEndRange**
- **class GenVertex**
  GenVertex (p. 128) contains information about decay vertices.
- **class HeavyIon**
  The HeavyIon (p. 154) class stores information about heavy ions.
- **class HEPEVT_Wrapper**
  Generic Wrapper for the fortran HEPEVT common block.
- **class IO_AsciiParticles**
  event input/output in ascii format for eye and machine reading

- **class IO_BaseClass**
  all input/output classes inherit from IO_BaseClass (p. 181)
- **class IO_Exception**
  IO exception handling.
- **class IO_GenEvent**
  IO_GenEvent (p. 186) also deals with HeavyIon (p. 154) and PdfInfo (p. 222).
- **class IO_HEPEVT**
  HEPEVT IO class.
- **class IO_HERWIG**
  IO_HERWIG (p. 195) is used to get Herwig information.
- **class PdfInfo**
  The PdfInfo (p. 222) class stores PDF information.
- **class Polarization**
  The Polarization (p. 234) class stores theta and phi for a GenParticle (p. 113).
- **class FourVector**
  FourVector (p. 61) is a simple representation of a physics 4 vector.
- **class ThreeVector**
  ThreeVector (p. 256) is a simple representation of a position or displacement 3 vector.
- **class StreamInfo**
  StreamInfo (p. 247) contains extra information needed when using streaming IO.
- **class TempParticleMap**
  TempParticleMap (p. 253) is a temporary GenParticle∗ container used during input.
- **class WeightContainer**
  Container for the Weights associated with an event or vertex.

Namespaces

- namespace **detail**
- **namespace Units**

Enumerations

- enum **IteratorRange {**
  parents, children, family, ancestors,
  descendants, relatives }

type of iteration

- **enum known_io {**
  gen = 1, ascii, extascii, ascii_pdt,
  extascii_pdt }
  The known_io enum is used to track which type of input is being read.

Functions

- **GenCrossSection getHerwigCrossSection (int ngen)**
- **bool compareGenEvent (GenEvent ∗, GenEvent ∗)**
- **bool compareSignalProcessVertex (GenEvent ∗, GenEvent ∗)**
- **bool compareBeamParticles (GenEvent ∗, GenEvent ∗)**
- **bool compareWeights (GenEvent ∗, GenEvent ∗)**
- **bool compareVertices (GenEvent ∗, GenEvent ∗)**
- **bool compareParticles (GenEvent ∗, GenEvent ∗)**
- **bool compareVertex (GenVertex ∗v1, GenVertex ∗v2)**
- **std::ostream & operator<< (std::ostream &os, GenCrossSection &xs)**
- **std::istream & operator>> (std::istream &is, GenCrossSection &xs)**
- **template<class InputIterator, class OutputIterator, class Predicate> void copy_if (InputIterator first, InputIterator last, OutputIterator out, Predicate pred)**
  define the type of iterator to use
- **std::ostream & operator<< (std::ostream &, GenEvent &)**
  standard streaming IO output operator
- **std::istream & operator>> (std::istream &, GenEvent &)**
  standard streaming IO input operator
- **std::istream & set_input_units (std::istream &, Units::MomentumUnit, Units::LengthUnit)**
  set the units for this input stream
- **std::ostream & write_HepMC_IO_block_begin (std::ostream &)**
  Explicitly write the begin block lines that IO_GenEvent (p. *186) uses.*
- **std::ostream & write_HepMC_IO_block_end (std::ostream &)**
  Explicitly write the end block line that IO_GenEvent (p. *186) uses.*
- **GenEvent & convert_units (GenEvent &evt, Units::MomentumUnit m, Units::LengthUnit l)**
- **std::ostream & operator<< (std::ostream &, HeavyIon const ∗)**
  Write the contents of HeavyIon (p. *154) to an output stream.*
- **std::istream & operator>> (std::istream &, HeavyIon ∗)**
  Read the contents of HeavyIon (p. *154) from an input stream.*
- **std::ostream & operator<< (std::ostream &, PdfInfo const ∗)**
- **std::istream & operator>> (std::istream &, PdfInfo ∗)**
- **GenCrossSection getPythiaCrossSection ()**

calculate the Pythia cross section and statistical error

- **bool not_in_vector (std::vector< HepMC::GenParticle * > *, GenParticle *)**

  returns true if it cannot find GenParticle in the vector*

- **std::vector< HepMC::GenParticle * >::iterator already_in_vector (std::vector< GenParticle * > *v, GenParticle *p)**

  returns true if GenParticle (p. 113) is in the vector

- **void version (std::ostream &os=std::cout)**

  print HepMC (p. 25) version

- **void writeVersion (std::ostream &os)**

  write HepMC (p. 25) version to os

- **std::string versionName ()**

  return HepMC (p. 25) version

- **std::ostream & operator<< (std::ostream &ostr, const Flow &f)**

  for printing

- **void HepMCStreamCallback (std::ios_base::event e, std::ios_base &b, int i)**
- **template<class IO> StreamInfo & get_stream_info (IO &iost)**
- **std::ostream & establish_output_stream_info (std::ostream &os)**

  used by IO_GenEvent (p. 186) constructor

- **std::istream & establish_input_stream_info (std::istream &is)**

  used by IO_GenEvent (p. 186) constructor

- **std::ostream & operator<< (std::ostream &ostr, const GenParticle &part)**

  print particle

- **std::ostream & operator<< (std::ostream &ostr, const GenVertex &vtx)**

  print vertex information

- **std::ostream & operator<< (std::ostream &ostr, const Polarization &polar)**

  print polarization information

Variables

- static const double **HepMC_pi = 3.14159265358979323846**

8.3.1 Detailed Description

All classes in the **HepMC** (p. 25) packages are in the **HepMC** (p. 25) namespace

8.3.2 Enumeration Type Documentation

8.3.2.1 enum HepMC::IteratorRange

type of iteration

Enumerator:

parents
children
family
ancestors
descendants
relatives

Definition at line 17 of file IteratorRange.h.

8.3.2.2 enum HepMC::known_io

The known_io enum is used to track which type of input is being read.

Enumerator:

gen
ascii
extascii
ascii_pdt
extascii_pdt

Definition at line 17 of file StreamInfo.h.

8.3.3 Function Documentation

8.3.3.1 GenCrossSection HepMC::getHerwigCrossSection (int *ngen*)

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 24 of file HerwigWrapper.cc.

References hwevnt, and HepMC::GenCrossSection::set_cross_section().

Referenced by main().

8.3.3.2 bool HepMC::compareGenEvent (GenEvent *, GenEvent *)

Examples:

fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, and testMultipleCopies.cc.in.

Definition at line 16 of file CompareGenEvent.cc.

References HepMC::GenEvent::alphaQCD(), HepMC::GenEvent::alphaQED(), compareBeamParticles(), compareParticles(), compareSignalProcessVertex(), compareVertices(), compareWeights(), HepMC::GenEvent::event_number(), HepMC::GenEvent::event_scale(), HepMC::GenEvent::heavy_ion(), HepMC::GenEvent::mpi(), HepMC::GenEvent::pdf_info(), HepMC::GenEvent::random_states(), and HepMC::GenEvent::signal_process_id().

Referenced by main().

8.3.3.3 bool HepMC::compareSignalProcessVertex (GenEvent *, GenEvent *)

Definition at line 64 of file CompareGenEvent.cc.

References HepMC::GenEvent::signal_process_vertex().

Referenced by compareGenEvent().

8.3.3.4 bool HepMC::compareBeamParticles (GenEvent *, GenEvent *)

Definition at line 77 of file CompareGenEvent.cc.

References HepMC::GenEvent::beam_particles().

Referenced by compareGenEvent().

8.3.3.5 bool HepMC::compareWeights (GenEvent *, GenEvent *)

Definition at line 92 of file CompareGenEvent.cc.

References HepMC::GenEvent::weights().

Referenced by compareGenEvent().

8.3.3.6 bool HepMC::compareVertices (GenEvent *, GenEvent *)

Definition at line 120 of file CompareGenEvent.cc.

References HepMC::GenEvent::barcode_to_vertex(), compareVertex(), v, HepMC::GenEvent::vertices_begin(), HepMC::GenEvent::vertices_end(), and HepMC::GenEvent::vertices_size().

Referenced by compareGenEvent().

8.3.3.7 bool HepMC::compareParticles (GenEvent *, GenEvent *)

Definition at line 98 of file CompareGenEvent.cc.

References HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), and HepMC::GenEvent::particles_size().

Referenced by compareGenEvent().

8.3.3.8 bool HepMC::compareVertex (GenVertex * *v1*, GenVertex * *v2*)

Definition at line 141 of file CompareGenEvent.cc.

References HepMC::GenVertex::barcode(), HepMC::GenVertex::particles_in_const_begin(), HepMC::GenVertex::particles_in_const_end(), HepMC::GenVertex::particles_in_size(), HepMC::GenVertex::particles_out_const_begin(), HepMC::GenVertex::particles_out_const_end(), HepMC::GenVertex::particles_out_size(), and HepMC::GenVertex::position().

Referenced by compareVertices().

8.3.3.9 std::ostream& HepMC::operator<< (std::ostream & *os*, GenCrossSection & *xs*) [inline]

Definition at line 89 of file GenCrossSection.h.

References HepMC::GenCrossSection::write().

8.3.3.10 std::istream& HepMC::operator>> (std::istream & *is*, GenCrossSection & *xs*) [inline]

Definition at line 92 of file GenCrossSection.h.

References HepMC::GenCrossSection::read().

8.3.3.11 template<class InputIterator, class OutputIterator, class Predicate> void HepMC::copy_if (InputIterator *first*, InputIterator *last*, OutputIterator *out*, Predicate *pred*)

define the type of iterator to use

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 50 of file GenEvent.h.

Referenced by main().

8.3.3.12 std::ostream & HepMC::operator<< (std::ostream &, GenEvent &)

standard streaming IO output operator

Writes evt to an output stream.

Definition at line 355 of file GenEventStreamIO.cc.

References HepMC::GenEvent::write().

8.3.3.13 std::istream & HepMC::operator>> (std::istream &, GenEvent &)

standard streaming IO input operator

Definition at line 362 of file GenEventStreamIO.cc.

References HepMC::GenEvent::read().

8.3.3.14 std::istream & HepMC::set_input_units (std::istream &, Units::MomentumUnit, Units::LengthUnit)

set the units for this input stream

Examples:

testStreamIO.cc.in.

Definition at line 370 of file GenEventStreamIO.cc.

References get_stream_info(), and HepMC::StreamInfo::use_input_units().

Referenced by HepMC::IO_GenEvent::use_input_units().

8.3.3.15 std::ostream & HepMC::write_HepMC_IO_block_begin (std::ostream &)

Explicitly write the begin block lines that **IO_GenEvent** (p. 186) uses.

Examples:

fio/example_PythiaStreamIO.cc, and testStreamIO.cc.in.

Definition at line 382 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), get_stream_info(), HepMC::StreamInfo::IO_GenEvent_Key(), and versionName().

Referenced by readPythiaStreamIO(), HepMC::IO_GenEvent::write_event(), and writePythiaStreamIO().

8.3.3.16 std::ostream & HepMC::write_HepMC_IO_block_end (std::ostream &)

Explicitly write the end block line that **IO_GenEvent** (p. 186) uses.

Examples:

fio/example_PythiaStreamIO.cc, and testStreamIO.cc.in.

Definition at line 395 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), get_stream_info(), and HepMC::StreamInfo::IO_GenEvent_End().

Referenced by readPythiaStreamIO(), HepMC::IO_GenEvent::write_comment(), writePythiaStreamIO(), and HepMC::IO_GenEvent::~IO_GenEvent().

8.3.3.17 GenEvent& HepMC::convert_units (GenEvent & *evt*, Units::MomentumUnit *m*, Units::LengthUnit *l*) [inline]

Definition at line 665 of file GenEvent.h.

References HepMC::GenEvent::use_units().

8.3.3.18 std::ostream & HepMC::operator<< (std::ostream & *os*, HeavyIon const * *ion*)

Write the contents of **HeavyIon** (p. 154) to an output stream.

Write the contents of **HeavyIon** (p. 154) to an output stream. **GenEvent** (p. 75) stores a pointer to a **HeavyIon** (p. 154).

Definition at line 23 of file HeavyIon.cc.

References HepMC::HeavyIon::eccentricity(), HepMC::HeavyIon::event_plane_angle(), HepMC::HeavyIon::impact_parameter(), HepMC::HeavyIon::N_Nwounded_collisions(), HepMC::HeavyIon::Ncoll(), HepMC::HeavyIon::Ncoll_hard(), HepMC::HeavyIon::Npart_proj(), HepMC::HeavyIon::Npart_targ(), HepMC::HeavyIon::Nwounded_N_collisions(), HepMC::HeavyIon::Nwounded_Nwounded_collisions(), HepMC::detail::output(), HepMC::HeavyIon::sigma_inel_NN(), HepMC::HeavyIon::spectator_neutrons(), and HepMC::HeavyIon::spectator_protons().

8.3.3.19 std::istream & HepMC::operator>> (std::istream & *is*, HeavyIon * *ion*)

Read the contents of **HeavyIon** (p. 154) from an input stream.

Read the contents of **HeavyIon** (p. 154) from an input stream. **GenEvent** (p. 75) stores a pointer to a **HeavyIon** (p. 154).

Definition at line 71 of file HeavyIon.cc.

References HepMC::HeavyIon::set_eccentricity(), HepMC::HeavyIon::set_event_plane_angle(), HepMC::HeavyIon::set_impact_parameter(), HepMC::HeavyIon::set_N_Nwounded_collisions(), HepMC::HeavyIon::set_Ncoll(), HepMC::HeavyIon::set_Ncoll_hard(), HepMC::HeavyIon::set_Npart_proj(), HepMC::HeavyIon::set_Npart_targ(), HepMC::HeavyIon::set_Nwounded_N_collisions(), HepMC::HeavyIon::set_Nwounded_Nwounded_collisions(), HepMC::HeavyIon::set_sigma_inel_NN(), HepMC::HeavyIon::set_spectator_neutrons(), and HepMC::HeavyIon::set_spectator_protons().

8.3.3.20 std::ostream & HepMC::operator<< (std::ostream &, PdfInfo const *)

Definition at line 21 of file PdfInfo.cc.

References HepMC::PdfInfo::id1(), HepMC::PdfInfo::id2(), HepMC::detail::output(), HepMC::PdfInfo::pdf1(), HepMC::PdfInfo::pdf2(), HepMC::PdfInfo::pdf_id1(), HepMC::PdfInfo::pdf_id2(), HepMC::PdfInfo::scalePDF(), HepMC::PdfInfo::x1(), and HepMC::PdfInfo::x2().

8.3.3.21 std::istream & HepMC::operator>> (std::istream &, PdfInfo *)

Definition at line 59 of file PdfInfo.cc.

References HepMC::PdfInfo::set_id1(), HepMC::PdfInfo::set_id2(), HepMC::PdfInfo::set_pdf1(), HepMC::PdfInfo::set_pdf2(), HepMC::PdfInfo::set_pdf_id1(), HepMC::PdfInfo::set_pdf_id2(), HepMC::PdfInfo::set_scalePDF(), HepMC::PdfInfo::set_x1(), HepMC::PdfInfo::set_x2(), and x1.

8.3.3.22 GenCrossSection HepMC::getPythiaCrossSection () [inline]

calculate the Pythia cross section and statistical error

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, and fio/testPythiaCopies.cc.

Definition at line 28 of file PythiaWrapper.h.

References pyint5, and HepMC::GenCrossSection::set_cross_section().

Referenced by event_selection(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), and write-PythiaStreamIO().

8.3.3.23 bool HepMC::not_in_vector (std::vector< HepMC::GenParticle * > *, GenParticle *)

returns true if it cannot find GenParticle* in the vector

Definition at line 11 of file SearchVector.cc.

References already_in_vector(), and p.

Referenced by HepMC::Flow::connected_partners(), and HepMC::Flow::dangling_connected_partners().

8.3.3.24 std::vector< HepMC::GenParticle * >::iterator HepMC::already_in_vector (std::vector< HepMC::GenParticle * > *, GenParticle *)

returns true if **GenParticle** (p. 113) is in the vector

Returns the index of a GenParticle* within a vector. Returns -1 if GenParticle* is not in the vector.

Definition at line 18 of file SearchVector.cc.

References p.

Referenced by not_in_vector(), HepMC::GenVertex::remove_particle_in(), and HepMC::Gen-Vertex::remove_particle_out().

8.3.3.25 void HepMC::version (std::ostream & *os* = `std::cout`) `[inline]`

print **HepMC** (p. 25) version

Examples:

testMass.cc.in.

Definition at line 27 of file Version.h.

References versionName().

8.3.3.26 void HepMC::writeVersion (std::ostream & *os*) `[inline]`

write **HepMC** (p. 25) version to os

Definition at line 33 of file Version.h.

References versionName().

Referenced by HepMC::GenEvent::print_version().

8.3.3.27 std::string HepMC::versionName () `[inline]`

return **HepMC** (p. 25) version

Definition at line 22 of file Version.h.

References HEPMC_VERSION.

Referenced by version(), HepMC::IO_AsciiParticles::write_event(), write_HepMC_IO_block_begin(), and writeVersion().

8.3.3.28 std::ostream& HepMC::operator<< (std::ostream & *ostr*, const Flow & *f*)

for printing

Definition at line 190 of file Flow.cc.

References HepMC::Flow::m_icode.

8.3.3.29 void HepMC::HepMCStreamCallback (std::ios_base::event *e*, std::ios_base & *b*, int *i*)

This method is called by the stream destructor. It does cleanup on stored user data (**StreamInfo** (p. 247)) and is registered by the first call to **get_stream_info()** (p. 35).

Definition at line 29 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::stream_id().

Referenced by get_stream_info().

8.3.3.30 template<class IO> StreamInfo& HepMC::get_stream_info (IO & *iost*)

A custom iomanip that allows us to store and access user data (**StreamInfo** (p. 247)) associated with the stream. This method creates the **StreamInfo** (p. 247) object the first time it is called.

Definition at line 51 of file GenEventStreamIO.cc.

References HepMCStreamCallback().

Referenced by HepMC::detail::establish_input_stream_info(), establish_input_stream_info(), HepMC::detail::establish_output_stream_info(), establish_output_stream_info(), HepMC::GenEvent::read(), HepMC::detail::read_particle(), set_input_units(), HepMC::GenEvent::write(), write_HepMC_IO_block_begin(), and write_HepMC_IO_block_end().

8.3.3.31 std::ostream& HepMC::establish_output_stream_info (std::ostream & *os*)

used by **IO_GenEvent** (p. 186) constructor

Definition at line 653 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), and get_stream_info().

Referenced by HepMC::IO_GenEvent::IO_GenEvent().

8.3.3.32 std::istream& HepMC::establish_input_stream_info (std::istream & *is*)

used by **IO_GenEvent** (p. 186) constructor

Definition at line 667 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), and get_stream_info().

Referenced by HepMC::IO_GenEvent::IO_GenEvent().

8.3.3.33 std::ostream& HepMC::operator<< (std::ostream & *ostr*, const GenParticle & *part*)

print particle

Definition at line 189 of file GenParticle.cc.

References HepMC::GenVertex::barcode(), HepMC::GenParticle::barcode(), HepMC::FourVector::e(), HepMC::GenParticle::end_vertex(), HepMC::GenParticle::momentum(), HepMC::GenParticle::pdg_id(), HepMC::FourVector::px(), HepMC::FourVector::py(), HepMC::FourVector::pz(), and HepMC::GenParticle::status().

8.3.3.34 std::ostream& HepMC::operator<< (std::ostream & *ostr*, const GenVertex & *vtx*)

print vertex information

Definition at line 440 of file GenVertex.cc.

References HepMC::GenVertex::barcode(), HepMC::GenVertex::position(), and HepMC::FourVector::x().

8.3.3.35 std::ostream& HepMC::operator<< (std::ostream & *ostr*, const Polarization & *polar*)

print polarization information

Definition at line 129 of file Polarization.cc.

References HepMC::Polarization::phi(), and HepMC::Polarization::theta().

8.3.4 Variable Documentation

8.3.4.1 const double HepMC::HepMC_pi = 3.14159265358979323846 [static]

Definition at line 19 of file Polarization.h.

8.4 HepMC::detail Namespace Reference

Classes

- struct **enable_if**
  internal - used to decide if a class is arithmetic
- **struct enable_if< true, T >**
  internal - use if class T is arithmetic
- **struct disable_if**
  internal - used by SimpleVector to decide if a class is arithmetic
- **struct disable_if< false, T >**
  internal - used by SimpleVector to decide if a class is arithmetic
- **struct is_arithmetic**
  undefined and therefore non-arithmetic
- **struct is_arithmetic< char >**
  character is arithmetic
- **struct is_arithmetic< unsigned char >**
  unsigned character is arithmetic
- **struct is_arithmetic< signed char >**
  signed character is arithmetic
- **struct is_arithmetic< short >**
  short is arithmetic
- **struct is_arithmetic< unsigned short >**
  unsigned short is arithmetic
- **struct is_arithmetic< int >**
  int is arithmetic
- **struct is_arithmetic< unsigned int >**
  unsigned int is arithmetic
- **struct is_arithmetic< long >**
  long is arithmetic
- **struct is_arithmetic< unsigned long >**
  unsigned long is arithmetic
- **struct is_arithmetic< float >**
  float is arithmetic
- **struct is_arithmetic< double >**

double is arithmetic

- **struct is_arithmetic< long double >**

  long double is arithmetic

Functions

- std::ostream & **establish_output_stream_info (std::ostream &)**

  *used by IO_GenEvent (*p. *186) constructor*

- **std::istream & establish_input_stream_info (std::istream &)**

  *used by IO_GenEvent (*p. *186) constructor*

- **std::istream & read_vertex (std::istream &, TempParticleMap &, GenVertex ∗)**
- **std::istream & read_particle (std::istream &, TempParticleMap &, GenParticle ∗)**
- **std::ostream & output (std::ostream &os, const double &d)**

  write a double - for internal use by streaming IO

- **std::ostream & output (std::ostream &os, const float &d)**

  write a float - for internal use by streaming IO

- **std::ostream & output (std::ostream &os, const int &i)**

  write an int - for internal use by streaming IO

- **std::ostream & output (std::ostream &os, const long &i)**

  write a long - for internal use by streaming IO

- **std::ostream & output (std::ostream &os, const char &c)**

  write a single char - for internal use by streaming IO

- **std::istream & find_event_end (std::istream &)**

  used to read to the end of a bad event

8.4.1 Function Documentation

8.4.1.1 std::ostream & HepMC::detail::establish_output_stream_info (std::ostream &)

used by **IO_GenEvent** (p. 186) constructor

Definition at line 769 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), and HepMC::get_stream_info().

Referenced by HepMC::IO_GenEvent::IO_GenEvent().

8.4.1.2 std::istream & HepMC::detail::establish_input_stream_info (std::istream &)

used by **IO_GenEvent** (p. 186) constructor

Definition at line 783 of file GenEventStreamIO.cc.

References HepMC::StreamInfo::finished_first_event(), and HepMC::get_stream_info().

Referenced by HepMC::IO_GenEvent::IO_GenEvent().

8.4.1.3 std::istream & HepMC::detail::read_vertex (std::istream &, TempParticleMap &, GenVertex ∗)

get a **GenVertex** (p. 128) from ASCII input **TempParticleMap** (p. 253) is used to track the associations of particles with vertices

Definition at line 23 of file StreamHelpers.cc.

References read_particle(), and v.

Referenced by HepMC::GenEvent::read().

8.4.1.4 std::istream & HepMC::detail::read_particle (std::istream &, TempParticleMap &, GenParticle ∗)

get a **GenParticle** (p. 113) from ASCII input **TempParticleMap** (p. 253) is used to track the associations of particles with vertices

Definition at line 688 of file GenEventStreamIO.cc.

References HepMC::TempParticleMap::addEndParticle(), HepMC::ascii, HepMC::get_stream_info(), HepMC::StreamInfo::io_type(), p, and HepMC::Flow::set_icode().

Referenced by read_vertex().

8.4.1.5 std::ostream& HepMC::detail::output (std::ostream & *os*, const double & *d*) [inline]

write a double - for internal use by streaming IO

Definition at line 35 of file StreamHelpers.h.

Referenced by HepMC::Flow::connected_partners(), HepMC::Flow::dangling_connected_partners(), HepMC::operator<<(), HepMC::GenEvent::write(), and HepMC::IO_AsciiParticles::write_event().

8.4.1.6 std::ostream& HepMC::detail::output (std::ostream & *os*, const float & *d*) [inline]

write a float - for internal use by streaming IO

Definition at line 47 of file StreamHelpers.h.

8.4.1.7 std::ostream& HepMC::detail::output (std::ostream & *os*, const int & *i*) [inline]

write an int - for internal use by streaming IO

Definition at line 59 of file StreamHelpers.h.

8.4.1.8 std::ostream& HepMC::detail::output (std::ostream & *os*, const long & *i*) [inline]

write a long - for internal use by streaming IO

Definition at line 71 of file StreamHelpers.h.

8.4.1.9 std::ostream& HepMC::detail::output (std::ostream & *os*, const char & *c*) [inline]

write a single char - for internal use by streaming IO

Definition at line 83 of file StreamHelpers.h.

8.4.1.10 std::istream & HepMC::detail::find_event_end (std::istream &)

used to read to the end of a bad event

Definition at line 98 of file StreamHelpers.cc.

Referenced by HepMC::GenEvent::read().

8.5 HepMC::Units Namespace Reference

Enumerations

- enum **MomentumUnit { MEV, GEV }**
- **enum LengthUnit { MM, CM }**

Functions

- **LengthUnit default_length_unit ()**

  default unit is defined by configure

- **MomentumUnit default_momentum_unit ()**

  default unit is defined by configure

- **std::string name (MomentumUnit)**

  convert enum to string

- **std::string name (LengthUnit)**

  convert enum to string

- **double conversion_factor (MomentumUnit from, MomentumUnit to)**

  scaling factor relative to MeV

- **double conversion_factor (LengthUnit from, LengthUnit to)**

8.5.1 Enumeration Type Documentation

8.5.1.1 enum HepMC::Units::MomentumUnit

Enumerator:

MEV

GEV

Definition at line 25 of file Units.h.

8.5.1.2 enum HepMC::Units::LengthUnit

Enumerator:

MM

CM

Definition at line 26 of file Units.h.

8.5.2 Function Documentation

8.5.2.1 LengthUnit HepMC::Units::default_length_unit ()

default unit is defined by configure

Examples:

testUnits.cc.

Referenced by HepMC::GenEvent::clear(), and main().

8.5.2.2 MomentumUnit HepMC::Units::default_momentum_unit ()

default unit is defined by configure

Examples:

testUnits.cc.

Referenced by HepMC::GenEvent::clear(), and main().

8.5.2.3 std::string HepMC::Units::name (MomentumUnit)

convert enum to string

Examples:

testHepMC.cc.in, testStreamIO.cc.in, and testUnits.cc.

Referenced by main(), HepMC::GenEvent::write(), and HepMC::GenEvent::write_units().

8.5.2.4 std::string HepMC::Units::name (LengthUnit)

convert enum to string

8.5.2.5 double HepMC::Units::conversion_factor (MomentumUnit *from*, MomentumUnit *to*)

scaling factor relative to MeV

Examples:

testUnits.cc.

Referenced by main(), and repairUnits().

8.5.2.6 double HepMC::Units::conversion_factor (LengthUnit *from*, LengthUnit *to*)

8.6 Pythia8 Namespace Reference

8.7 Units Namespace Reference

8.7.1 Detailed Description

Allow units to be specified within **HepMC** (p. 25). The default units are set at compile time.

Chapter 9

HepMC Class Documentation

9.1 HepMC::ConstGenEventParticleRange Class Reference

ConstGenEventParticleRange (p. 45) acts like a collection of particles.

`#include <GenRanges.h>`

Public Member Functions

- **ConstGenEventParticleRange (GenEvent const &e)**

  the constructor requires a const GenEvent (p. *75)*

- **GenEvent::particle_const_iterator begin () const**
- **GenEvent::particle_const_iterator end () const**

9.1.1 Detailed Description

ConstGenEventParticleRange (p. 45) acts like a collection of particles.

HepMC::ConstGenEventParticleRange (p. 45) is used to mimic a collection of particles for ease of use - especially with utilities such as the Boost foreach funtion This is the const partner of **GenEventParticleRange** (p. 111)

Definition at line 112 of file GenRanges.h.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 HepMC::ConstGenEventParticleRange::ConstGenEventParticleRange (GenEvent const & *e*) `[inline]`

the constructor requires a const **GenEvent** (p. 75)

Definition at line 117 of file GenRanges.h.

9.1.3 Member Function Documentation

9.1.3.1 GenEvent::particle_const_iterator HepMC::ConstGenEventParticleRange::begin () const [inline]

Definition at line 119 of file GenRanges.h.

References HepMC::GenEvent::particles_begin().

9.1.3.2 GenEvent::particle_const_iterator HepMC::ConstGenEventParticleRange::end () const [inline]

Definition at line 120 of file GenRanges.h.

References HepMC::GenEvent::particles_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.2 HepMC::ConstGenEventVertexRange Class Reference

ConstGenEventVertexRange (p. 47) acts like a collection of vertices.

`#include <GenRanges.h>`

Public Member Functions

- **ConstGenEventVertexRange (GenEvent const &e)**
  the constructor requires a const GenEvent (p. 75)
- **GenEvent::vertex_const_iterator begin () const**
- **GenEvent::vertex_const_iterator end () const**

9.2.1 Detailed Description

ConstGenEventVertexRange (p. 47) acts like a collection of vertices.

HepMC::ConstGenEventVertexRange (p. 47) is used to mimic a collection of vertices for ease of use - especially with utilities such as the Boost foreach funtion This is the const partner of **GenEventVertexRange** (p. 112)

Definition at line 55 of file GenRanges.h.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 HepMC::ConstGenEventVertexRange::ConstGenEventVertexRange (GenEvent const & *e*) [inline]

the constructor requires a const **GenEvent** (p. 75)

Definition at line 60 of file GenRanges.h.

9.2.3 Member Function Documentation

9.2.3.1 GenEvent::vertex_const_iterator HepMC::ConstGenEventVertexRange::begin () const [inline]

Definition at line 62 of file GenRanges.h.

References HepMC::GenEvent::vertices_begin().

9.2.3.2 GenEvent::vertex_const_iterator HepMC::ConstGenEventVertexRange::end () const [inline]

Definition at line 63 of file GenRanges.h.

References HepMC::GenEvent::vertices_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.3 HepMC::ConstGenParticleEndRange Class Reference

`#include <GenRanges.h>`

Public Member Functions

- **ConstGenParticleEndRange (GenParticle const &p, IteratorRange range=relatives)**
  *the constructor requires a GenParticle (*p. *113)*
- **GenVertex::particle_iterator begin ()**
  begin iterator throws an error if the particle end_vertex is undefined
- **GenVertex::particle_iterator end ()**
  end iterator throws an error if the particle end_vertex is undefined

9.3.1 Detailed Description

Definition at line 247 of file GenRanges.h.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 HepMC::ConstGenParticleEndRange::ConstGenParticleEndRange (GenParticle const & *p*, IteratorRange *range* = `relatives`) `[inline]`

the constructor requires a **GenParticle** (p. 113)

Definition at line 252 of file GenRanges.h.

9.3.3 Member Function Documentation

9.3.3.1 GenVertex::particle_iterator HepMC::ConstGenParticleEndRange::begin () `[inline]`

begin iterator throws an error if the particle end_vertex is undefined

Definition at line 313 of file GenRanges.h.

References HepMC::GenParticle::end_vertex(), and HepMC::GenVertex::particles_begin().

9.3.3.2 GenVertex::particle_iterator HepMC::ConstGenParticleEndRange::end () `[inline]`

end iterator throws an error if the particle end_vertex is undefined

Definition at line 319 of file GenRanges.h.

References HepMC::GenParticle::end_vertex(), and HepMC::GenVertex::particles_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.4 HepMC::ConstGenParticleProductionRange Class Reference

`#include <GenRanges.h>`

Public Member Functions

- **ConstGenParticleProductionRange (GenParticle const &p, IteratorRange range=relatives)**
  the constructor requires a GenParticle (p. ***113***)
- **GenVertex::particle_iterator begin ()**
  begin iterator throws an error if the particle production_vertex is undefined
- **GenVertex::particle_iterator end ()**
  end iterator throws an error if the particle production_vertex is undefined

9.4.1 Detailed Description

Definition at line 193 of file GenRanges.h.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 HepMC::ConstGenParticleProductionRange::ConstGenParticleProductionRange (GenParticle const & *p*, IteratorRange *range* = `relatives`) `[inline]`

the constructor requires a **GenParticle** (p. 113)

Definition at line 198 of file GenRanges.h.

9.4.3 Member Function Documentation

9.4.3.1 GenVertex::particle_iterator HepMC::ConstGenParticleProductionRange::begin () `[inline]`

begin iterator throws an error if the particle production_vertex is undefined

Definition at line 286 of file GenRanges.h.

References HepMC::GenVertex::particles_begin(), and HepMC::GenParticle::production_vertex().

9.4.3.2 GenVertex::particle_iterator HepMC::ConstGenParticleProductionRange::end () `[inline]`

end iterator throws an error if the particle production_vertex is undefined

Definition at line 293 of file GenRanges.h.

References HepMC::GenVertex::particles_end(), and HepMC::GenParticle::production_vertex().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.5 HepMC::detail::disable_if<, > Struct Template Reference

internal - used by SimpleVector to decide if a class is arithmetic

`#include <enable_if.h>`

9.5.1 Detailed Description

template<bool, class> struct HepMC::detail::disable_if<, >

internal - used by SimpleVector to decide if a class is arithmetic

Definition at line 33 of file enable_if.h.

The documentation for this struct was generated from the following file:

- **enable_if.h**

9.6 HepMC::detail::disable_if< false, T > Struct Template Reference

internal - used by SimpleVector to decide if a class is arithmetic

`#include <enable_if.h>`

Public Types

- typedef T **type**
  check type of class T

9.6.1 Detailed Description

template<class T> struct HepMC::detail::disable_if< false, T >

internal - used by SimpleVector to decide if a class is arithmetic

Definition at line 38 of file enable_if.h.

9.6.2 Member Typedef Documentation

9.6.2.1 template<class T> typedef T HepMC::detail::disable_if< false, T >::type

check type of class T

Definition at line 40 of file enable_if.h.

The documentation for this struct was generated from the following file:

- **enable_if.h**

9.7 HepMC::detail::enable_if<, > Struct Template Reference

internal - used to decide if a class is arithmetic

`#include <enable_if.h>`

9.7.1 Detailed Description

template<bool, class> struct HepMC::detail::enable_if<, >

internal - used to decide if a class is arithmetic

Definition at line 17 of file enable_if.h.

The documentation for this struct was generated from the following file:

- **enable_if.h**

9.8 HepMC::detail::enable_if< true, T > Struct Template Reference

internal - use if class T is arithmetic

`#include <enable_if.h>`

Public Types

- typedef T **type**
  check type of class T

9.8.1 Detailed Description

template<class T> struct HepMC::detail::enable_if< true, T >

internal - use if class T is arithmetic

Definition at line 22 of file enable_if.h.

9.8.2 Member Typedef Documentation

9.8.2.1 template<class T> typedef T HepMC::detail::enable_if< true, T >::type

check type of class T

Definition at line 24 of file enable_if.h.

The documentation for this struct was generated from the following file:

- **enable_if.h**

9.9 HepMC::Flow Class Reference

The flow object.

`#include <Flow.h>`

Public Types

- typedef std::map< int, int >::**iterator iterator**
  iterator for flow pattern container
- **typedef std::map< int, int >::const_iterator const_iterator**
  const iterator for flow pattern container

Public Member Functions

- **Flow (GenParticle *particle_owner=0)**
  default constructor
- **Flow (const Flow &)**
  copy
- **virtual ~Flow ()**
- **void swap (Flow &other)**
  swap
- **Flow & operator= (const Flow &)**
  make a copy
- **bool operator== (const Flow &a) const**
  equality
- **bool operator!= (const Flow &a) const**
  inequality
- **void print (std::ostream &ostr=std::cout) const**
  *print Flow (*p. *54) information to ostr*
- **std::vector< HepMC::GenParticle * > connected_partners (int code, int code_index=1, int num_indices=2) const**
- **std::vector< HepMC::GenParticle * > dangling_connected_partners (int code, int code_index=1, int num_indices=2) const**
- **const GenParticle * particle_owner () const**
  *find particle owning this Flow (*p. *54)*
- **int icode (int code_index=1) const**
  flow code
- **Flow set_icode (int code_index, int code)**

set flow code

- **Flow set_unique_icode (int code_index=1)**

  set unique flow code

- **bool empty () const**

  return true if there is no flow container

- **int size () const**

  size of flow pattern container

- **void clear ()**

  clear flow patterns

- **bool erase (int code_index)**

  empty flow pattern container

- **iterator begin ()**

  beginning of flow pattern container

- **iterator end ()**

  end of flow pattern container

- **const_iterator begin () const**

  beginning of flow pattern container

- **const_iterator end () const**

  end of flow pattern container

Protected Member Functions

- void **connected_partners (std::vector< HepMC::GenParticle * > *output, int code, int code_index, int num_indices) const**

  for internal use only

- **void dangling_connected_partners (std::vector< HepMC::GenParticle * > *output, std::vector< HepMC::GenParticle * > *visited_particles, int code, int code_index, int num_indices) const**

  for internal use only

Friends

- std::ostream & **operator<< (std::ostream &ostr, const Flow &f)**

  for printing

9.9.1 Detailed Description

The flow object.

The particle's flow object keeps track of an arbitrary number of flow patterns within a graph (i.e. color flow, charge flow, lepton number flow, ...) **Flow** (p. 54) patterns are coded with an integer, in the same manner as in Herwig.

Examples:

testFlow.cc.

Definition at line 66 of file Flow.h.

9.9.2 Member Typedef Documentation

9.9.2.1 typedef std::map<int,int>::const_iterator HepMC::Flow::const_iterator

const iterator for flow pattern container

Definition at line 128 of file Flow.h.

9.9.2.2 typedef std::map<int,int>::iterator HepMC::Flow::iterator

iterator for flow pattern container

Definition at line 126 of file Flow.h.

9.9.3 Constructor & Destructor Documentation

9.9.3.1 HepMC::Flow::Flow (GenParticle * *particle_owner* = 0)

default constructor

Definition at line 13 of file Flow.cc.

9.9.3.2 HepMC::Flow::Flow (const Flow &)

copy

copies both the m_icode AND the m_particle_owner

Definition at line 17 of file Flow.cc.

9.9.3.3 HepMC::Flow::~Flow () [virtual]

Definition at line 24 of file Flow.cc.

9.9.4 Member Function Documentation

9.9.4.1 Flow::const_iterator HepMC::Flow::begin () const [inline]

beginning of flow pattern container

Definition at line 186 of file Flow.h.

9.9.4.2 Flow::iterator HepMC::Flow::begin () [inline]

beginning of flow pattern container

Definition at line 184 of file Flow.h.

9.9.4.3 void HepMC::Flow::clear () [inline]

clear flow patterns

Definition at line 179 of file Flow.h.

9.9.4.4 void HepMC::Flow::connected_partners (std::vector< HepMC::GenParticle * > * *output*, int *code*, int *code_index*, int *num_indices*) const [protected]

for internal use only

protected: for recursive use by **Flow::connected_partners()** (p. 57)

Definition at line 60 of file Flow.cc.

References HepMC::GenParticle::end_vertex(), HepMC::family, HepMC::not_in_vector(), p, HepMC::GenVertex::particles_begin(), HepMC::GenVertex::particles_end(), and HepMC::GenParticle::production_vertex().

9.9.4.5 std::vector< GenParticle * > HepMC::Flow::connected_partners (int *code*, int *code_index* = 1, int *num_indices* = 2) const

returns all connected particles which have "code" in any of the num_indices beginning with index code_index.

Returns all flow partners which have "code" in any of the num_indices beginning with index code_index. m_particle_owner is included in the result. Return is by value since the set should never be very big. EXAMPLE: if you want to find all flow partners that have the same code in indices 2,3,4 as particle p has in index 2, you would use: set<GenParticle*> result = p->flow().connected_partners(p->flow().icode(2),2,3);

Definition at line 38 of file Flow.cc.

References icode(), and HepMC::detail::output().

9.9.4.6 void HepMC::Flow::dangling_connected_partners (std::vector< HepMC::GenParticle * > * *output*, std::vector< HepMC::GenParticle * > * *visited_particles*, int *code*, int *code_index*, int *num_indices*) const [protected]

for internal use only

protected: for recursive use by **Flow::dangling_connected_partners** (p. 58)

Definition at line 123 of file Flow.cc.

References HepMC::GenParticle::end_vertex(), HepMC::family, HepMC::not_in_vector(), p, HepMC::GenVertex::particles_begin(), HepMC::GenVertex::particles_end(), and HepMC::GenParticle::production_vertex().

9.9.4.7 std::vector< GenParticle * > HepMC::Flow::dangling_connected_partners (int *code*, int *code_index* = 1, int *num_indices* = 2) const

same as connected_partners, but returns only those particles which are connected to <=1 other particles (i.e. the flow line "dangles" at these particles)

Definition at line 108 of file Flow.cc.

References icode(), and HepMC::detail::output().

9.9.4.8 bool HepMC::Flow::empty () const [inline]

return true if there is no flow container

Definition at line 177 of file Flow.h.

9.9.4.9 Flow::const_iterator HepMC::Flow::end () const [inline]

end of flow pattern container

Definition at line 187 of file Flow.h.

9.9.4.10 Flow::iterator HepMC::Flow::end () [inline]

end of flow pattern container

Definition at line 185 of file Flow.h.

9.9.4.11 bool HepMC::Flow::erase (int *code_index*) [inline]

empty flow pattern container

Examples:

testFlow.cc.

Definition at line 180 of file Flow.h.

Referenced by main().

9.9.4.12 int HepMC::Flow::icode (int *code_index* = 1) const [inline]

flow code

Definition at line 163 of file Flow.h.

Referenced by connected_partners(), dangling_connected_partners(), and HepMC::GenParticle::flow().

9.9.4.13 bool HepMC::Flow::operator!= (const Flow & *a*) const [inline]

inequality

Definition at line 199 of file Flow.h.

9.9.4.14 Flow & HepMC::Flow::operator= (const Flow &) [inline]

make a copy

copies only the m_icode ... not the particle_owner this is intuitive behaviour so you can do oneparticle->flow() = otherparticle->flow()

Definition at line 202 of file Flow.h.

References m_icode.

9.9.4.15 bool HepMC::Flow::operator== (const Flow & *a*) const [inline]

equality

equivalent flows have the same flow codes for all flow_numbers (i.e. their m_icode maps are identical), but they need not have the same m_particle owner

Definition at line 193 of file Flow.h.

References m_icode.

9.9.4.16 const GenParticle * HepMC::Flow::particle_owner () const [inline]

find particle owning this **Flow** (p. 54)

Definition at line 160 of file Flow.h.

9.9.4.17 void HepMC::Flow::print (std::ostream & *ostr* = `std::cout`) const

print **Flow** (p. 54) information to ostr

Definition at line 34 of file Flow.cc.

9.9.4.18 Flow HepMC::Flow::set_icode (int *code_index*, int *code*) [inline]

set flow code

Definition at line 167 of file Flow.h.

Referenced by HepMC::detail::read_particle(), and HepMC::GenParticle::set_flow().

9.9.4.19 Flow HepMC::Flow::set_unique_icode (int *code_index* = `1`) [inline]

set unique flow code

use this method if you want to assign a unique flow code, but do not want the burden of choosing it yourself

Definition at line 171 of file Flow.h.

Referenced by HepMC::GenParticle::set_flow().

9.9.4.20 int HepMC::Flow::size () const [inline]

size of flow pattern container

Definition at line 178 of file Flow.h.

9.9.4.21 void HepMC::Flow::swap (Flow & *other*)

swap

Definition at line 28 of file Flow.cc.

References m_icode, and m_particle_owner.

Referenced by HepMC::GenParticle::swap().

9.9.5 Friends And Related Function Documentation

9.9.5.1 std::ostream& operator<< (std::ostream & *ostr*, const Flow & *f*) [friend]

for printing

Definition at line 190 of file Flow.cc.

The documentation for this class was generated from the following files:

- **Flow.h**
- **Flow.cc**

9.10 HepMC::FourVector Class Reference

FourVector (p. 61) is a simple representation of a physics 4 vector.

`#include <SimpleVector.h>`

Public Member Functions

- **FourVector (double xin, double yin, double zin, double tin=0)**
  constructor requiring at least x, y, and z
- **FourVector (double tin)**
  constructor requiring only t
- **FourVector ()**
- **template<class T> FourVector (const T &v, typename detail::disable_if< detail::is_arithmetic< T >::value, void >::type *=0)**
- **FourVector (const FourVector &v)**
  copy constructor
- **void swap (FourVector &other)**
  swap
- **double px () const**
  return px
- **double py () const**
  return py
- **double pz () const**
  return pz
- **double e () const**
  return E
- **double x () const**
  return x
- **double y () const**
  return y
- **double z () const**
  return z
- **double t () const**
  return t
- **double m2 () const**
  Invariant mass squared.

- **double m () const**

  *Invariant mass. If m2() (*p. *64) is negative then -sqrt(-m2()) is returned.*

- **double perp2 () const**

  Transverse component of the spatial vector squared.

- **double perp () const**

  Transverse component of the spatial vector (R in cylindrical system).

- **double theta () const**

  The polar angle.

- **double phi () const**

  The azimuth angle.

- **double rho () const**

  spatial vector component magnitude

- **FourVector & operator= (const FourVector &)**

  make a copy

- **bool operator== (const FourVector &) const**

  equality

- **bool operator!= (const FourVector &) const**

  inequality

- **double pseudoRapidity () const**

  Returns the pseudo-rapidity, i.e. -ln(tan(theta/2)).

- **double eta () const**

  Pseudorapidity (of the space part).

- **void set (double x, double y, double z, double t)**

  set x, y, z, and t

- **void setX (double xin)**

  set x

- **void setY (double yin)**

  set y

- **void setZ (double zin)**

  set z

- **void setT (double tin)**

  set t

- **void setPx (double xin)**

  set px

- **void setPy (double yin)**

  set py

- **void setPz (double zin)**

  set pz

- **void setE (double tin)**

  set E

9.10.1 Detailed Description

FourVector (p. 61) is a simple representation of a physics 4 vector.

For compatibility with existing code, the basic expected geometrical access methods are povided. Also, there is a templated constructor that will take another vector (HepLorentzVector, GenVector, ...) which must have the following methods: **x()** (p. 69), **y()** (p. 69), **z()** (p. 69), **t()** (p. 69).

Examples:

example_BuildEventFromScratch.cc, testFlow.cc, testPrintBug.cc, testSimpleVector.cc, and VectorConversion.h.

Definition at line 42 of file SimpleVector.h.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 HepMC::FourVector::FourVector (double *xin*, double *yin*, double *zin*, double *tin* = 0) [inline]

constructor requiring at least x, y, and z

Definition at line 47 of file SimpleVector.h.

9.10.2.2 HepMC::FourVector::FourVector (double *tin*) [inline]

constructor requiring only t

Definition at line 51 of file SimpleVector.h.

9.10.2.3 HepMC::FourVector::FourVector () [inline]

Definition at line 54 of file SimpleVector.h.

9.10.2.4 template<class T> HepMC::FourVector::FourVector (const T & *v*, typename detail::disable_if< detail::is_arithmetic< T >::value, void >::type * = 0) [inline]

templated constructor this is used ONLY if T is not arithmetic

Definition at line 60 of file SimpleVector.h.

9.10.2.5 HepMC::FourVector::FourVector (const FourVector & *v*) [inline]

copy constructor

Definition at line 65 of file SimpleVector.h.

9.10.3 Member Function Documentation

9.10.3.1 double HepMC::FourVector::e () const [inline]

return E

Examples:

testSimpleVector.cc.

Definition at line 73 of file SimpleVector.h.

Referenced by HepMC::GenParticle::convert_momentum(), main(), HepMC::operator<<(), HepMC::GenParticle::print(), repairUnits(), and HepMC::IO_HEPEVT::write_event().

9.10.3.2 double HepMC::FourVector::eta () const

Pseudorapidity (of the space part).

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.3 double HepMC::FourVector::m () const

Invariant mass. If **m2()** (p. 64) is negative then -sqrt(-m2()) is returned.

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.4 double HepMC::FourVector::m2 () const

Invariant mass squared.

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.5 bool HepMC::FourVector::operator!= (const FourVector &) const

inequality

9.10.3.6 FourVector& HepMC::FourVector::operator= (const FourVector &)

make a copy

9.10.3.7 bool HepMC::FourVector::operator== (const FourVector &) const

equality

9.10.3.8 double HepMC::FourVector::perp () const

Transverse component of the spatial vector (R in cylindrical system).

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.9 double HepMC::FourVector::perp2 () const

Transverse component of the spatial vector squared.

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.10 double HepMC::FourVector::phi () const

The azimuth angle.

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.11 double HepMC::FourVector::pseudoRapidity () const

Returns the pseudo-rapidity, i.e. -ln(tan(theta/2)).

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.12 double HepMC::FourVector::px () const [inline]

return px

Examples:

testSimpleVector.cc.

Definition at line 70 of file SimpleVector.h.

Referenced by HepMC::GenParticle::convert_momentum(), main(), HepMC::operator<<(), HepMC::GenParticle::print(), and HepMC::IO_HEPEVT::write_event().

9.10.3.13 double HepMC::FourVector::py () const [inline]

return py

Examples:

testSimpleVector.cc.

Definition at line 71 of file SimpleVector.h.

Referenced by HepMC::GenParticle::convert_momentum(), main(), HepMC::operator<<(), HepMC::GenParticle::print(), repairUnits(), and HepMC::IO_HEPEVT::write_event().

9.10.3.14 double HepMC::FourVector::pz () const [inline]

return pz

Examples:

testSimpleVector.cc.

Definition at line 72 of file SimpleVector.h.

Referenced by HepMC::GenParticle::convert_momentum(), main(), HepMC::operator<<(), HepMC::GenParticle::print(), repairUnits(), and HepMC::IO_HEPEVT::write_event().

9.10.3.15 double HepMC::FourVector::rho () const

spatial vector component magnitude

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.16 void HepMC::FourVector::set (double *x*, double *y*, double *z*, double *t*)

set x, y, z, and t

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.17 void HepMC::FourVector::setE (double *tin*) [inline]

set E

Examples:

testSimpleVector.cc.

Definition at line 110 of file SimpleVector.h.

Referenced by main().

9.10.3.18 void HepMC::FourVector::setPx (double *xin*) [inline]

set px

Examples:

testSimpleVector.cc.

Definition at line 107 of file SimpleVector.h.

Referenced by main().

9.10.3.19 void HepMC::FourVector::setPy (double *yin*) [inline]

set py

Examples:

testSimpleVector.cc.

Definition at line 108 of file SimpleVector.h.

Referenced by main().

9.10.3.20 void HepMC::FourVector::setPz (double *zin*) [inline]

set pz

Examples:

testSimpleVector.cc.

Definition at line 109 of file SimpleVector.h.

Referenced by main().

9.10.3.21 void HepMC::FourVector::setT (double *tin*) [inline]

set t

Examples:

testSimpleVector.cc.

Definition at line 105 of file SimpleVector.h.

Referenced by main().

9.10.3.22 void HepMC::FourVector::setX (double *xin*) [inline]

set x

Examples:

testSimpleVector.cc.

Definition at line 102 of file SimpleVector.h.

Referenced by main().

9.10.3.23 void HepMC::FourVector::setY (double *yin*) [inline]

set y

Examples:

testSimpleVector.cc.

Definition at line 103 of file SimpleVector.h.

Referenced by main().

9.10.3.24 void HepMC::FourVector::setZ (double *zin*) [inline]

set z

Examples:

testSimpleVector.cc.

Definition at line 104 of file SimpleVector.h.

Referenced by main().

9.10.3.25 void HepMC::FourVector::swap (FourVector & *other*)

swap

Referenced by HepMC::GenVertex::swap(), and HepMC::GenParticle::swap().

9.10.3.26 double HepMC::FourVector::t () const [inline]

return t

Examples:

testSimpleVector.cc.

Definition at line 78 of file SimpleVector.h.

Referenced by HepMC::GenVertex::convert_position(), main(), and HepMC::GenVertex::print().

9.10.3.27 double HepMC::FourVector::theta () const

The polar angle.

Examples:

testSimpleVector.cc.

Referenced by main().

9.10.3.28 double HepMC::FourVector::x () const [inline]

return x

Examples:

testSimpleVector.cc.

Definition at line 75 of file SimpleVector.h.

Referenced by HepMC::GenVertex::convert_position(), main(), HepMC::operator<<(), HepMC::GenVertex::point3d(), and HepMC::GenVertex::print().

9.10.3.29 double HepMC::FourVector::y () const [inline]

return y

Examples:

testSimpleVector.cc.

Definition at line 76 of file SimpleVector.h.

Referenced by HepMC::GenVertex::convert_position(), main(), HepMC::GenVertex::point3d(), and HepMC::GenVertex::print().

9.10.3.30 double HepMC::FourVector::z () const [inline]

return z

Examples:

testSimpleVector.cc.

Definition at line 77 of file SimpleVector.h.

Referenced by HepMC::GenVertex::convert_position(), main(), HepMC::GenVertex::point3d(), and HepMC::GenVertex::print().

The documentation for this class was generated from the following file:

- **SimpleVector.h**

9.11 HepMC::GenCrossSection Class Reference

The **GenCrossSection** (p. 71) class stores the generated cross section.

`#include <GenCrossSection.h>`

Public Member Functions

- **GenCrossSection ()**
- **~GenCrossSection ()**
- **GenCrossSection (GenCrossSection const &orig)**

  copy

- **void swap (GenCrossSection &other)**

  swap

- **GenCrossSection & operator= (GenCrossSection const &rhs)**
- **bool operator== (const GenCrossSection &) const**

  check for equality

- **bool operator!= (const GenCrossSection &) const**

  check for inequality

- **double cross_section () const**

  cross section in pb

- **double cross_section_error () const**

  error associated with this cross section in pb

- **bool is_set () const**

  True if the cross section has been set. False by default.

- **void set_cross_section (double xs, double xs_err)**

  Set cross section and error in pb.

- **void set_cross_section (double)**

  set cross section in pb

- **void set_cross_section_error (double)**

  set error associated with this cross section in pb

- **void clear ()**
- **std::ostream & write (std::ostream &) const**

  write to an output stream

- **std::istream & read (std::istream &)**

  read from an input stream

9.11.1 Detailed Description

The **GenCrossSection** (p. 71) class stores the generated cross section.

HepMC::GenCrossSection (p. 71) is used to store the generated cross section. This class is meant to be used to pass, on an event by event basis, the current best guess of the total cross section. It is expected that the final cross section will be stored elsewhere.

- double cross_section; // cross section in pb
- double cross_section_error; // error associated with this cross section

The units of cross_section and cross_section_error are expected to be pb.

GenCrossSection (p. 71) information will be written if **GenEvent** (p. 75) contains a pointer to a valid **GenCrossSection** (p. 71) object.

Examples:

testHepMC.cc.in.

Definition at line 32 of file GenCrossSection.h.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 HepMC::GenCrossSection::GenCrossSection () [inline]

Definition at line 35 of file GenCrossSection.h.

9.11.2.2 HepMC::GenCrossSection::~GenCrossSection () [inline]

Definition at line 40 of file GenCrossSection.h.

9.11.2.3 HepMC::GenCrossSection::GenCrossSection (GenCrossSection const & *orig*)

copy

Definition at line 19 of file GenCrossSection.cc.

9.11.3 Member Function Documentation

9.11.3.1 void HepMC::GenCrossSection::clear ()

Clear all **GenCrossSection** (p. 71) info (disables output of **GenCrossSection** (p. 71) until the cross section is set again)

Definition at line 52 of file GenCrossSection.cc.

9.11.3.2 double HepMC::GenCrossSection::cross_section () const [inline]

cross section in pb

Definition at line 55 of file GenCrossSection.h.

Referenced by operator==(), and HepMC::GenEvent::write_cross_section().

9.11.3.3 double HepMC::GenCrossSection::cross_section_error () const [inline]

error associated with this cross section in pb

Definition at line 57 of file GenCrossSection.h.

Referenced by operator==(), and HepMC::GenEvent::write_cross_section().

9.11.3.4 bool HepMC::GenCrossSection::is_set () const [inline]

True if the cross section has been set. False by default.

Definition at line 60 of file GenCrossSection.h.

Referenced by HepMC::GenEvent::read(), and write().

9.11.3.5 bool HepMC::GenCrossSection::operator!= (const GenCrossSection &) const

check for inequality

Definition at line 46 of file GenCrossSection.cc.

9.11.3.6 GenCrossSection & HepMC::GenCrossSection::operator= (GenCrossSection const & *rhs*)

shallow

Definition at line 32 of file GenCrossSection.cc.

References swap().

9.11.3.7 bool HepMC::GenCrossSection::operator== (const GenCrossSection &) const

check for equality

Definition at line 39 of file GenCrossSection.cc.

References cross_section(), and cross_section_error().

9.11.3.8 std::istream & HepMC::GenCrossSection::read (std::istream &)

read from an input stream

Definition at line 76 of file GenCrossSection.cc.

References set_cross_section().

Referenced by HepMC::operator>>(), and HepMC::GenEvent::read().

9.11.3.9 void HepMC::GenCrossSection::set_cross_section (double) [inline]

set cross section in pb

Definition at line 103 of file GenCrossSection.h.

9.11.3.10 void HepMC::GenCrossSection::set_cross_section (double *xs*, double *xs_err*) [inline]

Set cross section and error in pb.

Examples:

testHepMC.cc.in.

Definition at line 98 of file GenCrossSection.h.

References set_cross_section_error().

Referenced by HepMC::getHerwigCrossSection(), HepMC::getPythiaCrossSection(), and read().

9.11.3.11 void HepMC::GenCrossSection::set_cross_section_error (double) [inline]

set error associated with this cross section in pb

Definition at line 109 of file GenCrossSection.h.

Referenced by set_cross_section().

9.11.3.12 void HepMC::GenCrossSection::swap (GenCrossSection & *other*)

swap

Definition at line 25 of file GenCrossSection.cc.

References m_cross_section, m_cross_section_error, and m_is_set.

Referenced by operator=().

9.11.3.13 std::ostream & HepMC::GenCrossSection::write (std::ostream &) const

write to an output stream

Definition at line 59 of file GenCrossSection.cc.

References is_set().

Referenced by HepMC::operator<<(), and HepMC::GenEvent::write().

The documentation for this class was generated from the following files:

- **GenCrossSection.h**
- **GenCrossSection.cc**

9.12 HepMC::GenEvent Class Reference

The **GenEvent** (p. 75) class is the core of **HepMC** (p. 25).

`#include <GenEvent.h>`

Public Member Functions

- **GenEvent (int signal_process_id=0, int event_number=0, GenVertex ∗signal_vertex=0, const WeightContainer &weights=std::vector< double >(), const std::vector< long > &randomstates=std::vector< long >(), Units::MomentumUnit=Units::default_momentum_unit(), Units::LengthUnit=Units::default_length_unit())**

  default constructor creates null pointers to HeavyIon (p. 154), PdfInfo (p. 222), and GenCrossSection (p. 71)

- **GenEvent (int signal_process_id, int event_number, GenVertex ∗signal_vertex, const WeightContainer &weights, const std::vector< long > &randomstates, const HeavyIon &ion, const PdfInfo &pdf, Units::MomentumUnit=Units::default_momentum_unit(), Units::LengthUnit=Units::default_length_unit())**

  explicit constructor that takes HeavyIon (p. 154) and PdfInfo (p. 222)

- **GenEvent (Units::MomentumUnit, Units::LengthUnit, int signal_process_id=0, int event_number=0, GenVertex ∗signal_vertex=0, const WeightContainer &weights=std::vector< double >(), const std::vector< long > &randomstates=std::vector< long >())**

  constructor requiring units - all else is default

- **GenEvent (Units::MomentumUnit, Units::LengthUnit, int signal_process_id, int event_number, GenVertex ∗signal_vertex, const WeightContainer &weights, const std::vector< long > &randomstates, const HeavyIon &ion, const PdfInfo &pdf)**

  explicit constructor with units first that takes HeavyIon (p. 154) and PdfInfo (p. 222)

- **GenEvent (const GenEvent &inevent)**

  deep copy

- **GenEvent & operator= (const GenEvent &inevent)**

  make a deep copy

- **virtual ∼GenEvent ()**

  deletes all vertices/particles in this evt

- **void swap (GenEvent &other)**

  swap

- **void print (std::ostream &ostr=std::cout) const**

  dumps to ostr

- **void print_version (std::ostream &ostr=std::cout) const**

  dumps release version to ostr

- **GenParticle ∗ barcode_to_particle (int barCode) const**

  assign a barcode to a particle

- **GenVertex ∗ barcode_to_vertex (int barCode) const**
  assign a barcode to a vertex
- **int signal_process_id () const**
  unique signal process id
- **int event_number () const**
  event number
- **int mpi () const**
  number of multi parton interactions
- **double event_scale () const**
  energy scale, see hep-ph/0109068
- **double alphaQCD () const**
  QCD coupling, see hep-ph/0109068.
- **double alphaQED () const**
- **GenVertex ∗ signal_process_vertex () const**
  pointer to the vertex containing the signal process
- **bool valid_beam_particles () const**
  test to see if we have two valid beam particles
- **std::pair< HepMC::GenParticle ∗, HepMC::GenParticle ∗ > beam_particles () const**
  pair of pointers to the two incoming beam particles
- **bool is_valid () const**
- **WeightContainer & weights ()**
  *direct access to WeightContainer (*p. *262)*
- **const WeightContainer & weights () const**
  *direct access to WeightContainer (*p. *262)*
- **GenCrossSection const ∗ cross_section () const**
  *access the GenCrossSection (*p. *71) container if it exists*
- **GenCrossSection ∗ cross_section ()**
- **HeavyIon const ∗ heavy_ion () const**
  *access the HeavyIon (*p. *154) container if it exists*
- **HeavyIon ∗ heavy_ion ()**
- **PdfInfo const ∗ pdf_info () const**
  *access the PdfInfo (*p. *222) container if it exists*
- **PdfInfo ∗ pdf_info ()**
- **const std::vector< long > & random_states () const**
  vector of integers containing information about the random state

- **int particles_size () const**
  how many particle barcodes exist?
- **bool particles_empty () const**
  return true if there are no particle barcodes
- **int vertices_size () const**
  how many vertex barcodes exist?
- **bool vertices_empty () const**
  return true if there are no vertex barcodes
- **void write_units (std::ostream &os=std::cout) const**
- **void write_cross_section (std::ostream &ostr=std::cout) const**
- **Units::MomentumUnit momentum_unit () const**
  Units (p. *41) used by the GenParticle* (p. *113) momentum FourVector* (p. *61).*
- **Units::LengthUnit length_unit () const**
  Units (p. *41) used by the GenVertex* (p. *128) position FourVector* (p. *61).*
- **std::ostream & write (std::ostream &)**
- **std::istream & read (std::istream &)**
- **bool add_vertex (GenVertex ∗vtx)**
  adds to evt and adopts
- **bool remove_vertex (GenVertex ∗vtx)**
  erases vtx from evt
- **void clear ()**
  empties the entire event
- **void set_signal_process_id (int id)**
  set unique signal process id
- **void set_event_number (int eventno)**
  set event number
- **void set_mpi (int)**
  Use this to set the number of multi parton interactions in each event.
- **void set_event_scale (double scale)**
  set energy scale
- **void set_alphaQCD (double a)**
  set QCD coupling
- **void set_alphaQED (double a)**
  set QED coupling

- **void set_signal_process_vertex (GenVertex ∗)**
  set pointer to the vertex containing the signal process
- **bool set_beam_particles (GenParticle ∗, GenParticle ∗)**
  set incoming beam particles
- **bool set_beam_particles (std::pair< HepMC::GenParticle ∗, HepMC::GenParticle ∗ > const &)**
  use a pair of GenParticle∗'s to set incoming beam particles
- **void set_random_states (const std::vector< long > &randomstates)**
  provide random state information
- **void set_cross_section (const GenCrossSection &)**
  provide a pointer to the GenCrossSection (p. *71) container*
- **void set_heavy_ion (const HeavyIon &ion)**
  provide a pointer to the HeavyIon (p. *154) container*
- **void set_pdf_info (const PdfInfo &p)**
  provide a pointer to the PdfInfo (p. *222) container*
- **void use_units (Units::MomentumUnit, Units::LengthUnit)**
- **void use_units (std::string &, std::string &)**
- **void define_units (Units::MomentumUnit, Units::LengthUnit)**
- **void define_units (std::string &, std::string &)**
- **GenEventVertexRange vertex_range ()**
  vertex range
- **ConstGenEventVertexRange vertex_range () const**
  vertex range
- **GenEventParticleRange particle_range ()**
  particle range
- **ConstGenEventParticleRange particle_range () const**
  particle range
- **vertex_const_iterator vertices_begin () const**
  begin vertex iteration
- **vertex_const_iterator vertices_end () const**
  end vertex iteration
- **vertex_iterator vertices_begin ()**
  begin vertex iteration
- **vertex_iterator vertices_end ()**
  end vertex iteration

- **particle_const_iterator particles_begin () const**
  begin particle iteration
- **particle_const_iterator particles_end () const**
  end particle iteration
- **particle_iterator particles_begin ()**
  begin particle iteration
- **particle_iterator particles_end ()**
  end particle iteration

Protected Member Functions

- bool **set_barcode (GenParticle ∗p, int suggested_barcode=false)**
  *set the barcode - intended for use by GenParticle (*p. *113)*
- **bool set_barcode (GenVertex ∗v, int suggested_barcode=false)**
  *set the barcode - intended for use by GenVertex (*p. *128)*
- **void remove_barcode (GenParticle ∗p)**
  *intended for use by GenParticle (*p. *113)*
- **void remove_barcode (GenVertex ∗v)**
  *intended for use by GenVertex (*p. *128)*
- **void delete_all_vertices ()**
  delete all vertices owned by this event

Friends

- class **GenParticle**
- **class GenVertex**
- **class vertex_const_iterator**
- **class vertex_iterator**
- **class particle_const_iterator**
- **class particle_iterator**

Classes

- class **particle_const_iterator**
  const particle iterator
- **class particle_iterator**
  non-const particle iterator
- **class vertex_const_iterator**

const vertex iterator

- **class vertex_iterator**
  non-const vertex iterator

9.12.1 Detailed Description

The **GenEvent** (p. 75) class is the core of **HepMC** (p. 25).

HepMC::GenEvent (p. 75) contains information about generated particles. **GenEvent** (p. 75) is structured as a set of vertices which contain the particles.

Examples:

example_BuildEventFromScratch.cc, example_EventSelection.cc, example_MyPythiaOnlyToHepMC.cc, example_UsingIterators.cc, example_VectorConversion.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, testFlow.cc, testHepMC.cc.in, testHepMCIteration.cc.in, testMass.cc.in, testMultipleCopies.cc.in, testPrintBug.cc, and testStreamIO.cc.in.

Definition at line 155 of file GenEvent.h.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 HepMC::GenEvent::GenEvent (int *signal_process_id* = 0, int *event_number* = 0, GenVertex * *signal_vertex* = 0, const WeightContainer & *weights* = std::vector< double >(), const std::vector< long > & *randomstates* = std::vector< long >(), Units::MomentumUnit = Units::default_momentum_unit(), Units::LengthUnit = Units::default_length_unit())

default constructor creates null pointers to **HeavyIon** (p. 154), **PdfInfo** (p. 222), and **GenCrossSection** (p. 71)

This constructor only allows null pointers to **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

note: default values for m_event_scale, m_alphaQCD, m_alphaQED are as suggested in hep-ph/0109068, "Generic Interface..."

Definition at line 22 of file GenEvent.cc.

9.12.2.2 HepMC::GenEvent::GenEvent (int *signal_process_id*, int *event_number*, GenVertex * *signal_vertex*, const WeightContainer & *weights*, const std::vector< long > & *randomstates*, const HeavyIon & *ion*, const PdfInfo & *pdf*, Units::MomentumUnit = Units::default_momentum_unit(), Units::LengthUnit = Units::default_length_unit())

explicit constructor that takes **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

GenEvent (p. 75) makes its own copy of **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

note: default values for m_event_scale, m_alphaQCD, m_alphaQED are as suggested in hep-ph/0109068, "Generic Interface..."

Definition at line 55 of file GenEvent.cc.

9.12.2.3 HepMC::GenEvent::GenEvent (Units::MomentumUnit, Units::LengthUnit, int *signal_process_id* = 0, int *event_number* = 0, GenVertex * *signal_vertex* = 0, const WeightContainer & *weights* = std::vector< double >(), const std::vector< long > & *randomstates* = std::vector< long >())

constructor requiring units - all else is default

constructor requiring units - all else is default This constructor only allows null pointers to **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

note: default values for m_event_scale, m_alphaQCD, m_alphaQED are as suggested in hep-ph/0109068, "Generic Interface..."

Definition at line 88 of file GenEvent.cc.

9.12.2.4 HepMC::GenEvent::GenEvent (Units::MomentumUnit, Units::LengthUnit, int *signal_process_id*, int *event_number*, GenVertex * *signal_vertex*, const WeightContainer & *weights*, const std::vector< long > & *randomstates*, const HeavyIon & *ion*, const PdfInfo & *pdf*)

explicit constructor with units first that takes **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

explicit constructor with units first that takes **HeavyIon** (p. 154) and **PdfInfo** (p. 222) **GenEvent** (p. 75) makes its own copy of **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

note: default values for m_event_scale, m_alphaQCD, m_alphaQED are as suggested in hep-ph/0109068, "Generic Interface..."

Definition at line 122 of file GenEvent.cc.

9.12.2.5 HepMC::GenEvent::GenEvent (const GenEvent & *inevent*)

deep copy

deep copy - makes a copy of all vertices!

Definition at line 156 of file GenEvent.cc.

References add_vertex(), beam_particles(), GenParticle, GenVertex, p, particles_begin(), particles_end(), random_states(), set_beam_particles(), set_random_states(), set_signal_process_vertex(), signal_process_vertex(), v, vertices_begin(), vertices_end(), and weights().

9.12.2.6 HepMC::GenEvent::~GenEvent () [virtual]

deletes all vertices/particles in this evt

Deep destructor. deletes all vertices/particles in this **GenEvent** (p. 75) deletes the associated **HeavyIon** (p. 154) and **PdfInfo** (p. 222)

Definition at line 258 of file GenEvent.cc.

References delete_all_vertices().

9.12.3 Member Function Documentation

9.12.3.1 bool HepMC::GenEvent::add_vertex (GenVertex * *vtx*)

adds to evt and adopts

returns true if successful - generally will only return false if the inserted vertex is already included in the event.

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, testFlow.cc, and testPrintBug.cc.

Definition at line 334 of file GenEvent.cc.

References HepMC::GenVertex::barcode(), HepMC::GenVertex::parent_event(), remove_vertex(), and HepMC::GenVertex::set_parent_event_().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_HEPEVT::fill_next_event(), GenEvent(), main(), read(), and set_signal_process_vertex().

9.12.3.2 double HepMC::GenEvent::alphaQCD () const [inline]

QCD coupling, see hep-ph/0109068.

Definition at line 690 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.3 double HepMC::GenEvent::alphaQED () const [inline]

QED coupling, see hep-ph/0109068

Definition at line 692 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.4 GenParticle * HepMC::GenEvent::barcode_to_particle (int *barCode*) const [inline]

assign a barcode to a particle

Each vertex or particle has a barcode, which is just an integer which uniquely identifies it inside the event (i.e. there is a one to one mapping between particle memory addresses and particle barcodes... and the same applied for vertices).

The value of a barcode has NO MEANING and NO ORDER! For the user's convenience, when an event is read in via an IO_method from an indexed list (like the HEPEVT common block), then the index will become the barcode for that particle.

Particle barcodes are always positive integers. The barcodes are chosen and set automatically when a vertex or particle comes under the ownership of an event (i.e. it is contained in an event).

Please note that the barcodes are intended for internal use within **HepMC** (p. 25) as a unique identifier for the particles and vertices. Using the barcode to encode extra information is an abuse of the barcode data member and causes confusion among users.

Definition at line 798 of file GenEvent.h.

9.12.3.5 GenVertex * HepMC::GenEvent::barcode_to_vertex (int *barCode*) const [inline]

assign a barcode to a vertex

Each vertex or particle has a barcode, which is just an integer which uniquely identifies it inside the event (i.e. there is a one to one mapping between particle memory addresses and particle barcodes... and the same applied for vertices).

The value of a barcode has NO MEANING and NO ORDER! For the user's convenience, when an event is read in via an IO_method from an indexed list (like the HEPEVT common block), then the index will become the barcode for that particle.

Vertex barcodes are always negative integers. The barcodes are chosen and set automatically when a vertex or particle comes under the ownership of an event (i.e. it is contained in an event).

Please note that the barcodes are intended for internal use within **HepMC** (p. 25) as a unique identifier for the particles and vertices. Using the barcode to encode extra information is an abuse of the barcode data member and causes confusion among users.

Definition at line 823 of file GenEvent.h.

Referenced by HepMC::compareVertices(), and read().

9.12.3.6 std::pair< HepMC::GenParticle *, HepMC::GenParticle * > HepMC::GenEvent::beam_particles () const [inline]

pair of pointers to the two incoming beam particles

Examples:

testMass.cc.in.

Definition at line 844 of file GenEvent.h.

Referenced by HepMC::compareBeamParticles(), filterEvent(), GenEvent(), print(), and write().

9.12.3.7 void HepMC::GenEvent::clear ()

empties the entire event

remove all information from the event deletes all vertices/particles in this evt

Examples:

testHepMCIteration.cc.in, and testStreamIO.cc.in.

Definition at line 365 of file GenEvent.cc.

References HepMC::Units::default_length_unit(), HepMC::Units::default_momentum_unit(), and delete_all_vertices().

Referenced by HepMC::IO_GenEvent::fill_next_event(), and read().

9.12.3.8 GenCrossSection * HepMC::GenEvent::cross_section () [inline]

Definition at line 707 of file GenEvent.h.

9.12.3.9 GenCrossSection const * HepMC::GenEvent::cross_section () const [inline]

access the **GenCrossSection** (p. 71) container if it exists

Examples:

fio/example_PythiaStreamIO.cc, and testHepMC.cc.in.

Definition at line 704 of file GenEvent.h.

Referenced by readPythiaStreamIO(), and write_cross_section().

9.12.3.10 void HepMC::GenEvent::define_units (std::string &, std::string &)

set the units using strings the string must match the enum exactly This method will NOT convert momentum and position data

Definition at line 665 of file GenEvent.cc.

References HepMC::Units::CM, HepMC::Units::GEV, HepMC::Units::MEV, and HepMC::Units::MM.

9.12.3.11 void HepMC::GenEvent::define_units (Units::MomentumUnit, Units::LengthUnit) [inline]

set the units using enums This method will NOT convert momentum and position data

Examples:

testHepMC.cc.in.

Definition at line 866 of file GenEvent.h.

9.12.3.12 void HepMC::GenEvent::delete_all_vertices () [protected]

delete all vertices owned by this event

deletes all vertices in the vertex container (i.e. all vertices owned by this event) The vertices are the "owners" of the particles, so as we delete the vertices, the vertex desctructors are automatically deleting their particles.

Definition at line 403 of file GenEvent.cc.

References particles_empty(), and vertices_empty().

Referenced by clear(), and ~GenEvent().

9.12.3.13 int HepMC::GenEvent::event_number () const [inline]

event number

Examples:

example_EventSelection.cc, fio/example_MyPythia.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, testHepMC.cc.in, testHepMCIteration.cc.in, testMass.cc.in, testMultipleCopies.cc.in, and testStreamIO.cc.in.

Definition at line 682 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), main(), particleTypes(), print(), pythia_in(), pythia_in_out(), write(), HepMC::IO_HEPEVT::write_event(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.14 double HepMC::GenEvent::event_scale () const [inline]

energy scale, see hep-ph/0109068

Definition at line 688 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.15 HeavyIon * HepMC::GenEvent::heavy_ion () [inline]

Definition at line 713 of file GenEvent.h.

9.12.3.16 HeavyIon const * HepMC::GenEvent::heavy_ion () const [inline]

access the **HeavyIon** (p. 154) container if it exists

Definition at line 710 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), and write().

9.12.3.17 bool HepMC::GenEvent::is_valid () const

check **GenEvent** (p. 75) for validity A **GenEvent** (p. 75) is presumed valid if it has particles and/or vertices.

A **GenEvent** (p. 75) is presumed valid if it has both associated particles and vertices. No other information is checked.

Examples:

fio/example_PythiaStreamIO.cc, and testStreamIO.cc.in.

Definition at line 677 of file GenEvent.cc.

References particles_empty(), and vertices_empty().

Referenced by HepMC::IO_GenEvent::fill_next_event(), and readPythiaStreamIO().

9.12.3.18 Units::LengthUnit HepMC::GenEvent::length_unit () const [inline]

Units (p. 41) used by the **GenVertex** (p. 128) position **FourVector** (p. 61).

Definition at line 852 of file GenEvent.h.

Referenced by write(), and write_units().

9.12.3.19 Units::MomentumUnit HepMC::GenEvent::momentum_unit () const [inline]

Units (p. 41) used by the **GenParticle** (p. 113) momentum **FourVector** (p. 61).

Definition at line 849 of file GenEvent.h.

Referenced by write(), and write_units().

9.12.3.20 int HepMC::GenEvent::mpi () const [inline]

number of multi parton interactions

Returns the number of multi parton interactions in the event. This number is -1 if it is not set.

Definition at line 686 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), and write().

9.12.3.21 GenEvent & HepMC::GenEvent::operator= (const GenEvent & *inevent*)

make a deep copy

best practices implementation

Definition at line 269 of file GenEvent.cc.

References swap().

9.12.3.22 ConstGenEventParticleRange HepMC::GenEvent::particle_range () const

particle range

Definition at line 31 of file GenRanges.cc.

9.12.3.23 GenEventParticleRange HepMC::GenEvent::particle_range ()

particle range

Examples:

testHepMCIteration.cc.in.

Definition at line 26 of file GenRanges.cc.

9.12.3.24 particle_iterator HepMC::GenEvent::particles_begin () [inline]

begin particle iteration

Definition at line 565 of file GenEvent.h.

9.12.3.25 particle_const_iterator HepMC::GenEvent::particles_begin () const [inline]

begin particle iteration

Examples:

example_EventSelection.cc, example_UsingIterators.cc, example_VectorConversion.cc, fio/example_MyPythia.cc, testHepMCIteration.cc.in, testMass.cc.in, and testMultipleCopies.cc.in.

Definition at line 507 of file GenEvent.h.

Referenced by HepMC::ConstGenEventParticleRange::begin(), HepMC::GenEventParticleRange::begin(), HepMC::compareParticles(), filterEvent(), findPiZero(), GenEvent(), main(), IsGoodEvent::operator()(), IsGoodEventMyPythia::operator()(), IsEventGood::operator()(), particleTypes(), repairUnits(), valid_beam_particles(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.26 bool HepMC::GenEvent::particles_empty () const [inline]

return true if there are no particle barcodes

Definition at line 833 of file GenEvent.h.

Referenced by delete_all_vertices(), and is_valid().

9.12.3.27 particle_iterator HepMC::GenEvent::particles_end () [inline]

end particle iteration

Definition at line 569 of file GenEvent.h.

9.12.3.28 particle_const_iterator HepMC::GenEvent::particles_end () const [inline]

end particle iteration

Examples:

example_EventSelection.cc, example_UsingIterators.cc, example_VectorConversion.cc, fio/example_MyPythia.cc, testHepMCIteration.cc.in, testMass.cc.in, and testMultipleCopies.cc.in.

Definition at line 511 of file GenEvent.h.

Referenced by HepMC::compareParticles(), HepMC::ConstGenEventParticleRange::end(), HepMC::GenEventParticleRange::end(), filterEvent(), findPiZero(), GenEvent(), main(), IsGoodEvent::operator()(), IsGoodEventMyPythia::operator()(), IsEventGood::operator()(), particleTypes(), repairUnits(), valid_beam_particles(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.29 int HepMC::GenEvent::particles_size () const [inline]

how many particle barcodes exist?

Examples:

testMultipleCopies.cc.in.

Definition at line 830 of file GenEvent.h.

Referenced by HepMC::compareParticles(), particleTypes(), print(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.30 PdfInfo * HepMC::GenEvent::pdf_info () [inline]

Definition at line 719 of file GenEvent.h.

9.12.3.31 PdfInfo const * HepMC::GenEvent::pdf_info () const [inline]

access the **PdfInfo** (p. 222) container if it exists

Definition at line 716 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), and write().

9.12.3.32 void HepMC::GenEvent::print (std::ostream & *ostr* = std::cout) const

dumps to ostr

dumps the content of this event to ostr to dump to cout use: event.print(); if you want to write this event to file outfile.txt you could use: std::ofstream outfile("outfile.txt"); event.print(outfile);

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, fio/example_MyHerwig.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, testFlow.cc, testHepMC.cc.in, testMultipleCopies.cc.in, and testPrintBug.cc.

Definition at line 277 of file GenEvent.cc.

References alphaQCD(), alphaQED(), HepMC::GenVertex::barcode(), beam_particles(), event_number(), event_scale(), particles_size(), HepMC::WeightContainer::print(), signal_process_id(), signal_process_vertex(), HepMC::WeightContainer::size(), vertices_end(), vertices_size(), weights(), write_cross_section(), and write_units().

Referenced by main().

9.12.3.33 void HepMC::GenEvent::print_version (std::ostream & *ostr* = std::cout) const

dumps release version to ostr

Definition at line 328 of file GenEvent.cc.

References HepMC::writeVersion().

9.12.3.34 const std::vector< long > & HepMC::GenEvent::random_states () const [inline]

vector of integers containing information about the random state

Vector of integers which specify the random number generator's state for this event. It is left to the generator to make use of this. We envision a vector of RndmStatesTags to be included with a run class which would specify the meaning of the random_states.

Definition at line 727 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), GenEvent(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.35 std::istream & HepMC::GenEvent::read (std::istream &)

Examples:

fio/example_PythiaStreamIO.cc, and testStreamIO.cc.in.

Definition at line 155 of file GenEventStreamIO.cc.

References HepMC::GenVertex::add_particle_in(), add_vertex(), barcode_to_vertex(), clear(), HepMC::extascii, HepMC::detail::find_event_end(), HepMC::StreamInfo::finished_first_event(), HepMC::gen, GenVertex, HepMC::get_stream_info(), HepMC::StreamInfo::has_key(), HepMC::StreamInfo::io_momentum_unit(), HepMC::StreamInfo::io_position_unit(), HepMC::StreamInfo::io_type(), HepMC::GenCrossSection::is_set(), p, HepMC::GenCrossSection::read(), HepMC::detail::read_vertex(), HepMC::StreamInfo::reading_event_header(), set_beam_particles(), set_cross_section(), HepMC::StreamInfo::set_finished_first_event(), set_heavy_ion(), set_pdf_info(), HepMC::StreamInfo::set_reading_event_header(), set_signal_process_vertex(), signal_process_vertex(), use_units(), and v.

Referenced by HepMC::operator>>(), and readPythiaStreamIO().

9.12.3.36 void HepMC::GenEvent::remove_barcode (GenVertex * *v*) [inline, protected]

intended for use by **GenVertex** (p. 128)

Definition at line 777 of file GenEvent.h.

References v.

9.12.3.37 void HepMC::GenEvent::remove_barcode (GenParticle * *p*) [inline, protected]

intended for use by **GenParticle** (p. 113)

Definition at line 774 of file GenEvent.h.

References p.

Referenced by HepMC::GenParticle::set_end_vertex_(), HepMC::GenVertex::set_parent_event_(), HepMC::GenParticle::set_production_vertex_(), HepMC::GenParticle::~GenParticle(), and HepMC::GenVertex::~GenVertex().

9.12.3.38 bool HepMC::GenEvent::remove_vertex (GenVertex * *vtx*)

erases vtx from evt

this removes vtx from the event but does NOT delete it. returns True if an entry vtx existed in the table and was erased

Definition at line 357 of file GenEvent.cc.

References HepMC::GenVertex::barcode(), HepMC::GenVertex::parent_event(), and HepMC::GenVertex::set_parent_event_().

Referenced by add_vertex().

9.12.3.39 void HepMC::GenEvent::set_alphaQCD (double *a*) [inline]

set QCD coupling

Definition at line 743 of file GenEvent.h.

9.12.3.40 void HepMC::GenEvent::set_alphaQED (double *a*) [inline]

set QED coupling

Definition at line 745 of file GenEvent.h.

9.12.3.41 bool HepMC::GenEvent::set_barcode (GenVertex * *v*, int *suggested_barcode* = false) [protected]

set the barcode - intended for use by **GenVertex** (p. 128)

Definition at line 501 of file GenEvent.cc.

References v.

9.12.3.42 bool HepMC::GenEvent::set_barcode (GenParticle * *p*, int *suggested_barcode* = false) [protected]

set the barcode - intended for use by **GenParticle** (p. 113)

Definition at line 430 of file GenEvent.cc.

References p.

Referenced by HepMC::GenVertex::set_parent_event_(), HepMC::GenVertex::suggest_barcode(), and HepMC::GenParticle::suggest_barcode().

9.12.3.43 bool HepMC::GenEvent::set_beam_particles (std::pair< HepMC::GenParticle *, HepMC::GenParticle * > const & *bp*)

use a pair of GenParticle*'s to set incoming beam particles

construct the beam particle information using a std::pair of pointers to **GenParticle** (p. 113) returns false if either GenParticle* is null

Definition at line 595 of file GenEvent.cc.

References set_beam_particles().

9.12.3.44 bool HepMC::GenEvent::set_beam_particles (GenParticle * *bp1*, GenParticle * *bp2*)

set incoming beam particles

construct the beam particle information using pointers to **GenParticle** (p. 113) returns false if either GenParticle* is null

Definition at line 586 of file GenEvent.cc.

Referenced by HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_HEPEVT::fill_next_event(), GenEvent(), read(), and set_beam_particles().

9.12.3.45 void HepMC::GenEvent::set_cross_section (const GenCrossSection &) [inline]

provide a pointer to the **GenCrossSection** (p. 71) container

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, and testHepMC.cc.in.

Definition at line 752 of file GenEvent.h.

Referenced by event_selection(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), read(), and writePythiaStreamIO().

9.12.3.46 void HepMC::GenEvent::set_event_number (int *eventno*) [inline]

set event number

Examples:

fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, and fio/testHerwigCopies.cc.

Definition at line 733 of file GenEvent.h.

Referenced by HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_HEPEVT::fill_next_event(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), and writePythiaStreamIO().

9.12.3.47 void HepMC::GenEvent::set_event_scale (double *scale*) [inline]

set energy scale

Definition at line 741 of file GenEvent.h.

9.12.3.48 void HepMC::GenEvent::set_heavy_ion (const HeavyIon & *ion*) [inline]

provide a pointer to the **HeavyIon** (p. 154) container

Examples:

testMass.cc.in.

Definition at line 758 of file GenEvent.h.

Referenced by read().

9.12.3.49 void HepMC::GenEvent::set_mpi (int) [inline]

Use this to set the number of multi parton interactions in each event.

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, and fio/testPythiaCopies.cc.

Definition at line 737 of file GenEvent.h.

Referenced by event_selection(), main(), pythia_out(), and writePythiaStreamIO().

9.12.3.50 void HepMC::GenEvent::set_pdf_info (const PdfInfo & *p*) [inline]

provide a pointer to the **PdfInfo** (p. 222) container

Examples:

testMass.cc.in.

Definition at line 764 of file GenEvent.h.

References p.

Referenced by read().

9.12.3.51 void HepMC::GenEvent::set_random_states (const std::vector< long > & *randomstates*) [inline]

provide random state information

Definition at line 770 of file GenEvent.h.

Referenced by GenEvent().

9.12.3.52 void HepMC::GenEvent::set_signal_process_id (int *id*) [inline]

set unique signal process id

Examples:

fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, and fio/testHerwigCopies.cc.

Definition at line 730 of file GenEvent.h.

Referenced by main(), pythia_in_out(), pythia_out(), pythia_particle_out(), and writePythiaStreamIO().

9.12.3.53 void HepMC::GenEvent::set_signal_process_vertex (GenVertex *) [inline]

set pointer to the vertex containing the signal process

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, and testFlow.cc.

Definition at line 747 of file GenEvent.h.

References add_vertex().

Referenced by HepMC::IO_HERWIG::fill_next_event(), GenEvent(), main(), and read().

9.12.3.54 int HepMC::GenEvent::signal_process_id () const [inline]

unique signal process id

The integer ID that uniquely specifies this signal process, i.e. MSUB in Pythia. It is necessary to package this with each event rather than with the run because many processes may be generated within one run.

Definition at line 679 of file GenEvent.h.

Referenced by HepMC::compareGenEvent(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.55 GenVertex * HepMC::GenEvent::signal_process_vertex () const [inline]

pointer to the vertex containing the signal process

returns a (mutable) pointer to the signal process vertex

Definition at line 694 of file GenEvent.h.

Referenced by HepMC::compareSignalProcessVertex(), GenEvent(), print(), read(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.56 void HepMC::GenEvent::swap (GenEvent & *other*)

swap

Definition at line 226 of file GenEvent.cc.

References m_alphaQCD, m_alphaQED, m_beam_particle_1, m_beam_particle_2, m_cross_section, m_event_number, m_event_scale, m_heavy_ion, m_momentum_unit, m_mpi, m_particle_barcodes, m_pdf_info, m_position_unit, m_random_states, m_signal_process_id, m_signal_process_vertex, m_vertex_barcodes, m_weights, HepMC::WeightContainer::swap(), vertices_begin(), and vertices_end().

Referenced by operator=().

9.12.3.57 void HepMC::GenEvent::use_units (std::string &, std::string &) [inline]

set the units using strings the string must match the enum exactly This method will convert momentum and position data if necessary

Definition at line 861 of file GenEvent.h.

9.12.3.58 void HepMC::GenEvent::use_units (Units::MomentumUnit, Units::LengthUnit) [inline]

set the units using enums This method will convert momentum and position data if necessary

Examples:

example_BuildEventFromScratch.cc, example_MyPythiaOnlyToHepMC.cc, example_VectorConversion.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, testFlow.cc, and testPrintBug.cc.

Definition at line 856 of file GenEvent.h.

Referenced by HepMC::convert_units(), event_selection(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), read(), and writePythiaStreamIO().

9.12.3.59 bool HepMC::GenEvent::valid_beam_particles () const

test to see if we have two valid beam particles

Examples:

testMass.cc.in.

Definition at line 568 of file GenEvent.cc.

References p, particles_begin(), and particles_end().

9.12.3.60 ConstGenEventVertexRange HepMC::GenEvent::vertex_range () const

vertex range

Definition at line 21 of file GenRanges.cc.

9.12.3.61 GenEventVertexRange HepMC::GenEvent::vertex_range ()

vertex range

Examples:

testHepMCIteration.cc.in.

Definition at line 16 of file GenRanges.cc.

9.12.3.62 vertex_iterator HepMC::GenEvent::vertices_begin () [inline]

begin vertex iteration

Definition at line 440 of file GenEvent.h.

9.12.3.63 vertex_const_iterator HepMC::GenEvent::vertices_begin () const [inline]

begin vertex iteration

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 377 of file GenEvent.h.

Referenced by HepMC::ConstGenEventVertexRange::begin(), HepMC::GenEventVertexRange::begin(), HepMC::compareVertices(), filterEvent(), GenEvent(), main(), swap(), write(), and HepMC::IO_-HEPEVT::write_event().

9.12.3.64 bool HepMC::GenEvent::vertices_empty () const [inline]

return true if there are no vertex barcodes

Definition at line 839 of file GenEvent.h.

Referenced by delete_all_vertices(), and is_valid().

9.12.3.65 vertex_iterator HepMC::GenEvent::vertices_end () [inline]

end vertex iteration

Definition at line 444 of file GenEvent.h.

9.12.3.66 vertex_const_iterator HepMC::GenEvent::vertices_end () const [inline]

end vertex iteration

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 381 of file GenEvent.h.

Referenced by HepMC::compareVertices(), HepMC::ConstGenEventVertexRange::end(), HepMC::GenEventVertexRange::end(), filterEvent(), GenEvent(), main(), print(), swap(), write(), and HepMC::IO_HEPEVT::write_event().

9.12.3.67 int HepMC::GenEvent::vertices_size () const [inline]

how many vertex barcodes exist?

Examples:

testMultipleCopies.cc.in.

Definition at line 836 of file GenEvent.h.

Referenced by HepMC::compareVertices(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.68 const WeightContainer & HepMC::GenEvent::weights () const [inline]

direct access to **WeightContainer** (p. 262)

Definition at line 701 of file GenEvent.h.

9.12.3.69 WeightContainer & HepMC::GenEvent::weights () [inline]

direct access to **WeightContainer** (p. 262)

direct access to the weights container is allowed. Thus you can use myevt.weights()[2]; to access element 2 of the weights. or use myevt.weights().push_back(mywgt); to add an element. and you can set the weights with myevt.weights() = myvector;

Examples:

fio/testPythiaCopies.cc, testHepMC.cc.in, and testMass.cc.in.

Definition at line 699 of file GenEvent.h.

Referenced by HepMC::compareWeights(), GenEvent(), main(), print(), write(), and HepMC::IO_AsciiParticles::write_event().

9.12.3.70 std::ostream & HepMC::GenEvent::write (std::ostream &)

Examples:

fio/example_PythiaStreamIO.cc, testFlow.cc, and testStreamIO.cc.in.

Definition at line 72 of file GenEventStreamIO.cc.

References alphaQCD(), alphaQED(), beam_particles(), event_number(), event_scale(), HepMC::StreamInfo::finished_first_event(), HepMC::get_stream_info(), heavy_ion(), length_unit(), HepMC::WeightContainer::map_end(), momentum_unit(), mpi(), HepMC::Units::name(), HepMC::detail::output(), pdf_info(), HepMC::StreamInfo::set_finished_first_event(), signal_process_id(), signal_process_vertex(), HepMC::WeightContainer::size(), v, vertices_begin(), vertices_end(), vertices_size(), weights(), and HepMC::GenCrossSection::write().

Referenced by main(), HepMC::operator<<(), and readPythiaStreamIO().

9.12.3.71 void HepMC::GenEvent::write_cross_section (std::ostream & *ostr* = `std::cout`) const

If the cross section is defined, write the cross section information to an output stream. If the output stream is not defined, use std::cout.

Examples:

testHepMC.cc.in.

Definition at line 605 of file GenEvent.cc.

References HepMC::GenCrossSection::cross_section(), cross_section(), and HepMC::GenCrossSection::cross_section_error().

Referenced by print().

9.12.3.72 void HepMC::GenEvent::write_units (std::ostream & *os* = `std::cout`) const

Write the unit information to an output stream. If the output stream is not defined, use std::cout.

Examples:

testHepMC.cc.in, and testStreamIO.cc.in.

Definition at line 599 of file GenEvent.cc.

References length_unit(), momentum_unit(), and HepMC::Units::name().

Referenced by print().

9.12.4 Friends And Related Function Documentation

9.12.4.1 friend class GenParticle `[friend]`

Definition at line 156 of file GenEvent.h.

Referenced by GenEvent().

9.12.4.2 friend class GenVertex [friend]

Definition at line 157 of file GenEvent.h.

Referenced by GenEvent(), and read().

9.12.4.3 friend class particle_const_iterator [friend]

Definition at line 505 of file GenEvent.h.

Referenced by HepMC::GenEvent::particle_iterator::operator particle_const_iterator().

9.12.4.4 friend class particle_iterator [friend]

Definition at line 563 of file GenEvent.h.

9.12.4.5 friend class vertex_const_iterator [friend]

Definition at line 375 of file GenEvent.h.

Referenced by HepMC::GenEvent::vertex_iterator::operator vertex_const_iterator().

9.12.4.6 friend class vertex_iterator [friend]

Definition at line 438 of file GenEvent.h.

The documentation for this class was generated from the following files:

- **GenEvent.h**
- **GenEvent.cc**
- **GenEventStreamIO.cc**
- **GenRanges.cc**

9.13 HepMC::GenEvent::particle_const_iterator Class Reference

const particle iterator

`#include <GenEvent.h>`

Public Member Functions

- **particle_const_iterator (const std::map< int, HepMC::GenParticle * >::const_iterator &i)**
  iterate over particles
- **particle_const_iterator ()**
- **particle_const_iterator (const particle_const_iterator &i)**
  copy constructor
- **virtual ~particle_const_iterator ()**
- **particle_const_iterator & operator= (const particle_const_iterator &i)**
  make a copy
- **GenParticle * operator * (void) const**
  return a pointer to GenParticle (p. *113)*
- **particle_const_iterator & operator++ (void)**
  Pre-fix increment.
- **particle_const_iterator operator++ (int)**
  Post-fix increment.
- **bool operator== (const particle_const_iterator &a) const**
  equality
- **bool operator!= (const particle_const_iterator &a) const**
  inequality

Protected Attributes

- std::map< int, **HepMC::GenParticle * >::const_iterator m_map_iterator**
  *const iterator to the GenParticle (*p. *113) map*

9.13.1 Detailed Description

const particle iterator

HepMC::GenEvent::particle_const_iterator (p. 98) is used to iterate over all particles in the event.

Examples:

example_EventSelection.cc, example_VectorConversion.cc, fio/example_MyPythia.cc, testMass.cc.in, and testMultipleCopies.cc.in.

Definition at line 464 of file GenEvent.h.

9.13.2 Constructor & Destructor Documentation

9.13.2.1 HepMC::GenEvent::particle_const_iterator::particle_const_iterator (const std::map< int, HepMC::GenParticle * >::const_iterator & *i*) [inline]

iterate over particles

Definition at line 469 of file GenEvent.h.

9.13.2.2 HepMC::GenEvent::particle_const_iterator::particle_const_iterator () [inline]

Definition at line 472 of file GenEvent.h.

9.13.2.3 HepMC::GenEvent::particle_const_iterator::particle_const_iterator (const particle_const_iterator & *i*) [inline]

copy constructor

Definition at line 474 of file GenEvent.h.

9.13.2.4 virtual HepMC::GenEvent::particle_const_iterator::~particle_const_iterator () [inline, virtual]

Definition at line 476 of file GenEvent.h.

9.13.3 Member Function Documentation

9.13.3.1 GenParticle* HepMC::GenEvent::particle_const_iterator::operator * (void) const [inline]

return a pointer to **GenParticle** (p. 113)

Definition at line 482 of file GenEvent.h.

References m_map_iterator.

9.13.3.2 bool HepMC::GenEvent::particle_const_iterator::operator!= (const particle_const_iterator & *a*) const [inline]

inequality

Definition at line 494 of file GenEvent.h.

References m_map_iterator.

9.13.3.3 particle_const_iterator HepMC::GenEvent::particle_const_iterator::operator++ (int) [inline]

Post-fix increment.

Definition at line 488 of file GenEvent.h.

9.13.3.4 particle_const_iterator& HepMC::GenEvent::particle_const_iterator::operator++ (void) [inline]

Pre-fix increment.

Definition at line 485 of file GenEvent.h.

References m_map_iterator.

9.13.3.5 particle_const_iterator& HepMC::GenEvent::particle_const_iterator::operator= (const particle_const_iterator & *i*) [inline]

make a copy

Definition at line 478 of file GenEvent.h.

References m_map_iterator.

9.13.3.6 bool HepMC::GenEvent::particle_const_iterator::operator== (const particle_const_iterator & *a*) const [inline]

equality

Definition at line 491 of file GenEvent.h.

References m_map_iterator.

9.13.4 Member Data Documentation

9.13.4.1 std::map<int,HepMC::GenParticle*>::const_iterator HepMC::GenEvent::particle_const_iterator::m_map_iterator [protected]

const iterator to the **GenParticle** (p. 113) map

Definition at line 498 of file GenEvent.h.

Referenced by operator *(), operator!=(), operator++(), operator=(), and operator==().

The documentation for this class was generated from the following file:

- **GenEvent.h**

9.14 HepMC::GenEvent::particle_iterator Class Reference

non-const particle iterator

`#include <GenEvent.h>`

Public Member Functions

- **particle_iterator (const std::map< int, HepMC::GenParticle * >::iterator &i)**
  iterate over particles
- **particle_iterator ()**
- **particle_iterator (const particle_iterator &i)**
  copy constructor
- **virtual ~particle_iterator ()**
- **particle_iterator & operator= (const particle_iterator &i)**
  make a copy
- **operator particle_const_iterator () const**
  const particle iterator
- **GenParticle * operator * (void) const**
  *return pointer to GenParticle (*p. *113)*
- **particle_iterator & operator++ (void)**
  Pre-fix increment.
- **particle_iterator operator++ (int)**
  Post-fix increment.
- **bool operator== (const particle_iterator &a) const**
  equality
- **bool operator!= (const particle_iterator &a) const**
  inequality

Protected Attributes

- std::map< int, **HepMC::GenParticle * >::iterator m_map_iterator**
  *iterator for GenParticle (*p. *113) map*

9.14.1 Detailed Description

non-const particle iterator

HepMC::GenEvent::particle_iterator (p. 101) is used to iterate over all particles in the event.

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 520 of file GenEvent.h.

9.14.2 Constructor & Destructor Documentation

9.14.2.1 HepMC::GenEvent::particle_iterator::particle_iterator (const std::map< int, HepMC::GenParticle * >::iterator & *i*) [inline]

iterate over particles

Definition at line 525 of file GenEvent.h.

9.14.2.2 HepMC::GenEvent::particle_iterator::particle_iterator () [inline]

Definition at line 527 of file GenEvent.h.

9.14.2.3 HepMC::GenEvent::particle_iterator::particle_iterator (const particle_iterator & *i*) [inline]

copy constructor

Definition at line 529 of file GenEvent.h.

9.14.2.4 virtual HepMC::GenEvent::particle_iterator::~particle_iterator () [inline, virtual]

Definition at line 530 of file GenEvent.h.

9.14.3 Member Function Documentation

9.14.3.1 GenParticle* HepMC::GenEvent::particle_iterator::operator * (void) const [inline]

return pointer to **GenParticle** (p. 113)

Definition at line 540 of file GenEvent.h.

References m_map_iterator.

9.14.3.2 HepMC::GenEvent::particle_iterator::operator particle_const_iterator () const [inline]

const particle iterator

Definition at line 537 of file GenEvent.h.

References m_map_iterator, and HepMC::GenEvent::particle_const_iterator.

9.14.3.3 bool HepMC::GenEvent::particle_iterator::operator!= (const particle_iterator & *a*) const [inline]

inequality

Definition at line 552 of file GenEvent.h.

References m_map_iterator.

9.14.3.4 particle_iterator HepMC::GenEvent::particle_iterator::operator++ (int) [inline]

Post-fix increment.

Definition at line 546 of file GenEvent.h.

9.14.3.5 particle_iterator& HepMC::GenEvent::particle_iterator::operator++ (void) [inline]

Pre-fix increment.

Definition at line 543 of file GenEvent.h.

References m_map_iterator.

9.14.3.6 particle_iterator& HepMC::GenEvent::particle_iterator::operator= (const particle_iterator & *i*) [inline]

make a copy

Definition at line 532 of file GenEvent.h.

References m_map_iterator.

9.14.3.7 bool HepMC::GenEvent::particle_iterator::operator== (const particle_iterator & *a*) const [inline]

equality

Definition at line 549 of file GenEvent.h.

References m_map_iterator.

9.14.4 Member Data Documentation

9.14.4.1 std::map<int,HepMC::GenParticle*>::iterator HepMC::GenEvent::particle_iterator::m_map_iterator [protected]

iterator for **GenParticle** (p. 113) map

Definition at line 556 of file GenEvent.h.

Referenced by operator *(), operator particle_const_iterator(), operator!=(), operator++(), operator=(), and operator==().

The documentation for this class was generated from the following file:

- **GenEvent.h**

9.15 HepMC::GenEvent::vertex_const_iterator Class Reference

const vertex iterator

`#include <GenEvent.h>`

Public Member Functions

- **vertex_const_iterator (const std::map< int, HepMC::GenVertex *, std::greater< int > >::const_iterator &i)**
  constructor requiring vertex information
- **vertex_const_iterator ()**
- **vertex_const_iterator (const vertex_const_iterator &i)**
  copy constructor
- **virtual ~vertex_const_iterator ()**
- **vertex_const_iterator & operator= (const vertex_const_iterator &i)**
  make a copy
- **GenVertex * operator * (void) const**
  return a pointer to a GenVertex (p. 128)
- **vertex_const_iterator & operator++ (void)**
  Pre-fix increment.
- **vertex_const_iterator operator++ (int)**
  Post-fix increment.
- **bool operator== (const vertex_const_iterator &a) const**
  equality
- **bool operator!= (const vertex_const_iterator &a) const**
  inequality

Protected Attributes

- std::map< int, **HepMC::GenVertex *, std::greater< int > >::const_iterator m_map_iterator**
  const iterator to a vertex map

9.15.1 Detailed Description

const vertex iterator

HepMC::GenEvent::vertex_const_iterator (p. 104) is used to iterate over all vertices in the event.

Definition at line 334 of file GenEvent.h.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 HepMC::GenEvent::vertex_const_iterator::vertex_const_iterator (const std::map< int, HepMC::GenVertex *, std::greater< int > >::const_iterator & *i*) [inline]

constructor requiring vertex information

Definition at line 339 of file GenEvent.h.

9.15.2.2 HepMC::GenEvent::vertex_const_iterator::vertex_const_iterator () [inline]

Definition at line 343 of file GenEvent.h.

9.15.2.3 HepMC::GenEvent::vertex_const_iterator::vertex_const_iterator (const vertex_const_iterator & *i*) [inline]

copy constructor

Definition at line 345 of file GenEvent.h.

9.15.2.4 virtual HepMC::GenEvent::vertex_const_iterator::~vertex_const_iterator () [inline, virtual]

Definition at line 347 of file GenEvent.h.

9.15.3 Member Function Documentation

9.15.3.1 GenVertex* HepMC::GenEvent::vertex_const_iterator::operator * (void) const [inline]

return a pointer to a **GenVertex** (p. 128)

Definition at line 352 of file GenEvent.h.

References m_map_iterator.

9.15.3.2 bool HepMC::GenEvent::vertex_const_iterator::operator!= (const vertex_const_iterator & *a*) const [inline]

inequality

Definition at line 363 of file GenEvent.h.

References m_map_iterator.

9.15.3.3 vertex_const_iterator HepMC::GenEvent::vertex_const_iterator::operator++ (int) [inline]

Post-fix increment.

Definition at line 357 of file GenEvent.h.

9.15.3.4 vertex_const_iterator& HepMC::GenEvent::vertex_const_iterator::operator++ (void) [inline]

Pre-fix increment.

Definition at line 354 of file GenEvent.h.

References m_map_iterator.

9.15.3.5 vertex_const_iterator& HepMC::GenEvent::vertex_const_iterator::operator= (const vertex_const_iterator & *i*) [inline]

make a copy

Definition at line 349 of file GenEvent.h.

References m_map_iterator.

9.15.3.6 bool HepMC::GenEvent::vertex_const_iterator::operator== (const vertex_const_iterator & *a*) const [inline]

equality

Definition at line 360 of file GenEvent.h.

References m_map_iterator.

9.15.4 Member Data Documentation

9.15.4.1 std::map<int,HepMC::GenVertex*,std::greater<int> >::const_iterator HepMC::GenEvent::vertex_const_iterator::m_map_iterator [protected]

const iterator to a vertex map

Definition at line 368 of file GenEvent.h.

Referenced by operator *(), operator!=(), operator++(), operator=(), and operator==().

The documentation for this class was generated from the following file:

- **GenEvent.h**

9.16 HepMC::GenEvent::vertex_iterator Class Reference

non-const vertex iterator

`#include <GenEvent.h>`

Public Member Functions

- **vertex_iterator (const std::map< int, HepMC::GenVertex ∗, std::greater< int > >::iterator &i)**

  constructor requiring vertex information

- **vertex_iterator ()**
- **vertex_iterator (const vertex_iterator &i)**

  copy constructor

- **virtual ∼vertex_iterator ()**
- **vertex_iterator & operator= (const vertex_iterator &i)**

  make a copy

- **operator vertex_const_iterator () const**

  const vertex iterator

- **GenVertex ∗ operator ∗ (void) const**

  return a pointer to a GenVertex (p. *128)*

- **vertex_iterator & operator++ (void)**

  Pre-fix increment.

- **vertex_iterator operator++ (int)**

  Post-fix increment.

- **bool operator== (const vertex_iterator &a) const**

  equality

- **bool operator!= (const vertex_iterator &a) const**

  inequality

Protected Attributes

- std::map< int, **HepMC::GenVertex ∗, std::greater< int > >::iterator m_map_iterator**

  iterator to the vertex map

9.16.1 Detailed Description

non-const vertex iterator

HepMC::GenEvent::vertex_iterator (p. 107) is used to iterate over all vertices in the event.

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 391 of file GenEvent.h.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 HepMC::GenEvent::vertex_iterator::vertex_iterator (const std::map< int, HepMC::GenVertex *, std::greater< int > >::iterator & *i*) [inline]

constructor requiring vertex information

Definition at line 396 of file GenEvent.h.

9.16.2.2 HepMC::GenEvent::vertex_iterator::vertex_iterator () [inline]

Definition at line 400 of file GenEvent.h.

9.16.2.3 HepMC::GenEvent::vertex_iterator::vertex_iterator (const vertex_iterator & *i*) [inline]

copy constructor

Definition at line 402 of file GenEvent.h.

9.16.2.4 virtual HepMC::GenEvent::vertex_iterator::~vertex_iterator () [inline, virtual]

Definition at line 403 of file GenEvent.h.

9.16.3 Member Function Documentation

9.16.3.1 GenVertex* HepMC::GenEvent::vertex_iterator::operator * (void) const [inline]

return a pointer to a **GenVertex** (p. 128)

Definition at line 413 of file GenEvent.h.

References m_map_iterator.

9.16.3.2 HepMC::GenEvent::vertex_iterator::operator vertex_const_iterator () const [inline]

const vertex iterator

Definition at line 410 of file GenEvent.h.

References m_map_iterator, and HepMC::GenEvent::vertex_const_iterator.

9.16.3.3 bool HepMC::GenEvent::vertex_iterator::operator!= (const vertex_iterator & *a*) const [inline]

inequality

Definition at line 425 of file GenEvent.h.

References m_map_iterator.

9.16.3.4 vertex_iterator HepMC::GenEvent::vertex_iterator::operator++ (int) [inline]

Post-fix increment.

Definition at line 419 of file GenEvent.h.

9.16.3.5 vertex_iterator& HepMC::GenEvent::vertex_iterator::operator++ (void) [inline]

Pre-fix increment.

Definition at line 416 of file GenEvent.h.

References m_map_iterator.

9.16.3.6 vertex_iterator& HepMC::GenEvent::vertex_iterator::operator= (const vertex_iterator & *i*) [inline]

make a copy

Definition at line 405 of file GenEvent.h.

References m_map_iterator.

9.16.3.7 bool HepMC::GenEvent::vertex_iterator::operator== (const vertex_iterator & *a*) const [inline]

equality

Definition at line 422 of file GenEvent.h.

References m_map_iterator.

9.16.4 Member Data Documentation

9.16.4.1 std::map<int,HepMC::GenVertex*,std::greater<int> >::iterator HepMC::GenEvent::vertex_iterator::m_map_iterator [protected]

iterator to the vertex map

Definition at line 430 of file GenEvent.h.

Referenced by operator *(), operator vertex_const_iterator(), operator!=(), operator++(), operator=(), and operator==().

The documentation for this class was generated from the following file:

- **GenEvent.h**

9.17 HepMC::GenEventParticleRange Class Reference

GenEventParticleRange (p. 111) acts like a collection of particles.

`#include <GenRanges.h>`

Public Member Functions

- **GenEventParticleRange (GenEvent &e)**
  the constructor requires a GenEvent (p. *75)*
- **GenEvent::particle_iterator begin ()**
- **GenEvent::particle_iterator end ()**

9.17.1 Detailed Description

GenEventParticleRange (p. 111) acts like a collection of particles.

HepMC::GenEventParticleRange (p. 111) is used to mimic a collection of particles for ease of use - especially with utilities such as the Boost foreach funtion

Examples:

testHepMCIteration.cc.in.

Definition at line 83 of file GenRanges.h.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 HepMC::GenEventParticleRange::GenEventParticleRange (GenEvent & *e*) [inline]

the constructor requires a **GenEvent** (p. 75)

Definition at line 88 of file GenRanges.h.

9.17.3 Member Function Documentation

9.17.3.1 GenEvent::particle_iterator HepMC::GenEventParticleRange::begin () [inline]

Definition at line 90 of file GenRanges.h.

References HepMC::GenEvent::particles_begin().

9.17.3.2 GenEvent::particle_iterator HepMC::GenEventParticleRange::end () [inline]

Definition at line 91 of file GenRanges.h.

References HepMC::GenEvent::particles_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.18 HepMC::GenEventVertexRange Class Reference

GenEventVertexRange (p. 112) acts like a collection of vertices.

`#include <GenRanges.h>`

Public Member Functions

- **GenEventVertexRange (GenEvent &e)**
  the constructor requires a GenEvent (p. *75)*
- **GenEvent::vertex_iterator begin ()**
- **GenEvent::vertex_iterator end ()**

9.18.1 Detailed Description

GenEventVertexRange (p. 112) acts like a collection of vertices.

HepMC::GenEventVertexRange (p. 112) is used to mimic a collection of vertices for ease of use - especially with utilities such as the Boost foreach funtion

Examples:

testHepMCIteration.cc.in.

Definition at line 26 of file GenRanges.h.

9.18.2 Constructor & Destructor Documentation

9.18.2.1 HepMC::GenEventVertexRange::GenEventVertexRange (GenEvent & *e*) [inline]

the constructor requires a **GenEvent** (p. 75)

Definition at line 31 of file GenRanges.h.

9.18.3 Member Function Documentation

9.18.3.1 GenEvent::vertex_iterator HepMC::GenEventVertexRange::begin () [inline]

Definition at line 33 of file GenRanges.h.

References HepMC::GenEvent::vertices_begin().

9.18.3.2 GenEvent::vertex_iterator HepMC::GenEventVertexRange::end () [inline]

Definition at line 34 of file GenRanges.h.

References HepMC::GenEvent::vertices_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.19 HepMC::GenParticle Class Reference

The **GenParticle** (p. 113) class contains information about generated particles.

`#include <GenParticle.h>`

Public Member Functions

- **GenParticle (void)**
  default constructor
- **GenParticle (const FourVector &momentum, int pdg_id, int status=0, const Flow &itsflow=Flow(), const Polarization &polar=Polarization(0, 0))**
  constructor requires momentum and particle ID
- **GenParticle (const GenParticle &inparticle)**
  shallow copy.
- **virtual ~GenParticle ()**
- **void swap (GenParticle &other)**
  swap
- **GenParticle & operator= (const GenParticle &inparticle)**
- **bool operator== (const GenParticle &) const**
  check for equality
- **bool operator!= (const GenParticle &) const**
  check for inequality
- **void print (std::ostream &ostr=std::cout) const**
  dump this particle's full info to ostr
- **operator HepMC::FourVector () const**
  conversion operator
- **const FourVector & momentum () const**
  standard 4 momentum
- **int pdg_id () const**
  particle ID
- **int status () const**
  HEPEVT decay status.
- **const Flow & flow () const**
  particle flow
- **int flow (int code_index) const**
  particle flow index

- **const Polarization & polarization () const**
  polarization information
- **GenVertex ∗ production_vertex () const**
  pointer to the production vertex
- **GenVertex ∗ end_vertex () const**
  pointer to the decay vertex
- **GenEvent ∗ parent_event () const**
  pointer to the event that owns this particle
- **double generated_mass () const**
  mass as generated
- **double generatedMass () const**
  generatedMass() (p. *118) is included for backwards compatibility with CLHEP* (p. *23) HepMC* (p. *25)*
- **int barcode () const**
  particle barcode
- **bool is_undecayed () const**
  Convenience method. Returns true if status==1.
- **bool has_decayed () const**
  Convenience method. Returns true if status==2.
- **bool is_beam () const**
- **GenParticleProductionRange particles_in (IteratorRange range=relatives)**
  incoming particle range
- **ConstGenParticleProductionRange particles_in (IteratorRange range=relatives) const**
  incoming particle range
- **GenParticleEndRange particles_out (IteratorRange range=relatives)**
  outgoing particle range
- **ConstGenParticleEndRange particles_out (IteratorRange range=relatives) const**
  outgoing particle range
- **bool suggest_barcode (int the_bar_code)**
  In general there is no reason to "suggest_barcode".
- **void set_momentum (const FourVector &vec4)**
  set standard 4 momentum
- **void set_pdg_id (int id)**
  set particle ID
- **void set_status (int status=0)**

set decay status

- **void set_flow (const Flow &f)**
  set particle flow
- **void set_flow (int code_index, int code=0)**
- **void set_polarization (const Polarization &pol=Polarization(0, 0))**
  set polarization
- **void set_generated_mass (const double &m)**
  define the actual generated mass
- **void setGeneratedMass (const double &m)**
  *setGeneratedMass() (*p. *122) is included for backwards compatibility with CLHEP (*p. *23) HepMC (*p. *25)*

Protected Member Functions

- void **set_production_vertex_ (GenVertex ∗productionvertex=0)**
  set production vertex - for internal use only
- **void set_end_vertex_ (GenVertex ∗decayvertex=0)**
  set decay vertex - for internal use only
- **void set_barcode_ (int the_bar_code)**
  *for use by GenEvent (*p. *75) only*
- **void convert_momentum (const double &)**

Friends

- class **GenVertex**
- **class GenEvent**
- **std::ostream & operator<< (std::ostream &, const GenParticle &)**
  print particle

9.19.1 Detailed Description

The **GenParticle** (p. 113) class contains information about generated particles.

HepMC::GenParticle (p. 113) contains momentum, generated mass, particle ID, decay status, flow, polarization, pointers to production and decay vertices and a unique barcode identfier.

Examples:

example_BuildEventFromScratch.cc, example_UsingIterators.cc, example_VectorConversion.cc, testFlow.cc, testHepMCIteration.cc.in, testMass.cc.in, and testPrintBug.cc.

Definition at line 60 of file GenParticle.h.

9.19.2 Constructor & Destructor Documentation

9.19.2.1 HepMC::GenParticle::GenParticle (void)

default constructor

Definition at line 14 of file GenParticle.cc.

9.19.2.2 HepMC::GenParticle::GenParticle (const FourVector & *momentum*, int *pdg_id*, int *status* = 0, const Flow & *itsflow* = Flow(), const Polarization & *polar* = Polarization(0, 0))

constructor requires momentum and particle ID

Definition at line 23 of file GenParticle.cc.

References set_flow().

9.19.2.3 HepMC::GenParticle::GenParticle (const GenParticle & *inparticle*)

shallow copy.

Shallow copy: does not copy the vertex pointers (note - impossible to copy vertex pointers which having the vertex and particles in/out point-back to one another – unless you copy the entire tree – which we don't want to do)

Definition at line 37 of file GenParticle.cc.

References barcode(), set_end_vertex_(), set_production_vertex_(), and suggest_barcode().

9.19.2.4 HepMC::GenParticle::~GenParticle () [virtual]

Definition at line 58 of file GenParticle.cc.

References parent_event(), and HepMC::GenEvent::remove_barcode().

9.19.3 Member Function Documentation

9.19.3.1 int HepMC::GenParticle::barcode () const [inline]

particle barcode

The barcode is the particle's reference number, every vertex in the event has a unique barcode. Particle barcodes are positive numbers, vertex barcodes are negative numbers.

Please note that the barcodes are intended for internal use within **HepMC** (p. 25) as a unique identifier for the particles and vertices. Using the barcode to encode extra information is an abuse of the barcode data member and causes confusion among users.

Examples:

testFlow.cc.

Definition at line 252 of file GenParticle.h.

Referenced by GenParticle(), main(), HepMC::operator<<(), print(), set_end_vertex_(), and set_production_vertex_().

9.19.3.2 void HepMC::GenParticle::convert_momentum (const double &) [protected]

scale the momentum vector and generated mass this method is only for use by **GenEvent** (p. 75)

Definition at line 246 of file GenParticle.cc.

References HepMC::FourVector::e(), HepMC::FourVector::px(), HepMC::FourVector::py(), and HepMC::FourVector::pz().

9.19.3.3 GenVertex * HepMC::GenParticle::end_vertex () const [inline]

pointer to the decay vertex

Definition at line 221 of file GenParticle.h.

Referenced by HepMC::GenVertex::add_particle_in(), HepMC::ConstGenParticleEndRange::begin(), HepMC::GenParticleEndRange::begin(), HepMC::Flow::connected_partners(), HepMC::Flow::dangling_connected_partners(), HepMC::ConstGenParticleEndRange::end(), HepMC::GenParticleEndRange::end(), HepMC::operator<<(), parent_event(), print(), and HepMC::GenVertex::remove_particle().

9.19.3.4 int HepMC::GenParticle::flow (int *code_index*) const [inline]

particle flow index

Definition at line 225 of file GenParticle.h.

References HepMC::Flow::icode().

9.19.3.5 const Flow & HepMC::GenParticle::flow () const [inline]

particle flow

Examples:

testFlow.cc.

Definition at line 223 of file GenParticle.h.

Referenced by main().

9.19.3.6 double HepMC::GenParticle::generated_mass () const

mass as generated

Because of precision issues, the generated mass is not always the same as the mass calculated from the momentum 4 vector. If the generated mass has been set, then **generated_mass()** (p. 117) returns that value. If the generated mass has not been set, then **generated_mass()** (p. 117) returns the mass calculated from the momentum 4 vector.

Definition at line 236 of file GenParticle.cc.

Referenced by generatedMass(), and operator==().

9.19.3.7 double HepMC::GenParticle::generatedMass () const [inline]

generatedMass() (p. 118) is included for backwards compatibility with **CLHEP** (p. 23) **HepMC** (p. 25)

Definition at line 121 of file GenParticle.h.

References generated_mass().

9.19.3.8 bool HepMC::GenParticle::has_decayed () const [inline]

Convenience method. Returns true if status==2.

Definition at line 259 of file GenParticle.h.

9.19.3.9 bool HepMC::GenParticle::is_beam () const [inline]

Convenience method. Returns true if status==4 Note that using status 4 for beam particles is a new convention which may not have been implemented by the code originating this **GenEvent** (p. 75).

Definition at line 262 of file GenParticle.h.

9.19.3.10 bool HepMC::GenParticle::is_undecayed () const [inline]

Convenience method. Returns true if status==1.

Definition at line 256 of file GenParticle.h.

9.19.3.11 const FourVector & HepMC::GenParticle::momentum () const [inline]

standard 4 momentum

Definition at line 211 of file GenParticle.h.

Referenced by HepMC::operator<<(), operator==(), and print().

9.19.3.12 HepMC::GenParticle::operator HepMC::FourVector () const [inline]

conversion operator

Definition at line 208 of file GenParticle.h.

9.19.3.13 bool HepMC::GenParticle::operator!= (const GenParticle &) const

check for inequality

Definition at line 102 of file GenParticle.cc.

9.19.3.14 GenParticle & HepMC::GenParticle::operator= (const GenParticle & *inparticle*)

shallow.

Shallow: does not copy the vertex pointers (note - impossible to copy vertex pointers which having the vertex and particles in/out point-back to one another – unless you copy the entire tree – which we don't want to do)

Definition at line 77 of file GenParticle.cc.

References swap().

9.19.3.15 bool HepMC::GenParticle::operator== (const GenParticle &) const

check for equality

consistent with the definition of the copy constructor as a shallow constructor,.. this operator does not test the vertex pointers. Does not compare barcodes.

Definition at line 89 of file GenParticle.cc.

References generated_mass(), m_flow, momentum(), pdg_id(), polarization(), and status().

9.19.3.16 GenEvent * HepMC::GenParticle::parent_event () const

pointer to the event that owns this particle

Definition at line 123 of file GenParticle.cc.

References end_vertex(), HepMC::GenVertex::parent_event(), and production_vertex().

Referenced by set_end_vertex_(), set_production_vertex_(), suggest_barcode(), and ~GenParticle().

9.19.3.17 ConstGenParticleProductionRange HepMC::GenParticle::particles_in (IteratorRange *range* = `relatives`) const

incoming particle range

Definition at line 67 of file GenRanges.cc.

9.19.3.18 GenParticleProductionRange HepMC::GenParticle::particles_in (IteratorRange *range* = `relatives`)

incoming particle range

Definition at line 61 of file GenRanges.cc.

9.19.3.19 ConstGenParticleEndRange HepMC::GenParticle::particles_out (IteratorRange *range* = `relatives`) const

outgoing particle range

Definition at line 79 of file GenRanges.cc.

9.19.3.20 GenParticleEndRange HepMC::GenParticle::particles_out (IteratorRange *range* = `relatives`)

outgoing particle range

Definition at line 73 of file GenRanges.cc.

9.19.3.21 int HepMC::GenParticle::pdg_id () const [inline]

particle ID

Definition at line 214 of file GenParticle.h.

Referenced by PrintChildren::operator()(), HepMC::operator<<(), operator==(), and print().

9.19.3.22 const Polarization & HepMC::GenParticle::polarization () const [inline]

polarization information

Definition at line 228 of file GenParticle.h.

Referenced by operator==(), and print().

9.19.3.23 void HepMC::GenParticle::print (std::ostream & *ostr* = std::cout) const

dump this particle's full info to ostr

Dump this particle's full info to ostr, where by default particle.print(); will dump to cout.

Definition at line 106 of file GenParticle.cc.

References HepMC::GenVertex::barcode(), barcode(), HepMC::FourVector::e(), end_vertex(), momentum(), pdg_id(), polarization(), production_vertex(), HepMC::FourVector::px(), HepMC::FourVector::py(), HepMC::FourVector::pz(), and status().

9.19.3.24 GenVertex * HepMC::GenParticle::production_vertex () const [inline]

pointer to the production vertex

Definition at line 218 of file GenParticle.h.

Referenced by HepMC::GenVertex::add_particle_out(), HepMC::ConstGenParticleProductionRange::begin(), HepMC::GenParticleProductionRange::begin(), HepMC::Flow::connected_partners(), HepMC::Flow::dangling_connected_partners(), HepMC::ConstGenParticleProductionRange::end(), HepMC::GenParticleProductionRange::end(), parent_event(), print(), and HepMC::GenVertex::remove_particle().

9.19.3.25 void HepMC::GenParticle::set_barcode_ (int *the_bar_code*) [inline, protected]

for use by **GenEvent** (p. 75) only

Definition at line 254 of file GenParticle.h.

Referenced by suggest_barcode().

9.19.3.26 void HepMC::GenParticle::set_end_vertex_ (GenVertex * *decayvertex* = 0) [protected]

set decay vertex - for internal use only

Definition at line 142 of file GenParticle.cc.

References barcode(), parent_event(), and HepMC::GenEvent::remove_barcode().

Referenced by HepMC::GenVertex::add_particle_in(), GenParticle(), and HepMC::GenVertex::remove_particle().

9.19.3.27 void HepMC::GenParticle::set_flow (int *code_index*, int *code* = 0) [inline]

set particle flow index

Definition at line 240 of file GenParticle.h.

References HepMC::Flow::set_icode(), and HepMC::Flow::set_unique_icode().

9.19.3.28 void HepMC::GenParticle::set_flow (const Flow & *f*) [inline]

set particle flow

Examples:

testFlow.cc.

Definition at line 238 of file GenParticle.h.

Referenced by GenParticle(), and main().

9.19.3.29 void HepMC::GenParticle::set_generated_mass (const double & *m*)

define the actual generated mass

If you do not call **set_generated_mass()** (p. 121), then **generated_mass()** (p. 117) will simply return the mass calculated from **momentum()** (p. 118)

Definition at line 240 of file GenParticle.cc.

Referenced by setGeneratedMass().

9.19.3.30 void HepMC::GenParticle::set_momentum (const FourVector & *vec4*) [inline]

set standard 4 momentum

Definition at line 231 of file GenParticle.h.

9.19.3.31 void HepMC::GenParticle::set_pdg_id (int *id*) [inline]

set particle ID

Definition at line 234 of file GenParticle.h.

9.19.3.32 void HepMC::GenParticle::set_polarization (const Polarization & *pol* = Polarization(0, 0)) [inline]

set polarization

Definition at line 249 of file GenParticle.h.

Referenced by main().

9.19.3.33 void HepMC::GenParticle::set_production_vertex_ (GenVertex * *productionvertex* = 0) [protected]

set production vertex - for internal use only

Definition at line 129 of file GenParticle.cc.

References barcode(), parent_event(), and HepMC::GenEvent::remove_barcode().

Referenced by HepMC::GenVertex::add_particle_out(), GenParticle(), and HepMC::GenVertex::remove_particle().

9.19.3.34 void HepMC::GenParticle::set_status (int *status* = 0) [inline]

set decay status

Definition at line 236 of file GenParticle.h.

9.19.3.35 void HepMC::GenParticle::setGeneratedMass (const double & *m*) [inline]

setGeneratedMass() (p. 122) is included for backwards compatibility with **CLHEP** (p. 23) **HepMC** (p. 25)

Definition at line 173 of file GenParticle.h.

References set_generated_mass().

9.19.3.36 int HepMC::GenParticle::status () const [inline]

HEPEVT decay status.

Definition at line 216 of file GenParticle.h.

Referenced by PrintChildren::operator()(), HepMC::operator<<(), operator==(), and print().

9.19.3.37 bool HepMC::GenParticle::suggest_barcode (int *the_bar_code*)

In general there is no reason to "suggest_barcode".

allows a barcode to be suggested for this particle. In general it is better to let the event pick the barcode for you, which is automatic. Returns TRUE if the suggested barcode has been accepted (i.e. the suggested barcode has not already been used in the event, and so it was used). Returns FALSE if the suggested barcode was rejected, or if the particle is not yet part of an event, such that it is not yet possible to know if the suggested barcode will be accepted).

Definition at line 153 of file GenParticle.cc.

References parent_event(), HepMC::GenEvent::set_barcode(), and set_barcode_().

Referenced by GenParticle().

9.19.3.38 void HepMC::GenParticle::swap (GenParticle & *other*)

swap

Definition at line 63 of file GenParticle.cc.

References m_barcode, m_end_vertex, m_flow, m_generated_mass, m_momentum, m_pdg_id, m_-polarization, m_production_vertex, m_status, HepMC::Polarization::swap(), HepMC::Flow::swap(), and HepMC::FourVector::swap().

Referenced by operator=().

9.19.4 Friends And Related Function Documentation

9.19.4.1 friend class GenEvent [friend]

Definition at line 63 of file GenParticle.h.

9.19.4.2 friend class GenVertex [friend]

Definition at line 62 of file GenParticle.h.

9.19.4.3 std::ostream& operator<< (std::ostream & *ostr*, const GenParticle & *part*) [friend]

print particle

Definition at line 189 of file GenParticle.cc.

The documentation for this class was generated from the following files:

- **GenParticle.h**
- **GenParticle.cc**
- **GenRanges.cc**

9.20 HepMC::GenParticleEndRange Class Reference

GenParticleEndRange (p. 124) acts like a collection of particles.

`#include <GenRanges.h>`

Public Member Functions

- **GenParticleEndRange (GenParticle const &p, IteratorRange range=relatives)**
  the constructor requires a GenParticle (p. ***113)***
- **GenVertex::particle_iterator begin ()**
  begin iterator throws an error if the particle end_vertex is undefined
- **GenVertex::particle_iterator end ()**
  end iterator throws an error if the particle end_vertex is undefined

9.20.1 Detailed Description

GenParticleEndRange (p. 124) acts like a collection of particles.

HepMC::GenParticleEndRange (p. 124) is used to mimic a collection of particles associated with the particle's end vertex for ease of use Utilities such as the Boost foreach funtion will want to use this class.

Definition at line 224 of file GenRanges.h.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 HepMC::GenParticleEndRange::GenParticleEndRange (GenParticle const & *p*, IteratorRange *range* = `relatives`) `[inline]`

the constructor requires a **GenParticle** (p. 113)

Definition at line 229 of file GenRanges.h.

9.20.3 Member Function Documentation

9.20.3.1 GenVertex::particle_iterator HepMC::GenParticleEndRange::begin () `[inline]`

begin iterator throws an error if the particle end_vertex is undefined

Definition at line 300 of file GenRanges.h.

References HepMC::GenParticle::end_vertex(), and HepMC::GenVertex::particles_begin().

9.20.3.2 GenVertex::particle_iterator HepMC::GenParticleEndRange::end () `[inline]`

end iterator throws an error if the particle end_vertex is undefined

Definition at line 306 of file GenRanges.h.

References HepMC::GenParticle::end_vertex(), and HepMC::GenVertex::particles_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.21 HepMC::GenParticleProductionRange Class Reference

GenParticleProductionRange (p. 126) acts like a collection of particles.

`#include <GenRanges.h>`

Public Member Functions

- **GenParticleProductionRange (GenParticle const &p, IteratorRange range=relatives)**
  the constructor requires a GenParticle (p. ***113***)
- **GenVertex::particle_iterator begin ()**
  begin iterator throws an error if the particle production_vertex is undefined
- **GenVertex::particle_iterator end ()**
  end iterator throws an error if the particle production_vertex is undefined

9.21.1 Detailed Description

GenParticleProductionRange (p. 126) acts like a collection of particles.

HepMC::GenParticleProductionRange (p. 126) is used to mimic a collection of particles associated with the particle's production vertex for ease of use Utilities such as the Boost foreach funtion will want to use this class.

Definition at line 170 of file GenRanges.h.

9.21.2 Constructor & Destructor Documentation

9.21.2.1 HepMC::GenParticleProductionRange::GenParticleProductionRange (GenParticle const & *p*, IteratorRange *range* = `relatives`) `[inline]`

the constructor requires a **GenParticle** (p. 113)

Definition at line 175 of file GenRanges.h.

9.21.3 Member Function Documentation

9.21.3.1 GenVertex::particle_iterator HepMC::GenParticleProductionRange::begin () `[inline]`

begin iterator throws an error if the particle production_vertex is undefined

Definition at line 271 of file GenRanges.h.

References HepMC::GenVertex::particles_begin(), and HepMC::GenParticle::production_vertex().

9.21.3.2 GenVertex::particle_iterator HepMC::GenParticleProductionRange::end () `[inline]`

end iterator throws an error if the particle production_vertex is undefined

Definition at line 278 of file GenRanges.h.

References HepMC::GenVertex::particles_end(), and HepMC::GenParticle::production_vertex().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.22 HepMC::GenVertex Class Reference

GenVertex (p. 128) contains information about decay vertices.

`#include <GenVertex.h>`

Public Types

- typedef std::vector< **HepMC::GenParticle** * >**::const_iterator particles_in_const_iterator**
  const iterator for incoming particles
- **typedef std::vector< HepMC::GenParticle * >::const_iterator particles_out_const_iterator**
  const iterator for outgoing particles

Public Member Functions

- **GenVertex (const FourVector &position=FourVector(0, 0, 0, 0), int id=0, const WeightContainer &weights=std::vector< double >())**
  default constructor
- **GenVertex (const GenVertex &invertex)**
  shallow copy
- **virtual ~GenVertex ()**
- **void swap (GenVertex &other)**
  swap
- **GenVertex & operator= (const GenVertex &invertex)**
  shallow
- **bool operator== (const GenVertex &a) const**
  equality
- **bool operator!= (const GenVertex &a) const**
  inequality
- **void print (std::ostream &ostr=std::cout) const**
  print vertex information
- **double check_momentum_conservation () const**
  |*Sum (three_mom_in-three_mom_out)*|
- **void add_particle_in (GenParticle *inparticle)**
  add incoming particle
- **void add_particle_out (GenParticle *outparticle)**
  add outgoing particle
- **GenParticle * remove_particle (GenParticle *particle)**

remove a particle

- **operator HepMC::FourVector () const**
  conversion operator
- **operator HepMC::ThreeVector () const**
  conversion operator
- **GenEvent ∗ parent_event () const**
  pointer to the event that owns this vertex
- **ThreeVector point3d () const**
  vertex position
- **const FourVector & position () const**
  vertex position and time
- **void set_position (const FourVector &position=FourVector(0, 0, 0, 0))**
  set vertex position and time
- **int id () const**
  vertex ID
- **void set_id (int id)**
  set vertex ID
- **int barcode () const**
  unique identifier
- **bool suggest_barcode (int the_bar_code)**
  In general there is no reason to "suggest_barcode".
- **WeightContainer & weights ()**
  direct access to the weights container is allowed.
- **const WeightContainer & weights () const**
  const direct access to the weights container
- **GenVertexParticleRange particles (IteratorRange range=relatives)**
  particle range
- **GenParticleProductionRange particles_in (GenParticle &, IteratorRange range=relatives)**
  incoming particle range
- **ConstGenParticleProductionRange particles_in (GenParticle const &, IteratorRange range=relatives) const**
  incoming particle range
- **GenParticleEndRange particles_out (GenParticle &, IteratorRange range=relatives)**
  outgoing particle range

- **ConstGenParticleEndRange particles_out (GenParticle const &, IteratorRange range=relatives) const**
  outgoing particle range
- **particles_in_const_iterator particles_in_const_begin () const**
  begin iteration of incoming particles
- **particles_in_const_iterator particles_in_const_end () const**
  end iteration of incoming particles
- **particles_out_const_iterator particles_out_const_begin () const**
  begin iteration of outgoing particles
- **particles_out_const_iterator particles_out_const_end () const**
  end iteration of outgoing particles
- **int particles_in_size () const**
  number of incoming particles
- **int particles_out_size () const**
  number of outgoing particles
- **vertex_iterator vertices_begin (IteratorRange range=relatives)**
  begin vertex range
- **vertex_iterator vertices_end (IteratorRange)**
  end vertex range
- **particle_iterator particles_begin (IteratorRange range=relatives)**
  begin particle range
- **particle_iterator particles_end (IteratorRange)**
  end particle range

Protected Member Functions

- void **set_parent_event_ (GenEvent *evt)**
  set parent event
- **void set_barcode_ (int the_bar_code)**
  set identifier
- **void change_parent_event_ (GenEvent *evt)**
  for use with swap
- **int edges_size (IteratorRange range=family) const**
  size

- **edge_iterator edges_begin (IteratorRange range=family) const**
  begin range
- **edge_iterator edges_end (IteratorRange) const**
  end range
- **void delete_adopted_particles ()**
  for internal use only
- **void remove_particle_in (GenParticle ∗)**
  for internal use only - remove particle from incoming list
- **void remove_particle_out (GenParticle ∗)**
  for internal use only - remove particle from outgoing list
- **void convert_position (const double &)**

Friends

- class **GenEvent**
- **class edge_iterator**
- **class vertex_iterator**
- **class particle_iterator**
- **std::ostream & operator<< (std::ostream &, const GenVertex &)**
  print vertex information

Classes

- class **edge_iterator**
  edge iterator
- **class particle_iterator**
  particle iterator
- **class vertex_iterator**
  vertex iterator

9.22.1 Detailed Description

GenVertex (p. 128) contains information about decay vertices.

HepMC::GenVertex (p. 128) contains the position in space and time of a decay. It also contains lists of incoming and outgoing particles.

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, testFlow.cc, and testPrintBug.cc.

Definition at line 52 of file GenVertex.h.

9.22.2 Member Typedef Documentation

9.22.2.1 typedef std::vector<HepMC::GenParticle*>::const_iterator HepMC::GenVertex::particles_in_const_iterator

const iterator for incoming particles

Definition at line 152 of file GenVertex.h.

9.22.2.2 typedef std::vector<HepMC::GenParticle*>::const_iterator HepMC::GenVertex::particles_out_const_iterator

const iterator for outgoing particles

Definition at line 155 of file GenVertex.h.

9.22.3 Constructor & Destructor Documentation

9.22.3.1 HepMC::GenVertex::GenVertex (const FourVector & *position* = FourVector(0, 0, 0, 0), int *id* = 0, const WeightContainer & *weights* = std::vector< double >())

default constructor

Definition at line 14 of file GenVertex.cc.

9.22.3.2 HepMC::GenVertex::GenVertex (const GenVertex & *invertex*)

shallow copy

Shallow copy: does not copy the FULL list of particle pointers. Creates a copy of - invertex

- outgoing particles of invertex, but sets the decay vertex of these particles to NULL
- all incoming particles which do not have a creation vertex. (i.e. it creates copies of all particles which it owns) (note - impossible to copy the FULL list of particle pointers while having the vertex and particles in/out point-back to one another – unless you copy the entire tree – which we don't want to do)

Definition at line 23 of file GenVertex.cc.

References add_particle_in(), add_particle_out(), barcode(), particles_in_const_begin(), particles_in_const_end(), particles_out_const_begin(), particles_out_const_end(), and suggest_barcode().

9.22.3.3 HepMC::GenVertex::~GenVertex () [virtual]

Definition at line 63 of file GenVertex.cc.

References delete_adopted_particles(), parent_event(), and HepMC::GenEvent::remove_barcode().

9.22.4 Member Function Documentation

9.22.4.1 void HepMC::GenVertex::add_particle_in (GenParticle * *inparticle*)

add incoming particle

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, testFlow.cc, and testPrintBug.cc.

Definition at line 273 of file GenVertex.cc.

References HepMC::GenParticle::end_vertex(), remove_particle_in(), and HepMC::GenParticle::set_end_vertex_().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), HepMC::IO_HERWIG::fill_next_event(), GenVertex(), main(), and HepMC::GenEvent::read().

9.22.4.2 void HepMC::GenVertex::add_particle_out (GenParticle * *outparticle*)

add outgoing particle

Examples:

example_BuildEventFromScratch.cc, example_VectorConversion.cc, testFlow.cc, and testPrintBug.cc.

Definition at line 284 of file GenVertex.cc.

References HepMC::GenParticle::production_vertex(), remove_particle_out(), and HepMC::GenParticle::set_production_vertex_().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_HEPEVT::fill_next_event(), GenVertex(), and main().

9.22.4.3 int HepMC::GenVertex::barcode () const [inline]

unique identifier

The barcode is the vertex's reference number, every vertex in the event has a unique barcode. Vertex barcodes are negative numbers, particle barcodes are positive numbers.

Please note that the barcodes are intended for internal use within **HepMC** (p. 25) as a unique identifier for the particles and vertices. Using the barcode to encode extra information is an abuse of the barcode data member and causes confusion among users.

Definition at line 416 of file GenVertex.h.

Referenced by HepMC::GenEvent::add_vertex(), HepMC::compareVertex(), GenVertex(), HepMC::operator<<(), print(), HepMC::GenParticle::print(), HepMC::GenEvent::print(), HepMC::GenEvent::remove_vertex(), set_parent_event_(), and HepMC::IO_AsciiParticles::write_event().

9.22.4.4 void HepMC::GenVertex::change_parent_event_ (GenEvent * *evt*) [protected]

for use with swap

Definition at line 419 of file GenVertex.cc.

9.22.4.5 double HepMC::GenVertex::check_momentum_conservation () const

|Sum (three_mom_in-three_mom_out)|

finds the difference between the total momentum out and the total momentum in vectors, and returns the magnitude of this vector i.e. returns | vec{p_in} - vec{p_out} |

Definition at line 253 of file GenVertex.cc.

References particles_in_const_begin(), particles_in_const_end(), particles_out_const_begin(), and particles_out_const_end().

9.22.4.6 void HepMC::GenVertex::convert_position (const double &) [protected]

scale the position vector this method is only for use by **GenEvent** (p. 75)

Definition at line 918 of file GenVertex.cc.

References HepMC::FourVector::t(), HepMC::FourVector::x(), HepMC::FourVector::y(), and HepMC::FourVector::z().

9.22.4.7 void HepMC::GenVertex::delete_adopted_particles () [protected]

for internal use only

deletes all particles which this vertex owns to be used by the vertex destructor and operator=

Definition at line 329 of file GenVertex.cc.

Referenced by ~GenVertex().

9.22.4.8 GenVertex::edge_iterator HepMC::GenVertex::edges_begin (IteratorRange *range* = family) const [inline, protected]

begin range

Definition at line 476 of file GenVertex.h.

Referenced by HepMC::GenVertex::vertex_iterator::vertex_iterator().

9.22.4.9 GenVertex::edge_iterator HepMC::GenVertex::edges_end (IteratorRange) const [inline, protected]

end range

Definition at line 481 of file GenVertex.h.

Referenced by HepMC::GenVertex::vertex_iterator::operator++(), and HepMC::GenVertex::vertex_iterator::vertex_iterator().

9.22.4.10 int HepMC::GenVertex::edges_size (IteratorRange *range* = family) const [protected]

size

Definition at line 595 of file GenVertex.cc.

References HepMC::children, HepMC::family, and HepMC::parents.

9.22.4.11 int HepMC::GenVertex::id () const [inline]

vertex ID

we don't define what you use the id for – but we imagine, for example it might code the meaning of the **weights()** (p. 141)

Definition at line 414 of file GenVertex.h.

Referenced by print().

9.22.4.12 HepMC::GenVertex::operator HepMC::FourVector () const [inline]

conversion operator

Definition at line 402 of file GenVertex.h.

References position().

9.22.4.13 HepMC::GenVertex::operator HepMC::ThreeVector () const [inline]

conversion operator

Definition at line 404 of file GenVertex.h.

References point3d().

9.22.4.14 bool HepMC::GenVertex::operator!= (const GenVertex & *a*) const

inequality

Definition at line 140 of file GenVertex.cc.

9.22.4.15 GenVertex & HepMC::GenVertex::operator= (const GenVertex & *invertex*)

shallow

Shallow: does not copy the FULL list of particle pointers. Creates a copy of - invertex

- outgoing particles of invertex, but sets the decay vertex of these particles to NULL
- all incoming particles which do not have a creation vertex.
- it does not alter ∗this's m_event (!) (i.e. it creates copies of all particles which it owns) (note - impossible to copy the FULL list of particle pointers while having the vertex and particles in/out point-back to one another – unless you copy the entire tree – which we don't want to do)

Definition at line 82 of file GenVertex.cc.

References swap().

9.22.4.16 bool HepMC::GenVertex::operator== (const GenVertex & *a*) const

equality

Returns true if the positions and the particles in the lists of a and this are identical. Does not compare barcodes. Note that it is impossible for two vertices to point to the same particle's address, so we need to do more than just compare the particle pointers

Definition at line 103 of file GenVertex.cc.

References particles_in_const_begin(), particles_in_const_end(), particles_in_size(), particles_out_const_begin(), particles_out_const_end(), particles_out_size(), and position().

9.22.4.17 GenEvent * HepMC::GenVertex::parent_event () const [inline]

pointer to the event that owns this vertex

Definition at line 408 of file GenVertex.h.

Referenced by HepMC::GenEvent::add_vertex(), HepMC::GenParticle::parent_event(), HepMC::GenEvent::remove_vertex(), suggest_barcode(), and ~GenVertex().

9.22.4.18 GenVertexParticleRange HepMC::GenVertex::particles (IteratorRange *range* = relatives)

particle range

Definition at line 36 of file GenRanges.cc.

9.22.4.19 GenVertex::particle_iterator HepMC::GenVertex::particles_begin (IteratorRange *range* = relatives) [inline]

begin particle range

Definition at line 525 of file GenVertex.h.

References particle_iterator.

Referenced by HepMC::ConstGenParticleEndRange::begin(), HepMC::GenParticleEndRange::begin(), HepMC::ConstGenParticleProductionRange::begin(), HepMC::GenParticleProductionRange::begin(), HepMC::GenVertexParticleRange::begin(), HepMC::Flow::connected_partners(), and HepMC::Flow::dangling_connected_partners().

9.22.4.20 GenVertex::particle_iterator HepMC::GenVertex::particles_end (IteratorRange) [inline]

end particle range

Definition at line 530 of file GenVertex.h.

References particle_iterator.

Referenced by HepMC::Flow::connected_partners(), HepMC::Flow::dangling_connected_partners(), HepMC::ConstGenParticleEndRange::end(), HepMC::GenParticleEndRange::end(), HepMC::ConstGenParticleProductionRange::end(), HepMC::GenParticleProductionRange::end(), and HepMC::GenVertexParticleRange::end().

9.22.4.21 ConstGenParticleProductionRange HepMC::GenVertex::particles_in (GenParticle const &, IteratorRange *range* = relatives) const

incoming particle range

Definition at line 46 of file GenRanges.cc.

References p.

9.22.4.22 GenParticleProductionRange HepMC::GenVertex::particles_in (GenParticle &, IteratorRange *range* = relatives)

incoming particle range

Definition at line 41 of file GenRanges.cc.

References p.

9.22.4.23 GenVertex::particles_in_const_iterator HepMC::GenVertex::particles_in_const_begin () const [inline]

begin iteration of incoming particles

Definition at line 435 of file GenVertex.h.

Referenced by check_momentum_conservation(), HepMC::compareVertex(), GenVertex(), operator==(), print(), and set_parent_event_().

9.22.4.24 GenVertex::particles_in_const_iterator HepMC::GenVertex::particles_in_const_end () const [inline]

end iteration of incoming particles

Definition at line 440 of file GenVertex.h.

Referenced by check_momentum_conservation(), HepMC::compareVertex(), GenVertex(), operator==(), print(), and set_parent_event_().

9.22.4.25 int HepMC::GenVertex::particles_in_size () const [inline]

number of incoming particles

Definition at line 454 of file GenVertex.h.

Referenced by HepMC::compareVertex(), and operator==().

9.22.4.26 ConstGenParticleEndRange HepMC::GenVertex::particles_out (GenParticle const &, IteratorRange *range* = relatives) const

outgoing particle range

Definition at line 56 of file GenRanges.cc.

References p.

9.22.4.27 GenParticleEndRange HepMC::GenVertex::particles_out (GenParticle &, IteratorRange *range* = relatives)

outgoing particle range

Definition at line 51 of file GenRanges.cc.

References p.

9.22.4.28 GenVertex::particles_out_const_iterator HepMC::GenVertex::particles_out_const_begin () const [inline]

begin iteration of outgoing particles

Definition at line 445 of file GenVertex.h.

Referenced by check_momentum_conservation(), HepMC::compareVertex(), filterEvent(), GenVertex(), operator==(), print(), and set_parent_event_().

9.22.4.29 GenVertex::particles_out_const_iterator HepMC::GenVertex::particles_out_const_end () const [inline]

end iteration of outgoing particles

Definition at line 450 of file GenVertex.h.

Referenced by check_momentum_conservation(), HepMC::compareVertex(), filterEvent(), GenVertex(), operator==(), print(), and set_parent_event_().

9.22.4.30 int HepMC::GenVertex::particles_out_size () const [inline]

number of outgoing particles

Definition at line 458 of file GenVertex.h.

Referenced by HepMC::compareVertex(), filterEvent(), and operator==().

9.22.4.31 ThreeVector HepMC::GenVertex::point3d () const [inline]

vertex position

Definition at line 410 of file GenVertex.h.

References HepMC::FourVector::x(), HepMC::FourVector::y(), and HepMC::FourVector::z().

Referenced by operator HepMC::ThreeVector().

9.22.4.32 const FourVector & HepMC::GenVertex::position () const [inline]

vertex position and time

Definition at line 406 of file GenVertex.h.

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), HepMC::compareVertex(), operator HepMC::FourVector(), HepMC::operator<<(), operator==(), and print().

9.22.4.33 void HepMC::GenVertex::print (std::ostream & *ostr* = `std::cout`) const

print vertex information

Definition at line 145 of file GenVertex.cc.

References barcode(), HepMC::WeightContainer::end(), id(), particles_in_const_begin(), particles_in_const_end(), particles_out_const_begin(), particles_out_const_end(), position(), HepMC::WeightContainer::size(), HepMC::FourVector::t(), weights(), HepMC::FourVector::x(), HepMC::FourVector::y(), and HepMC::FourVector::z().

Referenced by HepMC::IO_HERWIG::build_production_vertex().

9.22.4.34 GenParticle * HepMC::GenVertex::remove_particle (GenParticle * *particle*)

remove a particle

remove_particle finds *particle in the in and/or out list and removes it from these lists ... it DOES NOT DELETE THE PARTICLE or its relations. You could delete the particle too as follows: delete vtx->remove_particle(particle);

this finds *particle in the in and/or out list and removes it from these lists ... it DOES NOT DELETE THE PARTICLE or its relations. you could delete the particle too as follows: delete vtx->remove_particle(particle); or if the particle has an end vertex, you could: delete vtx->remove_particle(particle)->end_vertex(); which would delete the particle's end vertex, and thus would also delete the particle, since the particle would be owned by the end vertex.

Definition at line 295 of file GenVertex.cc.

References HepMC::GenParticle::end_vertex(), HepMC::GenParticle::production_vertex(), remove_particle_in(), remove_particle_out(), HepMC::GenParticle::set_end_vertex_(), and HepMC::GenParticle::set_production_vertex_().

Referenced by filterEvent().

9.22.4.35 void HepMC::GenVertex::remove_particle_in (GenParticle *) [protected]

for internal use only - remove particle from incoming list

this finds *particle in m_particles_in and removes it from that list

Definition at line 317 of file GenVertex.cc.

References HepMC::already_in_vector().

Referenced by add_particle_in(), and remove_particle().

9.22.4.36 void HepMC::GenVertex::remove_particle_out (GenParticle *) [protected]

for internal use only - remove particle from outgoing list

this finds *particle in m_particles_out and removes it from that list

Definition at line 323 of file GenVertex.cc.

References HepMC::already_in_vector().

Referenced by add_particle_out(), and remove_particle().

9.22.4.37 void HepMC::GenVertex::set_barcode_ (int *the_bar_code*) [inline, protected]

set identifier

Definition at line 417 of file GenVertex.h.

Referenced by suggest_barcode().

9.22.4.38 void HepMC::GenVertex::set_id (int *id*) [inline]

set vertex ID

Definition at line 428 of file GenVertex.h.

9.22.4.39 void HepMC::GenVertex::set_parent_event_ (GenEvent * *evt*) [protected]

set parent event

only the **GenEvent** (p. 75) (friend) is allowed to set the parent_event, and barcode. It is done automatically anytime you add a vertex to an event

Definition at line 388 of file GenVertex.cc.

References barcode(), particles_in_const_begin(), particles_in_const_end(), particles_out_const_begin(), particles_out_const_end(), HepMC::GenEvent::remove_barcode(), and HepMC::GenEvent::set_barcode().

Referenced by HepMC::GenEvent::add_vertex(), and HepMC::GenEvent::remove_vertex().

9.22.4.40 void HepMC::GenVertex::set_position (const FourVector & *position* = FourVector(0, 0, 0, 0)) [inline]

set vertex position and time

Definition at line 424 of file GenVertex.h.

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), and HepMC::IO_HEPEVT::build_production_vertex().

9.22.4.41 bool HepMC::GenVertex::suggest_barcode (int *the_bar_code*)

In general there is no reason to "suggest_barcode".

allows a barcode to be suggested for this vertex. In general it is better to let the event pick the barcode for you, which is automatic. Returns TRUE if the suggested barcode has been accepted (i.e. the suggested barcode has not already been used in the event, and so it was used). Returns FALSE if the suggested barcode was rejected, or if the vertex is not yet part of an event, such that it is not yet possible to know if the suggested barcode will be accepted).

Definition at line 363 of file GenVertex.cc.

References parent_event(), HepMC::GenEvent::set_barcode(), and set_barcode_().

Referenced by GenVertex().

9.22.4.42 void HepMC::GenVertex::swap (GenVertex & *other*)

swap

Definition at line 71 of file GenVertex.cc.

References m_barcode, m_event, m_id, m_particles_in, m_particles_out, m_position, m_weights, HepMC::WeightContainer::swap(), and HepMC::FourVector::swap().

Referenced by operator=().

9.22.4.43 GenVertex::vertex_iterator HepMC::GenVertex::vertices_begin (IteratorRange *range* = relatives) [inline]

begin vertex range

Definition at line 504 of file GenVertex.h.

References vertex_iterator.

9.22.4.44 GenVertex::vertex_iterator HepMC::GenVertex::vertices_end (IteratorRange) [inline]

end vertex range

Definition at line 510 of file GenVertex.h.

References vertex_iterator.

9.22.4.45 const WeightContainer & HepMC::GenVertex::weights () const [inline]

const direct access to the weights container

Definition at line 421 of file GenVertex.h.

9.22.4.46 WeightContainer & HepMC::GenVertex::weights () [inline]

direct access to the weights container is allowed.

Definition at line 419 of file GenVertex.h.

Referenced by print().

9.22.5 Friends And Related Function Documentation

9.22.5.1 friend class edge_iterator [friend]

Definition at line 233 of file GenVertex.h.

9.22.5.2 friend class GenEvent [friend]

Definition at line 56 of file GenVertex.h.

9.22.5.3 std::ostream& operator<< (std::ostream & *ostr*, const GenVertex & *vtx*) [friend]

print vertex information

Definition at line 440 of file GenVertex.cc.

9.22.5.4 friend class particle_iterator [friend]

Definition at line 366 of file GenVertex.h.

Referenced by particles_begin(), and particles_end().

9.22.5.5 friend class vertex_iterator [friend]

Definition at line 318 of file GenVertex.h.

Referenced by vertices_begin(), and vertices_end().

The documentation for this class was generated from the following files:

- **GenVertex.h**
- **GenRanges.cc**
- **GenVertex.cc**

9.23 HepMC::GenVertex::edge_iterator Class Reference

edge iterator

`#include <GenVertex.h>`

Public Member Functions

- **edge_iterator ()**
- **edge_iterator (const GenVertex &vtx, IteratorRange range=family)**
  used to set limits on the iteration
- **edge_iterator (const edge_iterator &p)**
  copy
- **virtual ~edge_iterator ()**
- **edge_iterator & operator= (const edge_iterator &p)**
  make a copy
- **GenParticle * operator * (void) const**
  return a pointer to a particle
- **edge_iterator & operator++ (void)**
  Pre-fix increment.
- **edge_iterator operator++ (int)**
  Post-fix increment.
- **bool operator== (const edge_iterator &a) const**
  equality
- **bool operator!= (const edge_iterator &a) const**
  inequality
- **bool is_parent () const**
  true if parent of root vtx
- **bool is_child () const**
  true if child of root vtx
- **const GenVertex * vertex_root () const**
  root vertex of this iteration

9.23.1 Detailed Description

edge iterator

iterate over the family of edges connected to m_vertex begins with parents (incoming particles) then children (outgoing) This is not a recursive iterator ... it is a building block for the public iterators and is intended for internal use only. The acceptable Iterator Ranges are: family, parents, children

Definition at line 194 of file GenVertex.h.

9.23.2 Constructor & Destructor Documentation

9.23.2.1 HepMC::GenVertex::edge_iterator::edge_iterator ()

Definition at line 462 of file GenVertex.cc.

9.23.2.2 HepMC::GenVertex::edge_iterator::edge_iterator (const GenVertex & *vtx*, IteratorRange *range* = `family`)

used to set limits on the iteration

Definition at line 466 of file GenVertex.cc.

References HepMC::ancestors, HepMC::children, HepMC::descendants, HepMC::family, HepMC::GenVertex::m_particles_in, HepMC::GenVertex::m_particles_out, and HepMC::parents.

9.23.2.3 HepMC::GenVertex::edge_iterator::edge_iterator (const edge_iterator & *p*)

copy

Definition at line 517 of file GenVertex.cc.

References p.

9.23.2.4 HepMC::GenVertex::edge_iterator::~edge_iterator () `[virtual]`

Definition at line 521 of file GenVertex.cc.

9.23.3 Member Function Documentation

9.23.3.1 bool HepMC::GenVertex::edge_iterator::is_child () const

true if child of root vtx

Definition at line 590 of file GenVertex.cc.

9.23.3.2 bool HepMC::GenVertex::edge_iterator::is_parent () const

true if parent of root vtx

Definition at line 585 of file GenVertex.cc.

Referenced by HepMC::GenVertex::particle_iterator::advance_to_first_(), and HepMC::GenVertex::vertex_iterator::follow_edge_().

9.23.3.3 GenParticle * HepMC::GenVertex::edge_iterator::operator * (void) const

return a pointer to a particle

Definition at line 533 of file GenVertex.cc.

9.23.3.4 bool HepMC::GenVertex::edge_iterator::operator!= (const edge_iterator & *a*) const [inline]

inequality

Definition at line 467 of file GenVertex.h.

9.23.3.5 GenVertex::edge_iterator HepMC::GenVertex::edge_iterator::operator++ (int)

Post-fix increment.

Definition at line 578 of file GenVertex.cc.

9.23.3.6 GenVertex::edge_iterator & HepMC::GenVertex::edge_iterator::operator++ (void)

Pre-fix increment.

Definition at line 538 of file GenVertex.cc.

References HepMC::family, HepMC::GenVertex::m_particles_in, HepMC::GenVertex::m_particles_out, and HepMC::parents.

9.23.3.7 GenVertex::edge_iterator & HepMC::GenVertex::edge_iterator::operator= (const edge_iterator & *p*)

make a copy

Definition at line 523 of file GenVertex.cc.

References p.

9.23.3.8 bool HepMC::GenVertex::edge_iterator::operator== (const edge_iterator & *a*) const [inline]

equality

Definition at line 462 of file GenVertex.h.

9.23.3.9 const GenVertex * HepMC::GenVertex::edge_iterator::vertex_root () const [inline]

root vertex of this iteration

Definition at line 472 of file GenVertex.h.

The documentation for this class was generated from the following files:

- **GenVertex.h**
- **GenVertex.cc**

9.24 HepMC::GenVertex::particle_iterator Class Reference

particle iterator

`#include <GenVertex.h>`

Public Member Functions

- **particle_iterator ()**
- **particle_iterator (GenVertex &vertex_root, IteratorRange range)**
  used to set limits on the iteration
- **particle_iterator (const particle_iterator &)**
  copy
- **virtual ~particle_iterator ()**
- **particle_iterator & operator= (const particle_iterator &)**
  make a copy
- **GenParticle * operator * (void) const**
  return a pointer to a particle
- **particle_iterator & operator++ (void)**
  Pre-fix increment.
- **particle_iterator operator++ (int)**
  Post-fix increment.
- **bool operator== (const particle_iterator &) const**
  equality
- **bool operator!= (const particle_iterator &) const**
  inequality

Protected Member Functions

- **GenParticle * advance_to_first_ ()**
  "first" particle

9.24.1 Detailed Description

particle iterator

Iterates over all particles connected via a graph. by iterating through all vertices in the m_range. For each vertex it returns orphaned parent particles (i.e. parents without production vertices) then children ... in this way each particle is associated to exactly one vertex and so it is returned exactly once. Is made friend so that it can access protected edge iterator

Examples:

example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 339 of file GenVertex.h.

9.24.2 Constructor & Destructor Documentation

9.24.2.1 HepMC::GenVertex::particle_iterator::particle_iterator ()

Definition at line 838 of file GenVertex.cc.

9.24.2.2 HepMC::GenVertex::particle_iterator::particle_iterator (GenVertex & *vertex_root*, IteratorRange *range*)

used to set limits on the iteration

Definition at line 840 of file GenVertex.cc.

References advance_to_first_(), HepMC::family, and HepMC::GenVertex::vertex_iterator::range().

9.24.2.3 HepMC::GenVertex::particle_iterator::particle_iterator (const particle_iterator &)

copy

Definition at line 854 of file GenVertex.cc.

9.24.2.4 HepMC::GenVertex::particle_iterator::~particle_iterator () [virtual]

Definition at line 859 of file GenVertex.cc.

9.24.3 Member Function Documentation

9.24.3.1 GenParticle * HepMC::GenVertex::particle_iterator::advance_to_first_ () [protected]

"first" particle

if the current edge is not a suitable return value (because it is a parent of the vertex root that itself belongs to a different vertex) it advances to the first suitable return value

Definition at line 900 of file GenVertex.cc.

References HepMC::GenVertex::edge_iterator::is_parent(), HepMC::GenVertex::vertex_iterator::range(), and HepMC::relatives.

Referenced by operator++(), and particle_iterator().

9.24.3.2 GenParticle * HepMC::GenVertex::particle_iterator::operator * (void) const

return a pointer to a particle

Definition at line 869 of file GenVertex.cc.

9.24.3.3 bool HepMC::GenVertex::particle_iterator::operator!= (const particle_iterator &) const [inline]

inequality

Definition at line 520 of file GenVertex.h.

9.24.3.4 GenVertex::particle_iterator HepMC::GenVertex::particle_iterator::operator++ (int)

Post-fix increment.

Definition at line 893 of file GenVertex.cc.

9.24.3.5 GenVertex::particle_iterator & HepMC::GenVertex::particle_iterator::operator++ (void)

Pre-fix increment.

Definition at line 874 of file GenVertex.cc.

References advance_to_first_(), and HepMC::GenVertex::vertex_iterator::range().

9.24.3.6 GenVertex::particle_iterator & HepMC::GenVertex::particle_iterator::operator= (const particle_iterator &)

make a copy

Definition at line 862 of file GenVertex.cc.

References m_edge, and m_vertex_iterator.

9.24.3.7 bool HepMC::GenVertex::particle_iterator::operator== (const particle_iterator &) const [inline]

equality

Definition at line 515 of file GenVertex.h.

The documentation for this class was generated from the following files:

- **GenVertex.h**
- **GenVertex.cc**

9.25 HepMC::GenVertex::vertex_iterator Class Reference

vertex iterator

`#include <GenVertex.h>`

Public Member Functions

- **vertex_iterator ()**
- **vertex_iterator (GenVertex &vtx_root, IteratorRange range)**

  used to set limits on the iteration

- **vertex_iterator (GenVertex &vtx_root, IteratorRange range, std::set< const HepMC::GenVertex * > &visited_vertices)**

  next constructor is intended for internal use only

- **vertex_iterator (const vertex_iterator &v_iter)**

  copy

- **virtual ~vertex_iterator ()**
- **vertex_iterator & operator= (const vertex_iterator &)**

  make a copy

- **GenVertex * operator * (void) const**

  return a pointer to a vertex

- **vertex_iterator & operator++ (void)**

  Pre-fix increment.

- **vertex_iterator operator++ (int)**

  Post-fix increment.

- **bool operator== (const vertex_iterator &) const**

  equality

- **bool operator!= (const vertex_iterator &) const**

  inequality

- **GenVertex * vertex_root () const**

  vertex that this iterator begins from

- **IteratorRange range () const**

  iterator range

- **void copy_with_own_set (const vertex_iterator &v_iter, std::set< const HepMC::GenVertex * > &visited_vertices)**

  intended for internal use only.

Protected Member Functions

- **GenVertex ∗ follow_edge_ ()**
  non-null if recursive iter. created
- **void copy_recursive_iterator_ (const vertex_iterator ∗recursive_v_iter)**
  copy recursive iterator

9.25.1 Detailed Description

vertex iterator

Iterates over all vertices connected via a graph to this vertex. this is made friend to that it can access protected edge iterator the range can be IteratorRange= (parents, children, family, ancestors, descendants, relatives) example for range=descendants the iterator will return all vertices which are children (connected by an outgoing particle edge), grandchildren, great-grandchildren, etc. of this vertex In all cases the iterator always returns this vertex (returned last). The algorithm is accomplished by converting the graph to a tree (by "chopping" the edges connecting to an already visited vertex) and returning the vertices in POST ORDER traversal.

Definition at line 263 of file GenVertex.h.

9.25.2 Constructor & Destructor Documentation

9.25.2.1 HepMC::GenVertex::vertex_iterator::vertex_iterator ()

Definition at line 607 of file GenVertex.cc.

Referenced by copy_recursive_iterator_(), and follow_edge_().

9.25.2.2 HepMC::GenVertex::vertex_iterator::vertex_iterator (GenVertex & *vtx_root*, IteratorRange *range*)

used to set limits on the iteration

Definition at line 612 of file GenVertex.cc.

References HepMC::GenVertex::edges_begin(), HepMC::GenVertex::edges_end(), and follow_edge_().

9.25.2.3 HepMC::GenVertex::vertex_iterator::vertex_iterator (GenVertex & *vtx_root*, IteratorRange *range*, std::set< const HepMC::GenVertex ∗ > & *visited_vertices*)

next constructor is intended for internal use only

Definition at line 628 of file GenVertex.cc.

References HepMC::GenVertex::edges_begin(), HepMC::GenVertex::edges_end(), and follow_edge_().

9.25.2.4 HepMC::GenVertex::vertex_iterator::vertex_iterator (const vertex_iterator & *v_iter*)

copy

Definition at line 645 of file GenVertex.cc.

9.25.2.5 HepMC::GenVertex::vertex_iterator::∼vertex_iterator () [virtual]

Definition at line 652 of file GenVertex.cc.

9.25.3 Member Function Documentation

9.25.3.1 void HepMC::GenVertex::vertex_iterator::copy_recursive_iterator_ (const vertex_iterator * *recursive_v_iter*) [protected]

copy recursive iterator

Definition at line 817 of file GenVertex.cc.

References copy_recursive_iterator_(), m_edge, m_it_owns_set, m_range, m_recursive_iterator, m_vertex, m_visited_vertices, and vertex_iterator().

Referenced by copy_recursive_iterator_(), copy_with_own_set(), and operator=().

9.25.3.2 void HepMC::GenVertex::vertex_iterator::copy_with_own_set (const vertex_iterator & *v_iter*, std::set< const HepMC::GenVertex * > & *visited_vertices*)

intended for internal use only.

intended for internal use only. (use with care!) this is the same as the operator= method, but it allows the user to specify which set container m_visited_vertices points to. in all cases, this vertex will NOT own its set.

Definition at line 758 of file GenVertex.cc.

References copy_recursive_iterator_(), m_edge, m_range, m_recursive_iterator, and m_vertex.

9.25.3.3 GenVertex * HepMC::GenVertex::vertex_iterator::follow_edge_ () [protected]

non-null if recursive iter. created

Definition at line 781 of file GenVertex.cc.

References HepMC::family, HepMC::GenVertex::edge_iterator::is_parent(), and vertex_iterator().

Referenced by operator++(), and vertex_iterator().

9.25.3.4 GenVertex * HepMC::GenVertex::vertex_iterator::operator * (void) const

return a pointer to a vertex

Definition at line 694 of file GenVertex.cc.

9.25.3.5 bool HepMC::GenVertex::vertex_iterator::operator!= (const vertex_iterator &) const [inline]

inequality

Definition at line 491 of file GenVertex.h.

9.25.3.6 GenVertex::vertex_iterator HepMC::GenVertex::vertex_iterator::operator++ (int)

Post-fix increment.

Definition at line 751 of file GenVertex.cc.

9.25.3.7 GenVertex::vertex_iterator & HepMC::GenVertex::vertex_iterator::operator++ (void)

Pre-fix increment.

Definition at line 709 of file GenVertex.cc.

References HepMC::GenVertex::edges_end(), and follow_edge_().

9.25.3.8 GenVertex::vertex_iterator & HepMC::GenVertex::vertex_iterator::operator= (const vertex_iterator &)

make a copy

Definition at line 657 of file GenVertex.cc.

References copy_recursive_iterator_(), m_edge, m_it_owns_set, m_range, m_recursive_iterator, m_vertex, and m_visited_vertices.

9.25.3.9 bool HepMC::GenVertex::vertex_iterator::operator== (const vertex_iterator &) const [inline]

equality

Definition at line 486 of file GenVertex.h.

9.25.3.10 IteratorRange HepMC::GenVertex::vertex_iterator::range () const [inline]

iterator range

Definition at line 500 of file GenVertex.h.

Referenced by HepMC::GenVertex::particle_iterator::advance_to_first_(), HepMC::GenVertex::particle_-iterator::operator++(), and HepMC::GenVertex::particle_iterator::particle_iterator().

9.25.3.11 GenVertex * HepMC::GenVertex::vertex_iterator::vertex_root () const [inline]

vertex that this iterator begins from

Definition at line 496 of file GenVertex.h.

The documentation for this class was generated from the following files:

- **GenVertex.h**
- **GenVertex.cc**

9.26 HepMC::GenVertexParticleRange Class Reference

GenVertexParticleRange (p. 153) acts like a collection of particles.

`#include <GenRanges.h>`

Public Member Functions

- **GenVertexParticleRange (GenVertex &v, IteratorRange range=relatives)**
  the constructor requires a GenVertex (p. 128)
- **GenVertex::particle_iterator begin ()**
- **GenVertex::particle_iterator end ()**

9.26.1 Detailed Description

GenVertexParticleRange (p. 153) acts like a collection of particles.

HepMC::GenVertexParticleRange (p. 153) is used to mimic a collection of particles for ease of use - especially with utilities such as the Boost foreach funtion

Definition at line 140 of file GenRanges.h.

9.26.2 Constructor & Destructor Documentation

9.26.2.1 HepMC::GenVertexParticleRange::GenVertexParticleRange (GenVertex & *v*, IteratorRange *range* = `relatives`) `[inline]`

the constructor requires a **GenVertex** (p. 128)

Definition at line 145 of file GenRanges.h.

9.26.3 Member Function Documentation

9.26.3.1 GenVertex::particle_iterator HepMC::GenVertexParticleRange::begin () `[inline]`

Definition at line 148 of file GenRanges.h.

References HepMC::GenVertex::particles_begin().

9.26.3.2 GenVertex::particle_iterator HepMC::GenVertexParticleRange::end () `[inline]`

Definition at line 149 of file GenRanges.h.

References HepMC::GenVertex::particles_end().

The documentation for this class was generated from the following file:

- **GenRanges.h**

9.27 HepMC::HeavyIon Class Reference

The **HeavyIon** (p. 154) class stores information about heavy ions.

`#include <HeavyIon.h>`

Public Member Functions

- **HeavyIon ()**
  default constructor
- **HeavyIon (int nh, int np, int nt, int nc, int ns, int nsp, int nnw=0, int nwn=0, int nwnw=0, float im=0., float pl=0., float ec=0., float s=0.)**
  The first 6 values must be provided.
- **∼HeavyIon ()**
- **HeavyIon (HeavyIon const &orig)**
  copy constructor
- **HeavyIon & operator= (HeavyIon const &rhs)**
  make a copy
- **void swap (HeavyIon &other)**
  swap two HeavyIon (p. *154) objects*
- **bool operator== (const HeavyIon &) const**
  check for equality
- **bool operator!= (const HeavyIon &) const**
  check for inequality
- **int Ncoll_hard () const**
  Number of hard scatterings.
- **int Npart_proj () const**
  Number of projectile participants.
- **int Npart_targ () const**
  Number of target participants.
- **int Ncoll () const**
  Number of NN (nucleon-nucleon) collisions.
- **int spectator_neutrons () const**
  Number of spectator neutrons.
- **int spectator_protons () const**
  Number of spectator protons.
- **int N_Nwounded_collisions () const**

Number of N-Nwounded collisions.

- **int Nwounded_N_collisions () const**
  Number of Nwounded-N collisons.
- **int Nwounded_Nwounded_collisions () const**
  Number of Nwounded-Nwounded collisions.
- **float impact_parameter () const**
  Impact Parameter(in fm) of collision.
- **float event_plane_angle () const**
  Azimuthal angle of event plane.
- **float eccentricity () const**
- **float sigma_inel_NN () const**
  nucleon-nucleon inelastic (including diffractive) cross-section
- **bool is_valid () const**
  verify that the instance contains non-zero information
- **void set_Ncoll_hard (const int &i)**
  set number of hard scatterings
- **void set_Npart_proj (const int &i)**
  set number of projectile participants
- **void set_Npart_targ (const int &i)**
  set number of target participants
- **void set_Ncoll (const int &i)**
  set number of NN (nucleon-nucleon) collisions
- **void set_spectator_neutrons (const int &i)**
  set number of spectator neutrons
- **void set_spectator_protons (const int &i)**
  set number of spectator protons
- **void set_N_Nwounded_collisions (const int &i)**
  set number of N-Nwounded collisions
- **void set_Nwounded_N_collisions (const int &i)**
  set number of Nwounded-N collisons
- **void set_Nwounded_Nwounded_collisions (const int &i)**
  set number of Nwounded-Nwounded collisions
- **void set_impact_parameter (const float &f)**
  set Impact Parameter in fm

- **void set_event_plane_angle (const float &f)**
  set azimuthal angle of event plane
- **void set_eccentricity (const float &f)**
  set eccentricity of participating nucleons in the transverse plane
- **void set_sigma_inel_NN (const float &f)**
  set nucleon-nucleon inelastic cross-section

9.27.1 Detailed Description

The **HeavyIon** (p. 154) class stores information about heavy ions.

HepMC::HeavyIon (p. 154) provides additional information storage for Heavy Ion generators in **GenEvent** (p. 75). Creation and use of this information is optional.

Examples:

testMass.cc.in.

Definition at line 45 of file HeavyIon.h.

9.27.2 Constructor & Destructor Documentation

9.27.2.1 HepMC::HeavyIon::HeavyIon () [inline]

default constructor

Definition at line 51 of file HeavyIon.h.

9.27.2.2 HepMC::HeavyIon::HeavyIon (int *nh*, int *np*, int *nt*, int *nc*, int *ns*, int *nsp*, int *nnw* = 0, int *nwn* = 0, int *nwnw* = 0, float *im* = 0., float *pl* = 0., float *ec* = 0., float *s* = 0.) [inline]

The first 6 values must be provided.

Required members are the number of hard scatterings, the number of projectile participants. the number of target participants. the number of nucleon-nucleon collisions, the number of spectator neutrons, and the number of spectator protons.

Definition at line 178 of file HeavyIon.h.

9.27.2.3 HepMC::HeavyIon::~HeavyIon () [inline]

Definition at line 72 of file HeavyIon.h.

9.27.2.4 HepMC::HeavyIon::HeavyIon (HeavyIon const & *orig*) [inline]

copy constructor

Definition at line 196 of file HeavyIon.h.

9.27.3 Member Function Documentation

9.27.3.1 float HepMC::HeavyIon::eccentricity () const [inline]

eccentricity of participating nucleons in the transverse plane (as in phobos nucl-ex/0510031)

Definition at line 110 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.2 float HepMC::HeavyIon::event_plane_angle () const [inline]

Azimuthal angle of event plane.

Definition at line 107 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.3 float HepMC::HeavyIon::impact_parameter () const [inline]

Impact Parameter(in fm) of collision.

Definition at line 105 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.4 bool HepMC::HeavyIon::is_valid () const [inline]

verify that the instance contains non-zero information

Definition at line 260 of file HeavyIon.h.

9.27.3.5 int HepMC::HeavyIon::N_Nwounded_collisions () const [inline]

Number of N-Nwounded collisions.

Definition at line 99 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.6 int HepMC::HeavyIon::Ncoll () const [inline]

Number of NN (nucleon-nucleon) collisions.

Definition at line 93 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.7 int HepMC::HeavyIon::Ncoll_hard () const [inline]

Number of hard scatterings.

Definition at line 87 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.8 int HepMC::HeavyIon::Npart_proj () const [inline]

Number of projectile participants.

Definition at line 89 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.9 int HepMC::HeavyIon::Npart_targ () const [inline]

Number of target participants.

Definition at line 91 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.10 int HepMC::HeavyIon::Nwounded_N_collisions () const [inline]

Number of Nwounded-N collisons.

Definition at line 101 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.11 int HepMC::HeavyIon::Nwounded_Nwounded_collisions () const [inline]

Number of Nwounded-Nwounded collisions.

Definition at line 103 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.12 bool HepMC::HeavyIon::operator!= (const HeavyIon &) const [inline]

check for inequality

any nonmatching member generates inequality

Definition at line 254 of file HeavyIon.h.

9.27.3.13 HeavyIon & HepMC::HeavyIon::operator= (HeavyIon const & *rhs*) [inline]

make a copy

Definition at line 212 of file HeavyIon.h.

References swap().

9.27.3.14 bool HepMC::HeavyIon::operator== (const HeavyIon &) const [inline]

check for equality

equality requires that each member match

Definition at line 236 of file HeavyIon.h.

References eccentricity(), event_plane_angle(), impact_parameter(), N_Nwounded_collisions(), Ncoll(), Ncoll_hard(), Npart_proj(), Npart_targ(), Nwounded_N_collisions(), Nwounded_Nwounded_collisions(), sigma_inel_NN(), spectator_neutrons(), and spectator_protons().

9.27.3.15 void HepMC::HeavyIon::set_eccentricity (const float & *f*) [inline]

set eccentricity of participating nucleons in the transverse plane

Definition at line 142 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.16 void HepMC::HeavyIon::set_event_plane_angle (const float & *f*) [inline]

set azimuthal angle of event plane

Definition at line 140 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.17 void HepMC::HeavyIon::set_impact_parameter (const float & *f*) [inline]

set Impact Parameter in fm

Definition at line 138 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.18 void HepMC::HeavyIon::set_N_Nwounded_collisions (const int & *i*) [inline]

set number of N-Nwounded collisions

Definition at line 131 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.19 void HepMC::HeavyIon::set_Ncoll (const int & *i*) [inline]

set number of NN (nucleon-nucleon) collisions

Definition at line 125 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.20 void HepMC::HeavyIon::set_Ncoll_hard (const int & *i*) [inline]

set number of hard scatterings

Definition at line 119 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.21 void HepMC::HeavyIon::set_Npart_proj (const int & *i*) [inline]

set number of projectile participants

Definition at line 121 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.22 void HepMC::HeavyIon::set_Npart_targ (const int & *i*) [inline]

set number of target participants

Definition at line 123 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.23 void HepMC::HeavyIon::set_Nwounded_N_collisions (const int & *i*) [inline]

set number of Nwounded-N collisons

Definition at line 133 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.24 void HepMC::HeavyIon::set_Nwounded_Nwounded_collisions (const int & *i*) [inline]

set number of Nwounded-Nwounded collisions

Definition at line 135 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.25 void HepMC::HeavyIon::set_sigma_inel_NN (const float & *f*) [inline]

set nucleon-nucleon inelastic cross-section

Definition at line 144 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.26 void HepMC::HeavyIon::set_spectator_neutrons (const int & *i*) [inline]

set number of spectator neutrons

Definition at line 127 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.27 void HepMC::HeavyIon::set_spectator_protons (const int & *i*) [inline]

set number of spectator protons

Definition at line 129 of file HeavyIon.h.

Referenced by HepMC::operator>>().

9.27.3.28 float HepMC::HeavyIon::sigma_inel_NN () const [inline]

nucleon-nucleon inelastic (including diffractive) cross-section

Definition at line 112 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.29 int HepMC::HeavyIon::spectator_neutrons () const [inline]

Number of spectator neutrons.

Definition at line 95 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.30 int HepMC::HeavyIon::spectator_protons () const [inline]

Number of spectator protons.

Definition at line 97 of file HeavyIon.h.

Referenced by HepMC::operator<<(), and operator==().

9.27.3.31 void HepMC::HeavyIon::swap (HeavyIon & *other*) [inline]

swap two **HeavyIon** (p. 154) objects

Definition at line 219 of file HeavyIon.h.

References m_eccentricity, m_event_plane_angle, m_impact_parameter, m_N_Nwounded_collisions, m_Ncoll, m_Ncoll_hard, m_Npart_proj, m_Npart_targ, m_Nwounded_N_collisions, m_Nwounded_-Nwounded_collisions, m_sigma_inel_NN, m_spectator_neutrons, and m_spectator_protons.

Referenced by operator=().

The documentation for this class was generated from the following file:

- **HeavyIon.h**

9.28 HepMC::HEPEVT_Wrapper Class Reference

Generic Wrapper for the fortran HEPEVT common block.

`#include <HEPEVT_Wrapper.h>`

Static Public Member Functions

- static void **print_hepevt (std::ostream &ostr=std::cout)**
  write information from HEPEVT common block
- **static void print_hepevt_particle (int index, std::ostream &ostr=std::cout)**
  write particle information to ostr
- **static bool is_double_precision ()**
  True if common block uses double.
- **static bool check_hepevt_consistency (std::ostream &ostr=std::cout)**
  check for problems with HEPEVT common block
- **static void zero_everything ()**
  set all entries in HEPEVT to zero
- **static int event_number ()**
  event number
- **static int number_entries ()**
  num entries in current evt
- **static int status (int index)**
  status code
- **static int id (int index)**
  PDG particle id.
- **static int first_parent (int index)**
  index of 1st mother
- **static int last_parent (int index)**
  index of last mother
- **static int number_parents (int index)**
  number of parents
- **static int first_child (int index)**
  index of 1st daughter
- **static int last_child (int index)**
  index of last daughter

- **static int number_children (int index)**
  number of children
- **static double px (int index)**
  X momentum.
- **static double py (int index)**
  Y momentum.
- **static double pz (int index)**
  Z momentum.
- **static double e (int index)**
  Energy.
- **static double m (int index)**
  generated mass
- **static double x (int index)**
  X Production vertex.
- **static double y (int index)**
  Y Production vertex.
- **static double z (int index)**
  Z Production vertex.
- **static double t (int index)**
  production time
- **static void set_event_number (int evtno)**
  set event number
- **static void set_number_entries (int noentries)**
  set number of entries in HEPEVT
- **static void set_status (int index, int status)**
  set particle status
- **static void set_id (int index, int id)**
  set particle ID
- **static void set_parents (int index, int firstparent, int lastparent)**
  define parents of a particle
- **static void set_children (int index, int firstchild, int lastchild)**
  define children of a particle
- **static void set_momentum (int index, double px, double py, double pz, double e)**
  set particle momentum

- **static void set_mass (int index, double mass)**
  set particle mass
- **static void set_position (int index, double x, double y, double z, double t)**
  set particle production vertex
- **static unsigned int sizeof_int ()**
  size of integer in bytes
- **static unsigned int sizeof_real ()**
  size of real in bytes
- **static int max_number_entries ()**
  size of common block
- **static void set_sizeof_int (unsigned int)**
  define size of integer
- **static void set_sizeof_real (unsigned int)**
  define size of real
- **static void set_max_number_entries (unsigned int)**
  define size of common block

Static Protected Member Functions

- static double **byte_num_to_double (unsigned int)**
  navigate a byte array
- **static int byte_num_to_int (unsigned int)**
  navigate a byte array
- **static void write_byte_num (double, unsigned int)**
  pretend common block is an array of bytes
- **static void write_byte_num (int, unsigned int)**
  pretend common block is an array of bytes
- **static void print_legend (std::ostream &ostr=std::cout)**
  print output legend

9.28.1 Detailed Description

Generic Wrapper for the fortran HEPEVT common block.

This class is intended for static use only - it makes no sense to instantiate it.

Definition at line 130 of file HEPEVT_Wrapper.h.

9.28.2 Member Function Documentation

9.28.2.1 double HepMC::HEPEVT_Wrapper::byte_num_to_double (unsigned *int*) [inline, static, protected]

navigate a byte array

Definition at line 255 of file HEPEVT_Wrapper.h.

References hepevt, and hepevt_bytes_allocation.

Referenced by e(), m(), px(), py(), pz(), t(), x(), y(), and z().

9.28.2.2 int HepMC::HEPEVT_Wrapper::byte_num_to_int (unsigned *int*) [inline, static, protected]

navigate a byte array

Definition at line 273 of file HEPEVT_Wrapper.h.

References hepevt, and hepevt_bytes_allocation.

Referenced by event_number(), first_child(), first_parent(), id(), last_child(), last_parent(), number_entries(), and status().

9.28.2.3 bool HepMC::HEPEVT_Wrapper::check_hepevt_consistency (std::ostream & *ostr* = std::cout) [static]

check for problems with HEPEVT common block

This method inspects the HEPEVT common block and looks for inconsistencies in the mother/daughter pointers

Definition at line 88 of file HEPEVT_Wrapper.cc.

References event_number(), first_child(), first_parent(), last_child(), last_parent(), m(), number_entries(), print_hepevt_particle(), and print_legend().

9.28.2.4 double HepMC::HEPEVT_Wrapper::e (int *index*) [inline, static]

Energy.

Definition at line 446 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.5 int HepMC::HEPEVT_Wrapper::event_number () [inline, static]

event number

Definition at line 343 of file HEPEVT_Wrapper.h.

References byte_num_to_int().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(),

check_hepevt_consistency(), HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_HEPEVT::fill_-next_event(), and print_hepevt().

9.28.2.6 int HepMC::HEPEVT_Wrapper::first_child (int *index*) [inline, static]

index of 1st daughter

Definition at line 394 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), max_number_entries(), number_entries(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), check_hepevt_consistency(), last_child(), number_children(), print_hepevt_particle(), HepMC::IO_-HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.7 int HepMC::HEPEVT_Wrapper::first_parent (int *index*) [inline, static]

index of 1st mother

Definition at line 362 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), max_number_entries(), number_entries(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_-production_vertex(), check_hepevt_consistency(), HepMC::IO_HERWIG::fill_next_event(), last_parent(), number_parents(), print_hepevt_particle(), HepMC::IO_HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.8 int HepMC::HEPEVT_Wrapper::id (int *index*) [inline, static]

PDG particle id.

Definition at line 356 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), max_number_entries(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), Hep-MC::IO_HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.9 bool HepMC::HEPEVT_Wrapper::is_double_precision () [inline, static]

True if common block uses double.

Definition at line 337 of file HEPEVT_Wrapper.h.

References sizeof_real().

Referenced by print_hepevt().

9.28.2.10 int HepMC::HEPEVT_Wrapper::last_child (int *index*) [inline, static]

index of last daughter

Definition at line 402 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), first_child(), max_number_entries(), number_entries(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_-vertex(), check_hepevt_consistency(), number_children(), print_hepevt_particle(), HepMC::IO_-HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.11 int HepMC::HEPEVT_Wrapper::last_parent (int *index*) [inline, static]

index of last mother

Definition at line 370 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), first_parent(), max_number_entries(), number_entries(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_-production_vertex(), check_hepevt_consistency(), number_parents(), print_hepevt_particle(), Hep-MC::IO_HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.12 double HepMC::HEPEVT_Wrapper::m (int *index*) [inline, static]

generated mass

Definition at line 452 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), check_-hepevt_consistency(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.13 int HepMC::HEPEVT_Wrapper::max_number_entries () [inline, static]

size of common block

Definition at line 229 of file HEPEVT_Wrapper.h.

Referenced by e(), first_child(), first_parent(), id(), last_child(), last_parent(), m(), number_entries(), print_hepevt(), px(), py(), pz(), set_children(), set_id(), set_mass(), set_momentum(), set_parents(), set_-position(), set_status(), t(), HepMC::IO_HEPEVT::write_event(), x(), y(), z(), zero_everything(), and Hep-MC::IO_HERWIG::zero_hepevt_entry().

9.28.2.14 int HepMC::HEPEVT_Wrapper::number_children (int *index*) [inline, static]

number of children

Definition at line 420 of file HEPEVT_Wrapper.h.

References first_child(), and last_child().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), and HepMC::IO_HEPEVT::build_end_-vertex().

9.28.2.15 int HepMC::HEPEVT_Wrapper::number_entries () [inline, static]

num entries in current evt

Definition at line 346 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), max_number_entries(), and sizeof_int().

Referenced by check_hepevt_consistency(), HepMC::IO_HERWIG::fill_next_event(), HepMC::IO_-HEPEVT::fill_next_event(), first_child(), first_parent(), last_child(), last_parent(), print_hepevt(), Hep-MC::IO_HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.16 int HepMC::HEPEVT_Wrapper::number_parents (int *index*) [inline, static]

number of parents

Definition at line 388 of file HEPEVT_Wrapper.h.

References first_parent(), and last_parent().

Referenced by HepMC::IO_HERWIG::build_production_vertex(), and HepMC::IO_HEPEVT::build_-production_vertex().

9.28.2.17 void HepMC::HEPEVT_Wrapper::print_hepevt (std::ostream & *ostr* = std::cout) [static]

write information from HEPEVT common block

dumps the content of this HEPEVT event to ostr (Width is 80)

Examples:

fio/example_MyHerwig.cc.

Definition at line 27 of file HEPEVT_Wrapper.cc.

References event_number(), is_double_precision(), max_number_entries(), number_entries(), print_-hepevt_particle(), print_legend(), sizeof_int(), and sizeof_real().

Referenced by main().

9.28.2.18 void HepMC::HEPEVT_Wrapper::print_hepevt_particle (int *index*, std::ostream & *ostr* = std::cout) [static]

write particle information to ostr

dumps the content HEPEVT particle entry i (Width is 120) here i is the C array index (i.e. it starts at 0 ... whereas the fortran array index starts at 1) So if there's 100 particles, the last valid index is 100-1=99

Definition at line 68 of file HEPEVT_Wrapper.cc.

References e(), first_child(), first_parent(), last_child(), last_parent(), m(), px(), py(), pz(), status(), t(), x(), y(), and z().

Referenced by check_hepevt_consistency(), and print_hepevt().

9.28.2.19 void HepMC::HEPEVT_Wrapper::print_legend (std::ostream & *ostr* = std::cout) [static, protected]

print output legend

Definition at line 55 of file HEPEVT_Wrapper.cc.

Referenced by check_hepevt_consistency(), and print_hepevt().

9.28.2.20 double HepMC::HEPEVT_Wrapper::px (int *index*) [inline, static]

X momentum.

Definition at line 427 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.21 double HepMC::HEPEVT_Wrapper::py (int *index*) [inline, static]

Y momentum.

Definition at line 433 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.22 double HepMC::HEPEVT_Wrapper::pz (int *index*) [inline, static]

Z momentum.

Definition at line 440 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.23 void HepMC::HEPEVT_Wrapper::set_children (int *index*, int *firstchild*, int *lastchild*) [inline, static]

define children of a particle

Definition at line 514 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HERWIG::repair_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.24 void HepMC::HEPEVT_Wrapper::set_event_number (int *evtno*) [inline, static]

set event number

Definition at line 486 of file HEPEVT_Wrapper.h.

References write_byte_num().

Referenced by HepMC::IO_HEPEVT::write_event(), and zero_everything().

9.28.2.25 void HepMC::HEPEVT_Wrapper::set_id (int *index*, int *id*) [inline, static]

set particle ID

Definition at line 498 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HERWIG::repair_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.26 void HepMC::HEPEVT_Wrapper::set_mass (int *index*, double *mass*) [inline, static]

set particle mass

Definition at line 538 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), sizeof_real(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.27 void HepMC::HEPEVT_Wrapper::set_max_number_entries (unsigned *int*) [inline, static]

define size of common block

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, and fio/testPythiaCopies.cc.

Definition at line 251 of file HEPEVT_Wrapper.h.

Referenced by event_selection(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), and writePythiaStreamIO().

9.28.2.28 void HepMC::HEPEVT_Wrapper::set_momentum (int *index*, double *px*, double *py*, double *pz*, double *e*) [inline, static]

set particle momentum

Definition at line 524 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), sizeof_real(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.29 void HepMC::HEPEVT_Wrapper::set_number_entries (int *noentries*) [inline, static]

set number of entries in HEPEVT

Definition at line 489 of file HEPEVT_Wrapper.h.

References sizeof_int(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HEPEVT::write_event(), and zero_everything().

9.28.2.30 void HepMC::HEPEVT_Wrapper::set_parents (int *index*, int *firstparent*, int *lastparent*) [inline, static]

define parents of a particle

Definition at line 504 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HERWIG::repair_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.31 void HepMC::HEPEVT_Wrapper::set_position (int *index*, double *x*, double *y*, double *z*, double *t*) [inline, static]

set particle production vertex

Definition at line 545 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), sizeof_real(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.32 void HepMC::HEPEVT_Wrapper::set_sizeof_int (unsigned *int*) [inline, static]

define size of integer

Definition at line 232 of file HEPEVT_Wrapper.h.

9.28.2.33 void HepMC::HEPEVT_Wrapper::set_sizeof_real (unsigned *int*) [inline, static]

define size of real

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, and fio/testPythiaCopies.cc.

Definition at line 242 of file HEPEVT_Wrapper.h.

Referenced by event_selection(), main(), pythia_in_out(), pythia_out(), pythia_particle_out(), and write-PythiaStreamIO().

9.28.2.34 void HepMC::HEPEVT_Wrapper::set_status (int *index*, int *status*) [inline, static]

set particle status

Definition at line 492 of file HEPEVT_Wrapper.h.

References max_number_entries(), sizeof_int(), and write_byte_num().

Referenced by HepMC::IO_HERWIG::remove_gaps_in_hepevt(), HepMC::IO_HEPEVT::write_event(), zero_everything(), and HepMC::IO_HERWIG::zero_hepevt_entry().

9.28.2.35 unsigned int HepMC::HEPEVT_Wrapper::sizeof_int () [inline, static]

size of integer in bytes

Definition at line 225 of file HEPEVT_Wrapper.h.

Referenced by e(), first_child(), first_parent(), id(), last_child(), last_parent(), m(), number_entries(), print_hepevt(), px(), py(), pz(), set_children(), set_id(), set_mass(), set_momentum(), set_number_entries(), set_parents(), set_position(), set_status(), status(), t(), x(), y(), and z().

9.28.2.36 unsigned int HepMC::HEPEVT_Wrapper::sizeof_real () [inline, static]

size of real in bytes

Definition at line 227 of file HEPEVT_Wrapper.h.

Referenced by e(), is_double_precision(), m(), print_hepevt(), px(), py(), pz(), set_mass(), set_momentum(), set_position(), t(), x(), y(), and z().

9.28.2.37 int HepMC::HEPEVT_Wrapper::status (int *index*) [inline, static]

status code

Definition at line 353 of file HEPEVT_Wrapper.h.

References byte_num_to_int(), and sizeof_int().

Referenced by HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), HepMC::IO_HERWIG::fill_next_event(), print_hepevt_particle(), HepMC::IO_HERWIG::remove_gaps_in_hepevt(), and HepMC::IO_HERWIG::repair_hepevt().

9.28.2.38 double HepMC::HEPEVT_Wrapper::t (int *index*) [inline, static]

production time

Definition at line 479 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.39 void HepMC::HEPEVT_Wrapper::write_byte_num (int, unsigned *int*) [inline, static, protected]

pretend common block is an array of bytes

Definition at line 312 of file HEPEVT_Wrapper.h.

References hepevt, and hepevt_bytes_allocation.

9.28.2.40 void HepMC::HEPEVT_Wrapper::write_byte_num (double, unsigned *int*) [inline, static, protected]

pretend common block is an array of bytes

Definition at line 295 of file HEPEVT_Wrapper.h.

References hepevt, and hepevt_bytes_allocation.

Referenced by set_children(), set_event_number(), set_id(), set_mass(), set_momentum(), set_number_entries(), set_parents(), set_position(), and set_status().

9.28.2.41 double HepMC::HEPEVT_Wrapper::x (int *index*) [inline, static]

X Production vertex.

Definition at line 458 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.42 double HepMC::HEPEVT_Wrapper::y (int *index*) [inline, static]

Y Production vertex.

Definition at line 465 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.43 double HepMC::HEPEVT_Wrapper::z (int *index*) [inline, static]

Z Production vertex.

Definition at line 472 of file HEPEVT_Wrapper.h.

References byte_num_to_double(), max_number_entries(), sizeof_int(), and sizeof_real().

Referenced by HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), print_hepevt_particle(), and HepMC::IO_HERWIG::remove_gaps_in_hepevt().

9.28.2.44 void HepMC::HEPEVT_Wrapper::zero_everything () [static]

set all entries in HEPEVT to zero

Definition at line 212 of file HEPEVT_Wrapper.cc.

References max_number_entries(), set_children(), set_event_number(), set_id(), set_mass(), set_momentum(), set_number_entries(), set_parents(), set_position(), and set_status().

The documentation for this class was generated from the following files:

- **HEPEVT_Wrapper.h**
- **HEPEVT_Wrapper.cc**

9.29 hwgev Struct Reference

`#include <HerwigWrapper.h>`

Public Attributes

- double **AVWGT**
- **double EVWGT**
- **double GAMWT**
- **double TLOUT**
- **double WBIGST**
- **double WGTMAX**
- **double WGTSUM**
- **double WSQSUM**
- **int IDHW [herwig_hepevt_size]**
- **int IERROR**
- **int ISTAT**
- **int LWEVT**
- **int MAXER**
- **int MAXPR**
- **int NOWGT**
- **int NRN [2]**
- **int NUMER**
- **int NUMERU**
- **int NWGTS**
- **int GENSOF**

9.29.1 Detailed Description

Definition at line 56 of file HerwigWrapper.h.

9.29.2 Member Data Documentation

9.29.2.1 double hwgev::AVWGT

Definition at line 57 of file HerwigWrapper.h.

9.29.2.2 double hwgev::EVWGT

Definition at line 57 of file HerwigWrapper.h.

9.29.2.3 double hwgev::GAMWT

Definition at line 57 of file HerwigWrapper.h.

9.29.2.4 int hwgev::GENSOF

Definition at line 60 of file HerwigWrapper.h.

9.29.2.5 int hwgev::IDHW[herwig_hepevt_size]

Definition at line 58 of file HerwigWrapper.h.

9.29.2.6 int hwgev::IERROR

Definition at line 58 of file HerwigWrapper.h.

9.29.2.7 int hwgev::ISTAT

Definition at line 58 of file HerwigWrapper.h.

9.29.2.8 int hwgev::LWEVT

Definition at line 58 of file HerwigWrapper.h.

9.29.2.9 int hwgev::MAXER

Definition at line 58 of file HerwigWrapper.h.

9.29.2.10 int hwgev::MAXPR

Definition at line 58 of file HerwigWrapper.h.

9.29.2.11 int hwgev::NOWGT

Definition at line 59 of file HerwigWrapper.h.

9.29.2.12 int hwgev::NRN[2]

Definition at line 59 of file HerwigWrapper.h.

9.29.2.13 int hwgev::NUMER

Definition at line 59 of file HerwigWrapper.h.

9.29.2.14 int hwgev::NUMERU

Definition at line 59 of file HerwigWrapper.h.

9.29.2.15 int hwgev::NWGTS

Definition at line 59 of file HerwigWrapper.h.

9.29.2.16 double hwgev::TLOUT

Definition at line 57 of file HerwigWrapper.h.

9.29.2.17 double hwgev::WBIGST

Definition at line 57 of file HerwigWrapper.h.

9.29.2.18 double hwgev::WGTMAX

Definition at line 57 of file HerwigWrapper.h.

9.29.2.19 double hwgev::WGTSUM

Definition at line 57 of file HerwigWrapper.h.

9.29.2.20 double hwgev::WSQSUM

Definition at line 57 of file HerwigWrapper.h.

The documentation for this struct was generated from the following file:

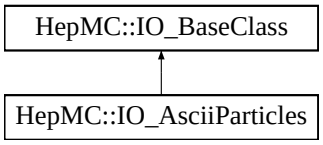
- **HerwigWrapper.h**

9.30 HepMC::IO_AsciiParticles Class Reference

event input/output in ascii format for eye and machine reading

`#include <IO_AsciiParticles.h>`

Inheritance diagram for HepMC::IO_AsciiParticles::



Public Member Functions

- **IO_AsciiParticles (const char ∗filename="IO_AsciiParticles.dat", std::ios::openmode mode=std::ios::out)**

  constructor requiring a file name and std::ios mode

- **virtual ∼IO_AsciiParticles ()**
- **void write_event (const GenEvent ∗evt)**

  write this event

- **bool fill_next_event (GenEvent ∗evt)**

  get the next event

- **void write_comment (const std::string comment)**
- **void setPrecision (int iprec)**

  set output precision

- **int rdstate () const**

  check the state of the IO stream

- **void clear ()**

  clear the IO stream

- **void print (std::ostream &ostr=std::cout) const**

  write to ostr

Protected Member Functions

- bool **write_end_listing ()**

  write end tag

9.30.1 Detailed Description

event input/output in ascii format for eye and machine reading

Strategy for reading or writing events as machine readable ascii to a file. When instantiating, the mode of file to be created must be specified.

Examples:

fio/example_MyPythia.cc, testHepMC.cc.in, and testStreamIO.cc.in.

Definition at line 54 of file IO_AsciiParticles.h.

9.30.2 Constructor & Destructor Documentation

9.30.2.1 HepMC::IO_AsciiParticles::IO_AsciiParticles (const char * *filename* = "IO_AsciiParticles.dat", std::ios::openmode *mode* = std::ios::out)

constructor requiring a file name and std::ios mode

Definition at line 17 of file IO_AsciiParticles.cc.

9.30.2.2 HepMC::IO_AsciiParticles::~IO_AsciiParticles () [virtual]

Definition at line 46 of file IO_AsciiParticles.cc.

9.30.3 Member Function Documentation

9.30.3.1 void HepMC::IO_AsciiParticles::clear () [inline]

clear the IO stream

Definition at line 97 of file IO_AsciiParticles.h.

9.30.3.2 bool HepMC::IO_AsciiParticles::fill_next_event (GenEvent * *evt*) [virtual]

get the next event

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_f1dffb95a44d521af510f6431a30f942}$ **??**)

Definition at line 179 of file IO_AsciiParticles.cc.

9.30.3.3 void HepMC::IO_AsciiParticles::print (std::ostream & *ostr* = std::cout) const [virtual]

write to ostr

Reimplemented from **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_8a23f5de9c6bb10931dcacdeb7677413}$ **??**)

Definition at line 53 of file IO_AsciiParticles.cc.

9.30.3.4 int HepMC::IO_AsciiParticles::rdstate () const [inline]

check the state of the IO stream

Definition at line 96 of file IO_AsciiParticles.h.

9.30.3.5 void HepMC::IO_AsciiParticles::setPrecision (int *iprec*) [inline]

set output precision

Definition at line 98 of file IO_AsciiParticles.h.

9.30.3.6 void HepMC::IO_AsciiParticles::write_comment (const std::string *comment*)

insert a comment directly into the output file — normally you only want to do this at the beginning or end of the file. All comments are preceded with "HepMC::IO_AsciiParticles-COMMENT\n"

Definition at line 202 of file IO_AsciiParticles.cc.

References write_end_listing().

9.30.3.7 bool HepMC::IO_AsciiParticles::write_end_listing () [protected]

write end tag

Definition at line 217 of file IO_AsciiParticles.cc.

Referenced by write_comment().

9.30.3.8 void HepMC::IO_AsciiParticles::write_event (const GenEvent * *evt*) [virtual]

write this event

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_7}929dfd8412207c7904f29652810c1f4$ **??**)

Definition at line 63 of file IO_AsciiParticles.cc.

References HepMC::GenEvent::alphaQCD(), HepMC::GenEvent::alphaQED(), HepMC::GenVertex::barcode(), HepMC::WeightContainer::begin(), HepMC::WeightContainer::end(), HepMC::GenEvent::event_number(), HepMC::GenEvent::event_scale(), HepMC::detail::output(), HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::GenEvent::particles_size(), HepMC::GenEvent::random_states(), HepMC::GenEvent::signal_process_id(), HepMC::GenEvent::signal_process_vertex(), HepMC::WeightContainer::size(), HepMC::versionName(), HepMC::GenEvent::vertices_size(), and HepMC::GenEvent::weights().

The documentation for this class was generated from the following files:

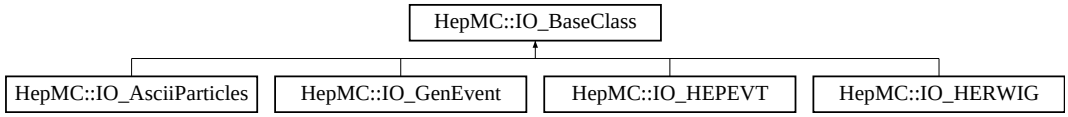
- **IO_AsciiParticles.h**
- **IO_AsciiParticles.cc**

9.31 HepMC::IO_BaseClass Class Reference

all input/output classes inherit from **IO_BaseClass** (p. 181)

`#include <IO_BaseClass.h>`

Inheritance diagram for HepMC::IO_BaseClass::



Public Member Functions

- virtual ~**IO_BaseClass** ()
- **virtual void write_event (const GenEvent ∗)=0**

  write this GenEvent (p. ***75)***

- **virtual bool fill_next_event (GenEvent ∗)=0**

  fill this GenEvent (p. ***75)***

- **virtual void print (std::ostream &ostr=std::cout) const**

  write output to ostr

- **GenEvent ∗ read_next_event ()**

  do not over-ride

- **virtual GenEvent ∗& operator>> (GenEvent ∗&)**

  the same as read_next_event

- **virtual const GenEvent ∗& operator<< (const GenEvent ∗&)**

  the same as write_event

- **virtual GenEvent ∗& operator<< (GenEvent ∗&)**

  the same as write_event

9.31.1 Detailed Description

all input/output classes inherit from **IO_BaseClass** (p. 181)

If you want to write a new IO class, then inherit from this class and re-define read_event() and **write_event()** (p. 183)

Definition at line 34 of file IO_BaseClass.h.

9.31.2 Constructor & Destructor Documentation

9.31.2.1 virtual HepMC::IO_BaseClass::~IO_BaseClass () [inline, virtual]

Definition at line 36 of file IO_BaseClass.h.

9.31.3 Member Function Documentation

9.31.3.1 virtual bool HepMC::IO_BaseClass::fill_next_event (GenEvent *) [pure virtual]

fill this **GenEvent** (p. 75)

Implemented in **HepMC::IO_AsciiParticles p.** (classHepMC$_{11}IO_{AsciiParticles_fdd859891c2ac09f8758d081357c17ce}$ **??**)**HepMC**

the same as write_event

Definition at line 105 of file IO_BaseClass.h.

References write_event().

9.31.3.3 const GenEvent *& HepMC::IO_BaseClass::operator<< (const GenEvent *&) [inline, virtual]

the same as write_event

Definition at line 99 of file IO_BaseClass.h.

References write_event().

9.31.3.4 GenEvent *& HepMC::IO_BaseClass::operator>> (GenEvent *&) [inline, virtual]

the same as read_next_event

Definition at line 94 of file IO_BaseClass.h.

References read_next_event().

9.31.3.5 void HepMC::IO_BaseClass::print (std::ostream & *ostr* = std::cout) const [inline, virtual]

write output to ostr

Reimplemented in **HepMC::IO_AsciiParticles p.** (classHepMC$_{11}IO_{AsciiParticles_2c9bebc0be07d8b946ff8c27bf2d0636}$ **??**)**HepM**

do not over-ride

creates a new event and fills it by calling the sister method read_next_event(GenEvent*)

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, and test-MultipleCopies.cc.in.

Definition at line 74 of file IO_BaseClass.h.

References fill_next_event().

Referenced by event_selection(), main(), operator>>(), pythia_in(), pythia_in_out(), pythia_out(), pythia_particle_out(), and writePythiaStreamIO().

9.31.3.7 virtual void HepMC::IO_BaseClass::write_event (const GenEvent *) [pure virtual]

write this **GenEvent** (p. 75)

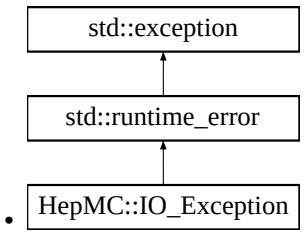
Implemented in **HepMC::IO_AsciiParticles p.** (classHepMC$_{11}IO_{A}sciiParticles_{2}d45b9474967ec0cdeccece86d5d8c5e$ **??**)**HepMC**

9.32 HepMC::IO_Exception Class Reference

IO exception handling.

`#include <IO_Exception.h>`

Inheritance diagram for HepMC::IO_Exception::



Public Types

- **OK**
- **NullEvent**
- **WrongFileType**
- **MissingStartKey**
- **EndOfStream**
- **EndKeyMismatch**
- **MissingEndKey**
- **InvalidData**
- **InputAndOutput**
- **BadOutputStream**
- **BadInputStream**
- **enum ErrorType {**
  OK, NullEvent, WrongFileType, MissingStartKey,
  EndOfStream, EndKeyMismatch, MissingEndKey, InvalidData,
  InputAndOutput, BadOutputStream, BadInputStream }
  IO error types.

Public Member Functions

- **IO_Exception (const std::string &msg)**

9.32.1 Detailed Description

IO exception handling.

IO_GenEvent (p. 186), etc. catch the throw and set data members with the error type and message Some of the messages are constructed with transient information (e.g., contents of a bad **GenParticle** (p. 113))

Examples:

testStreamIO.cc.in.

Definition at line 28 of file IO_Exception.h.

9.32.2 Member Enumeration Documentation

9.32.2.1 enum HepMC::IO_Exception::ErrorType

IO error types.

Enumerator:

OK
NullEvent
WrongFileType
MissingStartKey
EndOfStream
EndKeyMismatch
MissingEndKey
InvalidData
InputAndOutput
BadOutputStream
BadInputStream

Definition at line 34 of file IO_Exception.h.

9.32.3 Constructor & Destructor Documentation

9.32.3.1 HepMC::IO_Exception::IO_Exception (const std::string & *msg*) [inline]

Definition at line 30 of file IO_Exception.h.

The documentation for this class was generated from the following file:

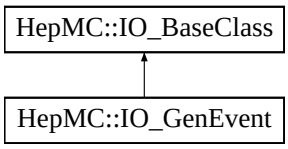
- **IO_Exception.h**

9.33 HepMC::IO_GenEvent Class Reference

IO_GenEvent (p. 186) also deals with **HeavyIon** (p. 154) and **PdfInfo** (p. 222).

`#include <IO_GenEvent.h>`

Inheritance diagram for HepMC::IO_GenEvent::



Public Member Functions

- **IO_GenEvent (const std::string &filename="IO_GenEvent.dat", std::ios::openmode mode=std::ios::out)**
  constructor requiring a file name and std::ios mode
- **IO_GenEvent (std::istream &)**
  constructor requiring an input stream
- **IO_GenEvent (std::ostream &)**
  constructor requiring an output stream
- **virtual ~IO_GenEvent ()**
- **void write_event (const GenEvent *evt)**
  write this event
- **bool fill_next_event (GenEvent *evt)**
  get the next event
- **void write_comment (const std::string comment)**
- **int rdstate () const**
  check the state of the IO stream
- **void clear ()**
  clear the IO stream
- **void print (std::ostream &ostr=std::cout) const**
  write to ostr
- **void use_input_units (Units::MomentumUnit, Units::LengthUnit)**
- **void precision (int)**
- **int error_type () const**
  integer (enum) associated with read error
- **const std::string & error_message () const**
  the read error message string

9.33.1 Detailed Description

IO_GenEvent (p. 186) also deals with **HeavyIon** (p. 154) and **PdfInfo** (p. 222).

event input/output in ascii format for machine reading extended format contains **HeavyIon** (p. 154) and **PdfInfo** (p. 222) classes

Strategy for reading or writing events using iostreams When instantiating with a file name, the mode of file to be created must be specified. Options are: std::ios::in open file for input std::ios::out open file for output std::ios::trunc erase old file when opening (i.e. ios::out|iostrunc removes oldfile, and creates a new one for output) std::ios::app append output to end of file for the purposes of this class, simultaneous input and output mode (std::ios::in | std::ios::out) is not allowed.

Event listings are preceded by the key: "HepMC::IO_GenEvent-START_EVENT_LISTING\n" and terminated by the key: "HepMC::IO_GenEvent-END_EVENT_LISTING\n" **GenParticle** (p. 113) Data tables are preceded by the key: "HepMC::IO_GenEvent-START_PARTICLE_DATA\n" and terminated by the key: "HepMC::IO_GenEvent-END_PARTICLE_DATA\n" Comments are allowed. They need not be preceded by anything, though if a comment is written using write_comment(const string) then it will be preceded by "HepMC::IO_GenEvent-COMMENT\n" Each event, vertex, particle, particle data, heavy ion, or pdf info line is preceded by "E ","V ","P ","D ","H ","F " respectively. Comments may appear anywhere in the file – so long as they do not contain any of the start/stop keys.

Examples:

example_EventSelection.cc, example_UsingIterators.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, testFlow.cc, testHepMC.cc.in, testHepMCIteration.cc.in, testMass.cc.in, testMultipleCopies.cc.in, and testStreamIO.cc.in.

Definition at line 63 of file IO_GenEvent.h.

9.33.2 Constructor & Destructor Documentation

9.33.2.1 HepMC::IO_GenEvent::IO_GenEvent (const std::string & *filename* = "IO_GenEvent.dat", std::ios::openmode *mode* = std::ios::out)

constructor requiring a file name and std::ios mode

Definition at line 16 of file IO_GenEvent.cc.

References HepMC::detail::establish_input_stream_info(), HepMC::detail::establish_output_stream_info(), and HepMC::IO_Exception::InputAndOutput.

9.33.2.2 HepMC::IO_GenEvent::IO_GenEvent (std::istream &)

constructor requiring an input stream

Definition at line 50 of file IO_GenEvent.cc.

References HepMC::detail::establish_input_stream_info().

9.33.2.3 HepMC::IO_GenEvent::IO_GenEvent (std::ostream &)

constructor requiring an output stream

Definition at line 61 of file IO_GenEvent.cc.

References HepMC::detail::establish_output_stream_info().

9.33.2.4 HepMC::IO_GenEvent::~IO_GenEvent () [virtual]

Definition at line 72 of file IO_GenEvent.cc.

References HepMC::write_HepMC_IO_block_end().

9.33.3 Member Function Documentation

9.33.3.1 void HepMC::IO_GenEvent::clear () [inline]

clear the IO stream

Definition at line 133 of file IO_GenEvent.h.

9.33.3.2 const std::string & HepMC::IO_GenEvent::error_message () const [inline]

the read error message string

Definition at line 145 of file IO_GenEvent.h.

9.33.3.3 int HepMC::IO_GenEvent::error_type () const [inline]

integer (enum) associated with read error

Definition at line 141 of file IO_GenEvent.h.

9.33.3.4 bool HepMC::IO_GenEvent::fill_next_event (GenEvent * *evt*) [virtual]

get the next event

Implements **HepMC::IO_BaseClass p.** (classHepMC_11$IO_{BaseClass_f1dffb95a44d521af510f6431a30f942}$ **??**)

Definition at line 109 of file IO_GenEvent.cc.

References HepMC::GenEvent::clear(), HepMC::IO_Exception::InvalidData, HepMC::GenEvent::is_valid(), HepMC::IO_Exception::NullEvent, HepMC::IO_Exception::OK, and HepMC::IO_Exception::WrongFileType.

9.33.3.5 void HepMC::IO_GenEvent::precision (int)

set output precision The default precision is 16.

Definition at line 96 of file IO_GenEvent.cc.

9.33.3.6 void HepMC::IO_GenEvent::print (std::ostream & *ostr* = std::cout) const [virtual]

write to ostr

Reimplemented from **HepMC::IO_BaseClass p.** (classHepMC_11$IO_{BaseClass_8a23f5de9c6bb10931dcacdeb7677413}$ **??**)

Definition at line 86 of file IO_GenEvent.cc.

9.33.3.7 int HepMC::IO_GenEvent::rdstate () const [inline]

check the state of the IO stream

Definition at line 123 of file IO_GenEvent.h.

Referenced by main().

9.33.3.8 void HepMC::IO_GenEvent::use_input_units (Units::MomentumUnit, Units::LengthUnit)

needed when reading a file without units if those units are different than the declared default units (e.g., the default units are MeV, but the file was written with GeV) This method is not necessary if the units are written in the file

Definition at line 79 of file IO_GenEvent.cc.

References HepMC::set_input_units().

9.33.3.9 void HepMC::IO_GenEvent::write_comment (const std::string *comment*)

insert a comment directly into the output file — normally you only want to do this at the beginning or end of the file. All comments are preceded with "HepMC::IO_GenEvent-COMMENT\n"

Definition at line 162 of file IO_GenEvent.cc.

References HepMC::write_HepMC_IO_block_end(), and HepMC::IO_Exception::WrongFileType.

9.33.3.10 void HepMC::IO_GenEvent::write_event (const GenEvent * *evt*) [virtual]

write this event

Writes evt to output stream. It does NOT delete the event after writing.

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_7}929dfd8412207c7904f29652810c1f4$ **??**)

Definition at line 143 of file IO_GenEvent.cc.

References HepMC::write_HepMC_IO_block_begin(), and HepMC::IO_Exception::WrongFileType.

The documentation for this class was generated from the following files:

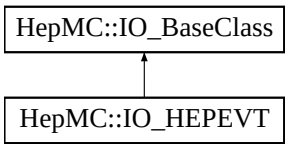
- **IO_GenEvent.h**
- **IO_GenEvent.cc**

9.34 HepMC::IO_HEPEVT Class Reference

HEPEVT IO class.

`#include <IO_HEPEVT.h>`

Inheritance diagram for HepMC::IO_HEPEVT::



Public Member Functions

- **IO_HEPEVT ()**
- **virtual ~IO_HEPEVT ()**
- **bool fill_next_event (GenEvent *)**
  *fill this GenEvent (*p. *75)*
- **void write_event (const GenEvent *)**
  *write this GenEvent (*p. *75)*
- **void print (std::ostream &ostr=std::cout) const**
  write output to ostr
- **bool trust_both_mothers_and_daughters () const**
  default is false
- **bool trust_mothers_before_daughters () const**
  default is true
- **bool print_inconsistency_errors () const**
  default is true
- **bool trust_beam_particles () const**
  default is true
- **void set_trust_mothers_before_daughters (bool b=true)**
  define mother daughter trust rules
- **void set_trust_both_mothers_and_daughters (bool b=false)**
  define mother daughter trust rules
- **void set_print_inconsistency_errors (bool b=true)**
- **void set_trust_beam_particles (bool b=true)**
  declare whether or not beam particles exist

Protected Member Functions

- **GenParticle * build_particle (int index)**

  *create a GenParticle (*p. *113)*

- **void build_production_vertex (int i, std::vector< HepMC::GenParticle * > &hepevt_particle, GenEvent *evt)**

  create a production vertex

- **void build_end_vertex (int i, std::vector< HepMC::GenParticle * > &hepevt_particle, GenEvent *evt)**

  create an end vertex

- **int find_in_map (const std::map< HepMC::GenParticle *, int > &m, GenParticle *p) const**

  find this particle in the particle map

9.34.1 Detailed Description

HEPEVT IO class.

IO class for reading the standard HEPEVT common block.

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyPythia.cc, fio/example_PythiaStream-IO.cc, and fio/testPythiaCopies.cc.

Definition at line 39 of file IO_HEPEVT.h.

9.34.2 Constructor & Destructor Documentation

9.34.2.1 HepMC::IO_HEPEVT::IO_HEPEVT ()

Definition at line 12 of file IO_HEPEVT.cc.

9.34.2.2 HepMC::IO_HEPEVT::~IO_HEPEVT () [virtual]

Definition at line 18 of file IO_HEPEVT.cc.

9.34.3 Member Function Documentation

9.34.3.1 void HepMC::IO_HEPEVT::build_end_vertex (int *i*, std::vector< HepMC::GenParticle * > & *hepevt_particle*, GenEvent * *evt*) [protected]

create an end vertex

for particle in HEPEVT with index i, build an end vertex if appropriate, and add that vertex to the event

Definition at line 257 of file IO_HEPEVT.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::first_child(), HepMC::HEPEVT_Wrapper::last_child(), HepMC::HEPEVT_Wrapper::number_children(), p, HepMC::GenVertex::position(), HepMC::GenVertex::set_position(), HepMC::HEPEVT_Wrapper::t(), HepMC::HEPEVT_Wrapper::x(), HepMC::HEPEVT_Wrapper::y(), and HepMC::HEPEVT_Wrapper::z().

Referenced by fill_next_event().

9.34.3.2 GenParticle * HepMC::IO_HEPEVT::build_particle (int *index*) [protected]

create a **GenParticle** (p. 113)

Builds a particle object corresponding to index in HEPEVT

Definition at line 325 of file IO_HEPEVT.cc.

References HepMC::HEPEVT_Wrapper::e(), HepMC::HEPEVT_Wrapper::id(), HepMC::HEPEVT_Wrapper::m(), p, HepMC::HEPEVT_Wrapper::px(), HepMC::HEPEVT_Wrapper::py(), HepMC::HEPEVT_Wrapper::pz(), and HepMC::HEPEVT_Wrapper::status().

Referenced by fill_next_event().

9.34.3.3 void HepMC::IO_HEPEVT::build_production_vertex (int *i*, std::vector< HepMC::GenParticle * > & *hepevt_particle*, GenEvent * *evt*) [protected]

create a production vertex

for particle in HEPEVT with index i, build a production vertex if appropriate, and add that vertex to the event

Definition at line 191 of file IO_HEPEVT.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::first_parent(), HepMC::HEPEVT_Wrapper::last_parent(), HepMC::HEPEVT_Wrapper::number_parents(), p, HepMC::GenVertex::position(), HepMC::GenVertex::set_position(), HepMC::HEPEVT_Wrapper::t(), HepMC::HEPEVT_Wrapper::x(), HepMC::HEPEVT_Wrapper::y(), and HepMC::HEPEVT_Wrapper::z().

Referenced by fill_next_event().

9.34.3.4 bool HepMC::IO_HEPEVT::fill_next_event (GenEvent *) [virtual]

fill this **GenEvent** (p. 75)

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_f1dffb95a44d521af510f6431a30f942}$ **??**)

Definition at line 31 of file IO_HEPEVT.cc.

References HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), build_end_vertex(), build_particle(), build_production_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::number_entries(), HepMC::GenEvent::set_beam_particles(), HepMC::GenEvent::set_event_number(), and trust_beam_particles().

9.34.3.5 int HepMC::IO_HEPEVT::find_in_map (const std::map< HepMC::GenParticle *, int > & *m*, GenParticle * *p*) const [protected]

find this particle in the particle map

Definition at line 340 of file IO_HEPEVT.cc.

References p.

Referenced by write_event().

9.34.3.6 void HepMC::IO_HEPEVT::print (std::ostream & *ostr* = std::cout) const [virtual]

write output to ostr

Reimplemented from **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_8a23f5de9c6bb10931dcacdeb7677413}$ **??**)

Definition at line 20 of file IO_HEPEVT.cc.

9.34.3.7 bool HepMC::IO_HEPEVT::print_inconsistency_errors () const [inline]

default is true

Definition at line 120 of file IO_HEPEVT.h.

9.34.3.8 void HepMC::IO_HEPEVT::set_print_inconsistency_errors (bool *b* = true) [inline]

Since HEPEVT has bi-directional pointers, it is possible that the mother/daughter pointers are inconsistent (though physically speaking this should never happen). In practise it happens often. When a conflict occurs (i.e. when mother/daughter pointers are in disagreement, where an empty (0) pointer is not considered a disagreement) an error is printed. These errors can be turned off with: myio_hepevt.set_print_inconsistency_errors(0); but it is STRONGLY recommended that you print the HEPEVT common and understand the inconsistency BEFORE you turn off the errors. The messages are there for a reason [remember, there is no message printed when the information is missing, ... only when is it inconsistent. User beware.] You can inspect the HEPEVT common block for inconsistencies with **HEPEVT_Wrapper::check_hepevt_consistency()** (p. 165)

There is a switch controlling whether the mother pointers or the daughters are to be trusted. For example, in Pythia the mother information is always correctly included, but the daughter information is often left unfilled: in this case we want to trust the mother pointers and not necessarily the daughters. [THIS IS THE DEFAULT]. Unfortunately the reverse happens for the stdhep(2001) translation of Isajet, so we need an option to toggle the choices.

Definition at line 129 of file IO_HEPEVT.h.

9.34.3.9 void HepMC::IO_HEPEVT::set_trust_beam_particles (bool *b* = true) [inline]

declare whether or not beam particles exist

Definition at line 135 of file IO_HEPEVT.h.

9.34.3.10 void HepMC::IO_HEPEVT::set_trust_both_mothers_and_daughters (bool *b* = false) [inline]

define mother daughter trust rules

Definition at line 123 of file IO_HEPEVT.h.

9.34.3.11 void HepMC::IO_HEPEVT::set_trust_mothers_before_daughters (bool *b* = true) [inline]

define mother daughter trust rules

Definition at line 126 of file IO_HEPEVT.h.

9.34.3.12 bool HepMC::IO_HEPEVT::trust_beam_particles () const [inline]

default is true

Definition at line 132 of file IO_HEPEVT.h.

Referenced by fill_next_event().

9.34.3.13 bool HepMC::IO_HEPEVT::trust_both_mothers_and_daughters () const [inline]

default is false

Definition at line 114 of file IO_HEPEVT.h.

9.34.3.14 bool HepMC::IO_HEPEVT::trust_mothers_before_daughters () const [inline]

default is true

Definition at line 117 of file IO_HEPEVT.h.

9.34.3.15 void HepMC::IO_HEPEVT::write_event (const GenEvent *) [virtual]

write this **GenEvent** (p. 75)

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_7}929dfd8412207c7904f29652810c1f4$ **??**)

Definition at line 110 of file IO_HEPEVT.cc.

References HepMC::FourVector::e(), HepMC::GenEvent::event_number(), find_in_map(), HepMC::HEPEVT_Wrapper::max_number_entries(), p, HepMC::FourVector::px(), HepMC::FourVector::py(), HepMC::FourVector::pz(), HepMC::HEPEVT_Wrapper::set_children(), HepMC::HEPEVT_Wrapper::set_event_number(), HepMC::HEPEVT_Wrapper::set_id(), HepMC::HEPEVT_Wrapper::set_mass(), HepMC::HEPEVT_Wrapper::set_momentum(), HepMC::HEPEVT_Wrapper::set_number_entries(), HepMC::HEPEVT_Wrapper::set_parents(), HepMC::HEPEVT_Wrapper::set_position(), HepMC::HEPEVT_Wrapper::set_status(), v, HepMC::GenEvent::vertices_begin(), and HepMC::GenEvent::vertices_end().

The documentation for this class was generated from the following files:

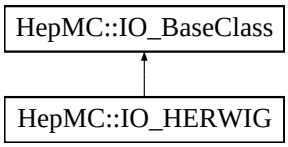
- **IO_HEPEVT.h**
- **IO_HEPEVT.cc**

9.35 HepMC::IO_HERWIG Class Reference

IO_HERWIG (p. 195) is used to get Herwig information.

`#include <IO_HERWIG.h>`

Inheritance diagram for HepMC::IO_HERWIG::



Public Member Functions

- **IO_HERWIG ()**
- **virtual ~IO_HERWIG ()**
- **bool fill_next_event (GenEvent *)**
  get the next event
- **void print (std::ostream &ostr=std::cout) const**
  write to ostr
- **double interfaces_to_version_number () const**
  this information is dubious
- **bool print_inconsistency_errors () const**
  default is true
- **void set_print_inconsistency_errors (bool b=true)**
  decide whether or not to print inconsistency errors
- **bool no_gaps_in_barcodes () const**
  ask how to deal with extra non-physical pseudo particles
- **void set_no_gaps_in_barcodes (bool a)**

Protected Member Functions

- bool **trust_both_mothers_and_daughters () const**
  default is true
- **bool trust_mothers_before_daughters () const**
  default is false
- **void set_trust_mothers_before_daughters (bool b=true)**
  define mother daughter trust rules
- **void set_trust_both_mothers_and_daughters (bool b=false)**

define mother daughter trust rules

- **GenParticle * build_particle (int index)**
  make a particle
- **void build_production_vertex (int i, std::vector< GenParticle * > &hepevt_particle, GenEvent *evt)**
  make a production vertex
- **void build_end_vertex (int i, std::vector< GenParticle * > &hepevt_particle, GenEvent *evt)**
  make a decay vertex
- **int find_in_map (const std::map< GenParticle *, int > &m, GenParticle *p) const**
  find this particle in the map
- **void repair_hepevt () const**
  make the HERWIG HEPEVT common block look like the standard
- **void remove_gaps_in_hepevt () const**
  deal with artifacts of repairing HEPEVT
- **void zero_hepevt_entry (int i) const**
  zero out a HEPEVT pseudo particle
- **int translate_herwig_to_pdg_id (int i) const**
  translate particle ID

9.35.1 Detailed Description

IO_HERWIG (p. 195) is used to get Herwig information.

IO class for reading the HEPEVT common block from the Herwig monte carlo program.

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 56 of file IO_HERWIG.h.

9.35.2 Constructor & Destructor Documentation

9.35.2.1 HepMC::IO_HERWIG::IO_HERWIG ()

Definition at line 12 of file IO_HERWIG.cc.

9.35.2.2 HepMC::IO_HERWIG::~IO_HERWIG () [virtual]

Definition at line 83 of file IO_HERWIG.cc.

9.35.3 Member Function Documentation

9.35.3.1 void HepMC::IO_HERWIG::build_end_vertex (int *i*, std::vector< GenParticle * > & *hepevt_particle*, GenEvent * *evt*) [protected]

make a decay vertex

for particle in HEPEVT with index i, build an end vertex if appropriate, and add that vertex to the event

Definition at line 304 of file IO_HERWIG.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::first_child(), HepMC::HEPEVT_Wrapper::last_child(), HepMC::HEPEVT_Wrapper::number_children(), p, HepMC::GenVertex::position(), HepMC::GenVertex::set_position(), HepMC::HEPEVT_Wrapper::t(), HepMC::HEPEVT_Wrapper::x(), HepMC::HEPEVT_Wrapper::y(), and HepMC::HEPEVT_Wrapper::z().

Referenced by fill_next_event().

9.35.3.2 GenParticle * HepMC::IO_HERWIG::build_particle (int *index*) [protected]

make a particle

Builds a particle object corresponding to index in HEPEVT

Definition at line 372 of file IO_HERWIG.cc.

References HepMC::HEPEVT_Wrapper::e(), HepMC::HEPEVT_Wrapper::id(), HepMC::HEPEVT_Wrapper::m(), p, HepMC::HEPEVT_Wrapper::px(), HepMC::HEPEVT_Wrapper::py(), HepMC::HEPEVT_Wrapper::pz(), and HepMC::HEPEVT_Wrapper::status().

Referenced by fill_next_event().

9.35.3.3 void HepMC::IO_HERWIG::build_production_vertex (int *i*, std::vector< GenParticle * > & *hepevt_particle*, GenEvent * *evt*) [protected]

make a production vertex

for particle in HEPEVT with index i, build a production vertex if appropriate, and add that vertex to the event

Definition at line 231 of file IO_HERWIG.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::first_parent(), HepMC::HEPEVT_Wrapper::last_parent(), HepMC::HEPEVT_Wrapper::number_parents(), p, HepMC::GenVertex::position(), HepMC::GenVertex::print(), HepMC::GenVertex::set_position(), HepMC::HEPEVT_Wrapper::t(), HepMC::HEPEVT_Wrapper::x(), HepMC::HEPEVT_Wrapper::y(), and HepMC::HEPEVT_Wrapper::z().

Referenced by fill_next_event().

9.35.3.4 bool HepMC::IO_HERWIG::fill_next_event (GenEvent *) [virtual]

get the next event

read one event from the Herwig HEPEVT common block and fill **GenEvent** (p. 75) return T/F =success/failure

sufficient to do one or the other.

Implements **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_f1dffb95a44d521af510f6431a30f942}$ **??**)

Definition at line 96 of file IO_HERWIG.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), build_end_vertex(), build_particle(), build_production_vertex(), HepMC::HEPEVT_Wrapper::event_number(), HepMC::HEPEVT_Wrapper::first_parent(), HepMC::HEPEVT_Wrapper::number_entries(), repair_hepevt(), HepMC::GenEvent::set_beam_particles(), HepMC::GenEvent::set_event_number(), HepMC::GenEvent::set_signal_process_vertex(), and HepMC::HEPEVT_Wrapper::status().

9.35.3.5 int HepMC::IO_HERWIG::find_in_map (const std::map< GenParticle *, int > & *m*, GenParticle * *p*) const [protected]

find this particle in the map

Definition at line 387 of file IO_HERWIG.cc.

References p.

9.35.3.6 double HepMC::IO_HERWIG::interfaces_to_version_number () const [inline]

this information is dubious

Definition at line 65 of file IO_HERWIG.h.

9.35.3.7 bool HepMC::IO_HERWIG::no_gaps_in_barcodes () const [inline]

ask how to deal with extra non-physical pseudo particles

Definition at line 74 of file IO_HERWIG.h.

9.35.3.8 void HepMC::IO_HERWIG::print (std::ostream & *ostr* = std::cout) const [virtual]

write to ostr

Reimplemented from **HepMC::IO_BaseClass p.** (classHepMC$_{11}IO_{BaseClass_8a23f5de9c6bb10931dcacdeb7677413}$ **??**)

Definition at line 85 of file IO_HERWIG.cc.

9.35.3.9 bool HepMC::IO_HERWIG::print_inconsistency_errors () const [inline]

default is true

Definition at line 145 of file IO_HERWIG.h.

9.35.3.10 void HepMC::IO_HERWIG::remove_gaps_in_hepevt () const [protected]

deal with artifacts of repairing HEPEVT

in this scenario, we do not allow there to be zero-ed entries in the HEPEVT common block, and so be reshuffle the common block, removing the zeero-ed entries as we go and making sure we keep the mother/daughter relationships appropriate

Definition at line 682 of file IO_HERWIG.cc.

References HepMC::HEPEVT_Wrapper::e(), HepMC::HEPEVT_Wrapper::first_child(), HepMC::HEPEVT_Wrapper::first_parent(), HepMC::HEPEVT_Wrapper::id(), HepMC::HEPEVT_Wrapper::last_child(), HepMC::HEPEVT_Wrapper::last_parent(), HepMC::HEPEVT_Wrapper::m(), HepMC::HEPEVT_Wrapper::number_entries(), HepMC::HEPEVT_Wrapper::px(), HepMC::HEPEVT_Wrapper::py(), HepMC::HEPEVT_Wrapper::pz(), HepMC::HEPEVT_Wrapper::set_children(), HepMC::HEPEVT_Wrapper::set_id(), HepMC::HEPEVT_Wrapper::set_mass(), HepMC::HEPEVT_Wrapper::set_momentum(), HepMC::HEPEVT_Wrapper::set_number_entries(), HepMC::HEPEVT_Wrapper::set_parents(), HepMC::HEPEVT_Wrapper::set_position(), HepMC::HEPEVT_Wrapper::set_status(), HepMC::HEPEVT_Wrapper::status(), HepMC::HEPEVT_Wrapper::t(), HepMC::HEPEVT_Wrapper::x(), HepMC::HEPEVT_Wrapper::y(), and HepMC::HEPEVT_Wrapper::z().

Referenced by repair_hepevt().

9.35.3.11 void HepMC::IO_HERWIG::repair_hepevt () const [protected]

make the HERWIG HEPEVT common block look like the standard

This routine takes the HEPEVT common block as used in HERWIG, and converts it into the HEPEVT common block in the standard format

This means it:

- removes the color structure, which herwig overloads into the mother/daughter fields
- zeros extra entries for hard subprocess, etc.

Special HERWIG status codes 101,102 colliding beam particles 103 beam-beam collision CMS vector 120 hard subprocess CMS vector 121,122 hard subprocess colliding partons 123-129 hard subprocess outgoing particles 141-149 (ID=94) mirror image of hard subrpocess particles 100 (ID=0 cone)

Special HERWIG particle id's 91 clusters 94 jets 0 others with no pdg code

Definition at line 394 of file IO_HERWIG.cc.

References HepMC::HEPEVT_Wrapper::first_child(), HepMC::HEPEVT_Wrapper::first_parent(), HepMC::HEPEVT_Wrapper::id(), HepMC::HEPEVT_Wrapper::last_child(), HepMC::HEPEVT_Wrapper::last_parent(), HepMC::HEPEVT_Wrapper::number_entries(), remove_gaps_in_hepevt(), HepMC::HEPEVT_Wrapper::set_children(), HepMC::HEPEVT_Wrapper::set_id(), HepMC::HEPEVT_Wrapper::set_parents(), HepMC::HEPEVT_Wrapper::status(), translate_herwig_to_pdg_id(), and zero_hepevt_entry().

Referenced by fill_next_event().

9.35.3.12 void HepMC::IO_HERWIG::set_no_gaps_in_barcodes (bool *a*) [inline]

The HERWIG HEPEVT common block has some EXTRA non-physical ENTRIES (such as CMS frame, HARD subprocess, and CONE). These are removed by **IO_HERWIG** (p. 195). Thus the **HepMC** (p. 25) event will APPEAR to have fewer particles in it that herwig did. There is a switch m_no_gaps_in_barcodes. For true - then the extra particles are removed from HEPEVT, with the result that the **HepMC** (p. 25) barcodes will be sequential, with no gaps. false - the barcodes will correspond directly to the HEPEVT index, but there will be gaps ... ie some barcodes will be unassigned. this switch requested by I Hinchliffe, October 31, 2002

Definition at line 87 of file IO_HERWIG.h.

9.35.3.13 void HepMC::IO_HERWIG::set_print_inconsistency_errors (bool *b* = true) [inline]

decide whether or not to print inconsistency errors

Definition at line 154 of file IO_HERWIG.h.

9.35.3.14 void HepMC::IO_HERWIG::set_trust_both_mothers_and_daughters (bool *b* = false) [inline, protected]

define mother daughter trust rules

Definition at line 148 of file IO_HERWIG.h.

9.35.3.15 void HepMC::IO_HERWIG::set_trust_mothers_before_daughters (bool *b* = true) [inline, protected]

define mother daughter trust rules

Definition at line 151 of file IO_HERWIG.h.

9.35.3.16 int HepMC::IO_HERWIG::translate_herwig_to_pdg_id (int *i*) const [protected]

translate particle ID

This routine is copied from Lynn Garren's stdhep 5.01. see http:///cepa.fnal.gov/psm/stdhep/

Definition at line 753 of file IO_HERWIG.cc.

Referenced by repair_hepevt().

9.35.3.17 bool HepMC::IO_HERWIG::trust_both_mothers_and_daughters () const [inline, protected]

default is true

Definition at line 139 of file IO_HERWIG.h.

9.35.3.18 bool HepMC::IO_HERWIG::trust_mothers_before_daughters () const [inline, protected]

default is false

Definition at line 142 of file IO_HERWIG.h.

9.35.3.19 void HepMC::IO_HERWIG::zero_hepevt_entry (int *i*) const [protected]

zero out a HEPEVT pseudo particle

Definition at line 742 of file IO_HERWIG.cc.

References HepMC::HEPEVT_Wrapper::max_number_entries(), HepMC::HEPEVT_Wrapper::set_children(), HepMC::HEPEVT_Wrapper::set_id(), HepMC::HEPEVT_Wrapper::set_mass(), HepMC::HEPEVT_Wrapper::set_momentum(), HepMC::HEPEVT_Wrapper::set_parents(), HepMC::HEPEVT_Wrapper::set_position(), and HepMC::HEPEVT_Wrapper::set_status().

Referenced by repair_hepevt().

The documentation for this class was generated from the following files:

- **IO_HERWIG.h**
- **IO_HERWIG.cc**

9.36 HepMC::detail::is_arithmetic< T > Struct Template Reference

undefined and therefore non-arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = false**

9.36.1 Detailed Description

template<class T> struct HepMC::detail::is_arithmetic< T >

undefined and therefore non-arithmetic

Definition at line 22 of file is_arithmetic.h.

9.36.2 Member Data Documentation

9.36.2.1 template<class T> bool const HepMC::detail::is_arithmetic< T >::value = false [static]

Definition at line 24 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.37 HepMC::detail::is_arithmetic< char > Struct Template Reference

character is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.37.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< char >

character is arithmetic

Definition at line 29 of file is_arithmetic.h.

9.37.2 Member Data Documentation

9.37.2.1 bool const HepMC::detail::is_arithmetic< char >::value = true [static]

Definition at line 30 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.38 HepMC::detail::is_arithmetic< double > Struct Template Reference

double is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.38.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< double >

double is arithmetic

Definition at line 79 of file is_arithmetic.h.

9.38.2 Member Data Documentation

9.38.2.1 bool const HepMC::detail::is_arithmetic< double >::value = true [static]

Definition at line 80 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.39 HepMC::detail::is_arithmetic< float > Struct Template Reference

float is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.39.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< float >

float is arithmetic

Definition at line 74 of file is_arithmetic.h.

9.39.2 Member Data Documentation

9.39.2.1 bool const HepMC::detail::is_arithmetic< float >::value = true [static]

Definition at line 75 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.40 HepMC::detail::is_arithmetic< int > Struct Template Reference

int is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.40.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< int >

int is arithmetic

Definition at line 54 of file is_arithmetic.h.

9.40.2 Member Data Documentation

9.40.2.1 bool const HepMC::detail::is_arithmetic< int >::value = true [static]

Definition at line 55 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.41 HepMC::detail::is_arithmetic< long > Struct Template Reference

long is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.41.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< long >

long is arithmetic

Definition at line 64 of file is_arithmetic.h.

9.41.2 Member Data Documentation

9.41.2.1 bool const HepMC::detail::is_arithmetic< long >::value = true [static]

Definition at line 65 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.42 HepMC::detail::is_arithmetic< long double > Struct Template Reference

long double is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.42.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< long double >

long double is arithmetic

Definition at line 84 of file is_arithmetic.h.

9.42.2 Member Data Documentation

9.42.2.1 bool const HepMC::detail::is_arithmetic< long double >::value = true [static]

Definition at line 85 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.43 HepMC::detail::is_arithmetic< short > Struct Template Reference

short is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.43.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< short >

short is arithmetic

Definition at line 44 of file is_arithmetic.h.

9.43.2 Member Data Documentation

9.43.2.1 bool const HepMC::detail::is_arithmetic< short >::value = true [static]

Definition at line 45 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.44 HepMC::detail::is_arithmetic< signed char > Struct Template Reference

signed character is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.44.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< signed char >

signed character is arithmetic

Definition at line 39 of file is_arithmetic.h.

9.44.2 Member Data Documentation

9.44.2.1 bool const HepMC::detail::is_arithmetic< signed char >::value = true [static]

Definition at line 40 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.45 HepMC::detail::is_arithmetic< unsigned char > Struct Template Reference

unsigned character is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.45.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< unsigned char >

unsigned character is arithmetic

Definition at line 34 of file is_arithmetic.h.

9.45.2 Member Data Documentation

9.45.2.1 bool const HepMC::detail::is_arithmetic< unsigned char >::value = true [static]

Definition at line 35 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.46 HepMC::detail::is_arithmetic< unsigned int > Struct Template Reference

unsigned int is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.46.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< unsigned int >

unsigned int is arithmetic

Definition at line 59 of file is_arithmetic.h.

9.46.2 Member Data Documentation

9.46.2.1 bool const HepMC::detail::is_arithmetic< unsigned int >::value = true [static]

Definition at line 60 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.47 HepMC::detail::is_arithmetic< unsigned long > Struct Template Reference

unsigned long is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.47.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< unsigned long >

unsigned long is arithmetic

Definition at line 69 of file is_arithmetic.h.

9.47.2 Member Data Documentation

9.47.2.1 bool const HepMC::detail::is_arithmetic< unsigned long >::value = true [static]

Definition at line 70 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.48 HepMC::detail::is_arithmetic< unsigned short > Struct Template Reference

unsigned short is arithmetic

#include <is_arithmetic.h>

Static Public Attributes

- static bool const **value = true**

9.48.1 Detailed Description

template<> struct HepMC::detail::is_arithmetic< unsigned short >

unsigned short is arithmetic

Definition at line 49 of file is_arithmetic.h.

9.48.2 Member Data Documentation

9.48.2.1 bool const HepMC::detail::is_arithmetic< unsigned short >::value = true [static]

Definition at line 50 of file is_arithmetic.h.

The documentation for this struct was generated from the following file:

- **is_arithmetic.h**

9.49 IsEventGood Class Reference

example class

Public Member Functions

- bool **operator() (const HepMC::GenEvent ∗evt)**
  check this event for goodness

9.49.1 Detailed Description

example class

event selection predicate. returns true if the event contains a photon with pT > 50 GeV

Examples:

example_EventSelection.cc.

Definition at line 20 of file example_EventSelection.cc.

9.49.2 Member Function Documentation

9.49.2.1 bool IsEventGood::operator() (const HepMC::GenEvent ∗ *evt*) [inline]

check this event for goodness

Examples:

example_EventSelection.cc.

Definition at line 23 of file example_EventSelection.cc.

References p, HepMC::GenEvent::particles_begin(), and HepMC::GenEvent::particles_end().

The documentation for this class was generated from the following file:

- **example_EventSelection.cc**

9.50 IsFinalState Class Reference

`#include <testHepMCIteration.h>`

Public Member Functions

- `bool` **operator() (const HepMC::GenParticle *p)**
  returns true if the GenParticle does not decay

9.50.1 Detailed Description

this predicate returns true if the input has no decay vertex

Examples:

testHepMCIteration.cc.in.

Definition at line 24 of file testHepMCIteration.h.

9.50.2 Member Function Documentation

9.50.2.1 bool IsFinalState::operator() (const HepMC::GenParticle * *p*) [inline]

returns true if the GenParticle does not decay

Definition at line 27 of file testHepMCIteration.h.

References p.

The documentation for this class was generated from the following file:

- **testHepMCIteration.h**

9.51 IsGoodEvent Class Reference

used in the tests

#include <IsGoodEvent.h>

Public Member Functions

- bool **operator() (const HepMC::GenEvent *evt)**

9.51.1 Detailed Description

used in the tests

event selection predicate. returns true if the event contains a photon with pT > 50 GeV

Examples:

testHepMC.cc.in, testHepMCIteration.cc.in, testMass.cc.in, testMultipleCopies.cc.in, and testStreamIO.cc.in.

Definition at line 14 of file IsGoodEvent.h.

9.51.2 Member Function Documentation

9.51.2.1 bool IsGoodEvent::operator() (const HepMC::GenEvent * *evt*) [inline]

Definition at line 16 of file IsGoodEvent.h.

References p, HepMC::GenEvent::particles_begin(), and HepMC::GenEvent::particles_end().

The documentation for this class was generated from the following file:

- **IsGoodEvent.h**

9.52 IsGoodEventMyPythia Class Reference

example class

Public Member Functions

- bool **operator() (const HepMC::GenEvent ∗evt)**
  returns true if event is "good"

9.52.1 Detailed Description

example class

event selection predicate. returns true if the event contains a photon with pT > 25 GeV

Examples:

fio/example_MyPythia.cc.

Definition at line 61 of file example_MyPythia.cc.

9.52.2 Member Function Documentation

9.52.2.1 bool IsGoodEventMyPythia::operator() (const HepMC::GenEvent ∗ *evt*) [inline]

returns true if event is "good"

Examples:

fio/example_MyPythia.cc.

Definition at line 64 of file example_MyPythia.cc.

References p, HepMC::GenEvent::particles_begin(), and HepMC::GenEvent::particles_end().

The documentation for this class was generated from the following file:

- **example_MyPythia.cc**

9.53 IsPhoton Class Reference

example class

Public Member Functions

- bool **operator() (const HepMC::GenParticle ∗p)**

  returns true if the GenParticle is a photon with more than 10 GeV transverse momentum

9.53.1 Detailed Description

example class

this predicate returns true if the input particle is a photon in the central region (eta < 2.5) with pT > 10 GeV

Examples:

example_UsingIterators.cc.

Definition at line 20 of file example_UsingIterators.cc.

9.53.2 Member Function Documentation

9.53.2.1 bool IsPhoton::operator() (const HepMC::GenParticle ∗ *p*) [inline]

returns true if the GenParticle is a photon with more than 10 GeV transverse momentum

Examples:

example_UsingIterators.cc.

Definition at line 23 of file example_UsingIterators.cc.

References p.

The documentation for this class was generated from the following file:

- **example_UsingIterators.cc**

9.54 IsStateFinal Class Reference

example class

Public Member Functions

- bool **operator() (const HepMC::GenParticle ∗p)**
  returns true if the GenParticle does not decay

9.54.1 Detailed Description

example class

this predicate returns true if the input has no decay vertex

Examples:

example_UsingIterators.cc.

Definition at line 47 of file example_UsingIterators.cc.

9.54.2 Member Function Documentation

9.54.2.1 bool IsStateFinal::operator() (const HepMC::GenParticle ∗ *p*) [inline]

returns true if the GenParticle does not decay

Examples:

example_UsingIterators.cc.

Definition at line 50 of file example_UsingIterators.cc.

References p.

The documentation for this class was generated from the following file:

- **example_UsingIterators.cc**

9.55 IsW_Boson Class Reference

example class

Public Member Functions

- bool **operator() (const HepMC::GenParticle ∗p)**
  returns true if the GenParticle is a W

9.55.1 Detailed Description

example class

this predicate returns true if the input particle is a W+/W−

Examples:

example_UsingIterators.cc.

Definition at line 34 of file example_UsingIterators.cc.

9.55.2 Member Function Documentation

9.55.2.1 bool IsW_Boson::operator() (const HepMC::GenParticle ∗ *p*) [inline]

returns true if the GenParticle is a W

Examples:

example_UsingIterators.cc.

Definition at line 37 of file example_UsingIterators.cc.

References p.

The documentation for this class was generated from the following file:

- **example_UsingIterators.cc**

9.56 HepMC::PdfInfo Class Reference

The **PdfInfo** (p. 222) class stores PDF information.

#include <PdfInfo.h>

Public Member Functions

- **PdfInfo ()**
  default constructor
- **PdfInfo (int i1, int i2, double x1, double x2, double q, double p1, double p2, int pdf_id1=0, int pdf_id2=0)**
  all values EXCEPT pdf_id1 and pdf_id2 must be provided
- **~PdfInfo ()**
- **PdfInfo (PdfInfo const &orig)**
  copy constructor
- **PdfInfo & operator= (PdfInfo const &rhs)**
  make a copy
- **void swap (PdfInfo &other)**
  swap two PdfInfo (p. 222) objects
- **bool operator== (const PdfInfo &) const**
  check for equality
- **bool operator!= (const PdfInfo &) const**
  check for inequality
- **int id1 () const**
  flavour code of first parton
- **int id2 () const**
  flavour code of second parton
- **int pdf_id1 () const**
  LHAPDF set id of first parton.
- **int pdf_id2 () const**
  LHAPDF set id of second parton.
- **double x1 () const**
  fraction of beam momentum carried by first parton ("beam side")
- **double x2 () const**
  fraction of beam momentum carried by second parton ("target side")
- **double scalePDF () const**

Q-scale used in evaluation of PDF's (in GeV).

- **double pdf1 () const**
  PDF (id1, x1, Q) - x∗f(x).
- **double pdf2 () const**
  PDF (id2, x2, Q) - x∗f(x).
- **bool is_valid () const**
  verify that the instance contains non-zero information
- **void set_id1 (const int &i)**
  set flavour code of first parton
- **void set_id2 (const int &i)**
  set flavour code of second parton
- **void set_pdf_id1 (const int &i)**
  set LHAPDF set id of first parton
- **void set_pdf_id2 (const int &i)**
  set LHAPDF set id of second parton
- **void set_x1 (const double &f)**
  set fraction of beam momentum carried by first parton ("beam side")
- **void set_x2 (const double &f)**
  set fraction of beam momentum carried by second parton ("target side")
- **void set_scalePDF (const double &f)**
  set Q-scale used in evaluation of PDF's (in GeV)
- **void set_pdf1 (const double &f)**
  set x∗f(x) of first parton
- **void set_pdf2 (const double &f)**
  set x∗f(x) of second parton

9.56.1 Detailed Description

The **PdfInfo** (p. 222) class stores PDF information.

HepMC::PdfInfo (p. 222) stores additional PDF information for a **GenEvent** (p. 75). Creation and use of this information is optional.

- `int id1; // flavour code of first parton`
- `int id2; // flavour code of second parton`
- `int pdf_id1; // LHAPDF set id of first parton (zero by default)`

- int pdf_id2; // LHAPDF set id of second parton (zero by default)
- double x1; // fraction of beam momentum carried by first parton ("beam side")
- double x2; // fraction of beam momentum carried by second parton ("target side")
- double scalePDF; // Q-scale used in evaluation of PDF's (in GeV)
- double pdf1; // PDF (id1, x1, Q)
- double pdf2; // PDF (id2, x2, Q)

Input parton flavour codes id1 & id2 are expected to obey the PDG code conventions, especially g = 21.

The contents of pdf1 and pdf2 are expected to be x*f(x).
The LHAPDF set ids are the entries in the first column of
http:///projects.hepforge.org/lhapdf/PDFsets.index

Examples:

testMass.cc.in.

Definition at line 37 of file PdfInfo.h.

9.56.2 Constructor & Destructor Documentation

9.56.2.1 HepMC::PdfInfo::PdfInfo () [inline]

default constructor

Definition at line 43 of file PdfInfo.h.

9.56.2.2 HepMC::PdfInfo::PdfInfo (int *i1*, int *i2*, double *x1*, double *x2*, double *q*, double *p1*, double *p2*, int *pdf_id1* = 0, int *pdf_id2* = 0) [inline]

all values EXCEPT pdf_id1 and pdf_id2 must be provided

Definition at line 136 of file PdfInfo.h.

9.56.2.3 HepMC::PdfInfo::~PdfInfo () [inline]

Definition at line 60 of file PdfInfo.h.

9.56.2.4 HepMC::PdfInfo::PdfInfo (PdfInfo const & *orig*) [inline]

copy constructor

Definition at line 150 of file PdfInfo.h.

9.56.3 Member Function Documentation

9.56.3.1 int HepMC::PdfInfo::id1 () const [inline]

flavour code of first parton

Definition at line 75 of file PdfInfo.h.

Referenced by HepMC::operator<<(), and operator==().

9.56.3.2 int HepMC::PdfInfo::id2 () const [inline]

flavour code of second parton

Definition at line 77 of file PdfInfo.h.

Referenced by HepMC::operator<<(), and operator==().

9.56.3.3 bool HepMC::PdfInfo::is_valid () const [inline]

verify that the instance contains non-zero information

Definition at line 202 of file PdfInfo.h.

9.56.3.4 bool HepMC::PdfInfo::operator!= (const PdfInfo &) const [inline]

check for inequality

any nonmatching member generates inequality

Definition at line 196 of file PdfInfo.h.

9.56.3.5 PdfInfo & HepMC::PdfInfo::operator= (PdfInfo const & *rhs*) [inline]

make a copy

Definition at line 162 of file PdfInfo.h.

References swap().

9.56.3.6 bool HepMC::PdfInfo::operator== (const PdfInfo &) const [inline]

check for equality

equality requires that each member match

Definition at line 182 of file PdfInfo.h.

References id1(), id2(), pdf1(), pdf2(), pdf_id1(), pdf_id2(), scalePDF(), x1(), and x2().

9.56.3.7 double HepMC::PdfInfo::pdf1 () const [inline]

PDF (id1, x1, Q) * x*f(x).

Definition at line 89 of file PdfInfo.h.

Referenced by HepMC::operator<<(), and operator==().

9.56.3.8 double HepMC::PdfInfo::pdf2 () const [inline]

PDF (id2, x2, Q) - x*f(x).
Definition at line 91 of file PdfInfo.h.
Referenced by HepMC::operator<<(), and operator==().

9.56.3.9 int HepMC::PdfInfo::pdf_id1 () const [inline]

LHAPDF set id of first parton.
Definition at line 79 of file PdfInfo.h.
Referenced by HepMC::operator<<(), and operator==().

9.56.3.10 int HepMC::PdfInfo::pdf_id2 () const [inline]

LHAPDF set id of second parton.
Definition at line 81 of file PdfInfo.h.
Referenced by HepMC::operator<<(), and operator==().

9.56.3.11 double HepMC::PdfInfo::scalePDF () const [inline]

Q-scale used in evaluation of PDF's (in GeV).
Definition at line 87 of file PdfInfo.h.
Referenced by HepMC::operator<<(), and operator==().

9.56.3.12 void HepMC::PdfInfo::set_id1 (const int & *i*) [inline]

set flavour code of first parton
Definition at line 98 of file PdfInfo.h.
Referenced by HepMC::operator>>().

9.56.3.13 void HepMC::PdfInfo::set_id2 (const int & *i*) [inline]

set flavour code of second parton
Definition at line 100 of file PdfInfo.h.
Referenced by HepMC::operator>>().

9.56.3.14 void HepMC::PdfInfo::set_pdf1 (const double & *f*) [inline]

set x*f(x) of first parton
Definition at line 112 of file PdfInfo.h.
Referenced by HepMC::operator>>().

9.56.3.15 void HepMC::PdfInfo::set_pdf2 (const double & *f*) [inline]

set x*f(x) of second parton

Definition at line 114 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.16 void HepMC::PdfInfo::set_pdf_id1 (const int & *i*) [inline]

set LHAPDF set id of first parton

Definition at line 102 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.17 void HepMC::PdfInfo::set_pdf_id2 (const int & *i*) [inline]

set LHAPDF set id of second parton

Definition at line 104 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.18 void HepMC::PdfInfo::set_scalePDF (const double & *f*) [inline]

set Q-scale used in evaluation of PDF's (in GeV)

Definition at line 110 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.19 void HepMC::PdfInfo::set_x1 (const double & *f*) [inline]

set fraction of beam momentum carried by first parton ("beam side")

Definition at line 106 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.20 void HepMC::PdfInfo::set_x2 (const double & *f*) [inline]

set fraction of beam momentum carried by second parton ("target side")

Definition at line 108 of file PdfInfo.h.

Referenced by HepMC::operator>>().

9.56.3.21 void HepMC::PdfInfo::swap (PdfInfo & *other*) [inline]

swap two **PdfInfo** (p. 222) objects

Definition at line 169 of file PdfInfo.h.

References m_id1, m_id2, m_pdf1, m_pdf2, m_pdf_id1, m_pdf_id2, m_scalePDF, m_x1, and m_x2.

Referenced by operator=().

9.56.3.22 double HepMC::PdfInfo::x1 () const [inline]

fraction of beam momentum carried by first parton ("beam side")

Definition at line 83 of file PdfInfo.h.

Referenced by HepMC::operator<<(), and operator==().

9.56.3.23 double HepMC::PdfInfo::x2 () const [inline]

fraction of beam momentum carried by second parton ("target side")

Definition at line 85 of file PdfInfo.h.

Referenced by HepMC::operator<<(), and operator==().

The documentation for this class was generated from the following file:

- **PdfInfo.h**

9.57 pin3 Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **xsfx [81][2]**
- **int isig [3][1000]**
- **double sigh [1000]**

9.57.1 Detailed Description

`Definition at line 115 of file PythiaWrapper6_4.h.`

9.57.2 Member Data Documentation

9.57.2.1 int pin3::isig[3][1000]

`Definition at line 117 of file PythiaWrapper6_4.h.`

9.57.2.2 double pin3::sigh[1000]

`Definition at line 118 of file PythiaWrapper6_4.h.`

9.57.2.3 double pin3::xsfx[81][2]

`Definition at line 116 of file PythiaWrapper6_4.h.`

`The documentation for this struct was generated from the following file:`

- **PythiaWrapper6_4.h**

9.58 pin5 Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `int` **ngenpd**
- **int ngen [3][501]**
- **double xsec [3][501]**

9.58.1 Detailed Description

`Definition at line 132 of file PythiaWrapper6_4.h.`

9.58.2 Member Data Documentation

9.58.2.1 int pin5::ngen[3][501]

`Definition at line 133 of file PythiaWrapper6_4.h.`

9.58.2.2 int pin5::ngenpd

`Definition at line 133 of file PythiaWrapper6_4.h.`

9.58.2.3 double pin5::xsec[3][501]

`Definition at line 134 of file PythiaWrapper6_4.h.`

`The documentation for this struct was generated from the following file:`

- **PythiaWrapper6_4.h**

9.59 pin7 Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **sigt [6][7][7]**

9.59.1 Detailed Description

`Definition at line 140 of file PythiaWrapper6_4.h.`

9.59.2 Member Data Documentation

9.59.2.1 double pin7::sigt[6][7][7]

`Definition at line 141 of file PythiaWrapper6_4.h.`

`The documentation for this struct was generated from the following file:`

- **PythiaWrapper6_4.h**

9.60 pin8 Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **xpvmd [13]**
- **double xpanl [13]**
- **double xpanh [13]**
- **double xpbeh [13]**
- **double xpdir [13]**

9.60.1 Detailed Description

Definition at line 147 of file PythiaWrapper6_4.h.

9.60.2 Member Data Documentation

9.60.2.1 double pin8::xpanh[13]

Definition at line 150 of file PythiaWrapper6_4.h.

9.60.2.2 double pin8::xpanl[13]

Definition at line 149 of file PythiaWrapper6_4.h.

9.60.2.3 double pin8::xpbeh[13]

Definition at line 151 of file PythiaWrapper6_4.h.

9.60.2.4 double pin8::xpdir[13]

Definition at line 152 of file PythiaWrapper6_4.h.

9.60.2.5 double pin8::xpvmd[13]

Definition at line 148 of file PythiaWrapper6_4.h.

The documentation for this struct was generated from the following file:

- **PythiaWrapper6_4.h**

9.61 pin9 Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **vxpvmd [13]**
- **double vxpanl [13]**
- **double vxpanh [13]**
- **double vxpdgm [13]**

9.61.1 Detailed Description

Definition at line 158 of file PythiaWrapper6_4.h.

9.61.2 Member Data Documentation

9.61.2.1 double pin9::vxpanh[13]

Definition at line 161 of file PythiaWrapper6_4.h.

9.61.2.2 double pin9::vxpanl[13]

Definition at line 160 of file PythiaWrapper6_4.h.

9.61.2.3 double pin9::vxpdgm[13]

Definition at line 162 of file PythiaWrapper6_4.h.

9.61.2.4 double pin9::vxpvmd[13]

Definition at line 159 of file PythiaWrapper6_4.h.

The documentation for this struct was generated from the following file:

- **PythiaWrapper6_4.h**

9.62 HepMC::Polarization Class Reference

The **Polarization** (p. 234) class stores theta and phi for a **GenParticle** (p. 113).

`#include <Polarization.h>`

Public Member Functions

- **Polarization ()**
  default constructor
- **Polarization (double theta, double phi=0)**
  constructor requiring at least one value
- **Polarization (const Polarization &inpolar)**
  construct from another polarization object
- **Polarization (const ThreeVector &vec3in)**
  construct using the polar and azimuthal angles from a ThreeVector (p. 256)
- **virtual ~Polarization ()**
- **void swap (Polarization &other)**
  swap
- **Polarization & operator= (const Polarization &inpolar)**
  make a copy
- **bool operator== (const Polarization &) const**
  equality requires that theta and phi are equal
- **bool operator!= (const Polarization &) const**
  inequality results if either theta or phi differ
- **void print (std::ostream &ostr=std::cout) const**
  print theta and phi
- **double theta () const**
  returns polar angle in radians
- **double phi () const**
  returns azimuthal angle in radians
- **ThreeVector normal3d () const**
  unit 3 vector for easy manipulation
- **bool is_defined () const**
  returns true if the Polarization (p. 234) has been defined
- **double set_theta (double theta)**

set polar angle in radians

- **double set_phi (double phi)**
  set azimuthal angle in radians
- **void set_theta_phi (double theta, double phi)**
  set both polar and azimuthal angles in radians
- **ThreeVector set_normal3d (const ThreeVector &vec3in)**
  sets polarization according to direction of 3 vec
- **void set_undefined ()**
  *declares the Polarization (*p. *234) as undefined and zeros the values*

Friends

- `std::ostream &` **operator<< (std::ostream &, const Polarization &)**
  print polarization information

9.62.1 Detailed Description

The **Polarization** (p. 234) class stores theta and phi for a **GenParticle** (p. 113).

HepMC::Polarization (p. 234) stores a particle's theta and phi in radians. Use of this information is optional. By default, the polarization is set to zero.

Definition at line 29 of file Polarization.h.

9.62.2 Constructor & Destructor Documentation

9.62.2.1 HepMC::Polarization::Polarization ()

default constructor

Definition at line 11 of file Polarization.cc.

9.62.2.2 HepMC::Polarization::Polarization (double *theta*, double *phi* = 0)

constructor requiring at least one value

Definition at line 17 of file Polarization.cc.

9.62.2.3 HepMC::Polarization::Polarization (const Polarization & *inpolar*)

construct from another polarization object

Definition at line 23 of file Polarization.cc.

9.62.2.4 HepMC::Polarization::Polarization (const ThreeVector & *vec3in*)

construct using the polar and azimuthal angles from a **ThreeVector** (p. 256)

Definition at line 29 of file Polarization.cc.

9.62.2.5 virtual HepMC::Polarization::~Polarization () [inline, virtual]

Definition at line 43 of file Polarization.h.

9.62.3 Member Function Documentation

9.62.3.1 bool HepMC::Polarization::is_defined () const

returns true if the **Polarization** (p. 234) has been defined

Definition at line 77 of file Polarization.cc.

Referenced by operator==().

9.62.3.2 ThreeVector HepMC::Polarization::normal3d () const

unit 3 vector for easy manipulation

Definition at line 57 of file Polarization.cc.

References phi(), HepMC::ThreeVector::setPhi(), HepMC::ThreeVector::setTheta(), and theta().

9.62.3.3 bool HepMC::Polarization::operator!= (const Polarization &) const [inline]

inequality results if either theta or phi differ

Definition at line 104 of file Polarization.h.

9.62.3.4 Polarization & HepMC::Polarization::operator= (const Polarization & *inpolar*)

make a copy

best practices implementation

Definition at line 42 of file Polarization.cc.

References swap().

9.62.3.5 bool HepMC::Polarization::operator== (const Polarization &) const [inline]

equality requires that theta and phi are equal

Definition at line 99 of file Polarization.h.

References is_defined(), phi(), and theta().

9.62.3.6 double HepMC::Polarization::phi () const [inline]

returns azimuthal angle in radians

Definition at line 93 of file Polarization.h.

Referenced by normal3d(), HepMC::operator<<(), and operator==().

9.62.3.7 void HepMC::Polarization::print (std::ostream & *ostr* = std::cout) const

print theta and phi

Definition at line 49 of file Polarization.cc.

9.62.3.8 ThreeVector HepMC::Polarization::set_normal3d (const ThreeVector & *vec3in*)

sets polarization according to direction of 3 vec

Definition at line 93 of file Polarization.cc.

References HepMC::ThreeVector::phi(), set_phi(), set_theta(), and HepMC::ThreeVector::theta().

9.62.3.9 double HepMC::Polarization::set_phi (double *phi*)

set azimuthal angle in radians

Phi is restricted to be between 0 -> 2pi if an out of range value is given, it is translated to this range.

Definition at line 71 of file Polarization.cc.

Referenced by set_normal3d(), and set_theta_phi().

9.62.3.10 double HepMC::Polarization::set_theta (double *theta*)

set polar angle in radians

Theta is restricted to be between 0 -> pi if an out of range value is given, it is translated to this range.

Definition at line 65 of file Polarization.cc.

Referenced by set_normal3d(), and set_theta_phi().

9.62.3.11 void HepMC::Polarization::set_theta_phi (double *theta*, double *phi*)

set both polar and azimuthal angles in radians

Definition at line 87 of file Polarization.cc.

References set_phi(), and set_theta().

9.62.3.12 void HepMC::Polarization::set_undefined ()

declares the **Polarization** (p. 234) as undefined and zeros the values

Definition at line 81 of file Polarization.cc.

9.62.3.13 void HepMC::Polarization::swap (Polarization & *other*)

swap

Definition at line 35 of file Polarization.cc.

References m_defined, m_phi, and m_theta.

Referenced by operator=(), and HepMC::GenParticle::swap().

9.62.3.14 double HepMC::Polarization::theta () const [inline]

returns polar angle in radians

Definition at line 92 of file Polarization.h.

Referenced by normal3d(), HepMC::operator<<(), and operator==().

9.62.4 Friends And Related Function Documentation

9.62.4.1 std::ostream& operator<< (std::ostream & *ostr*, const Polarization & *polar*) [friend]

print polarization information

Definition at line 129 of file Polarization.cc.

The documentation for this class was generated from the following files:

- **Polarization.h**
- **Polarization.cc**

9.63 PrintChildren Class Reference

test class

#include <testHepMCIteration.h>

Public Member Functions

- **PrintChildren (std::ostream &os)**
- **void operator() (HepMC::GenParticle ∗p)**

9.63.1 Detailed Description

test class

prints the particle

Examples:

testHepMCIteration.cc.in.

Definition at line 62 of file testHepMCIteration.h.

9.63.2 Constructor & Destructor Documentation

9.63.2.1 PrintChildren::PrintChildren (std::ostream & *os*) [inline]

Definition at line 64 of file testHepMCIteration.h.

9.63.3 Member Function Documentation

9.63.3.1 void PrintChildren::operator() (HepMC::GenParticle ∗ *p*) [inline]

Definition at line 65 of file testHepMCIteration.h.

References p, HepMC::GenParticle::pdg_id(), and HepMC::GenParticle::status().

The documentation for this class was generated from the following file:

- **testHepMCIteration.h**

9.64 PrintDescendants Class Reference

test class

#include <testHepMCIteration.h>

Public Member Functions

- **PrintDescendants (std::ostream &os)**
- **void operator() (const HepMC::GenParticle ∗p)**

9.64.1 Detailed Description

test class

prints the particle

Examples:

testHepMCIteration.cc.in.

Definition at line 82 of file testHepMCIteration.h.

9.64.2 Constructor & Destructor Documentation

9.64.2.1 PrintDescendants::PrintDescendants (std::ostream & *os*) [inline]

Definition at line 84 of file testHepMCIteration.h.

9.64.3 Member Function Documentation

9.64.3.1 void PrintDescendants::operator() (const HepMC::GenParticle ∗ *p*) [inline]

Definition at line 85 of file testHepMCIteration.h.

References p.

The documentation for this class was generated from the following file:

- **testHepMCIteration.h**

9.65 PrintParticle Class Reference

`#include <testHepMCIteration.h>`

Public Member Functions

- **PrintParticle (std::ostream &os)**
- **void operator() (const HepMC::GenParticle ∗p)**

9.65.1 Detailed Description

prints the particle

Examples:

testHepMCIteration.cc.in.

Definition at line 47 of file testHepMCIteration.h.

9.65.2 Constructor & Destructor Documentation

9.65.2.1 PrintParticle::PrintParticle (std::ostream & *os*) [inline]

Definition at line 49 of file testHepMCIteration.h.

9.65.3 Member Function Documentation

9.65.3.1 void PrintParticle::operator() (const HepMC::GenParticle ∗ *p*) [inline]

Definition at line 50 of file testHepMCIteration.h.

References p.

The documentation for this class was generated from the following file:

- **testHepMCIteration.h**

9.66 PrintPhoton Class Reference

`#include <testHepMCIteration.h>`

Public Member Functions

- **PrintPhoton (std::ostream &os)**
- **void operator() (const HepMC::GenParticle ∗p)**

9.66.1 Detailed Description

prints the particle if it is a photon

Examples:

testHepMCIteration.cc.in.

Definition at line 35 of file testHepMCIteration.h.

9.66.2 Constructor & Destructor Documentation

9.66.2.1 PrintPhoton::PrintPhoton (std::ostream & *os*) [inline]

Definition at line 37 of file testHepMCIteration.h.

9.66.3 Member Function Documentation

9.66.3.1 void PrintPhoton::operator() (const HepMC::GenParticle ∗ *p*) [inline]

Definition at line 38 of file testHepMCIteration.h.

References IsPhoton(), and p.

The documentation for this class was generated from the following file:

- **testHepMCIteration.h**

9.67 prvnv Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **ab [2][16][2]**
- **double rms [4]**
- **double res [5][6]**
- **int idr**
- **int idr2**
- **double dcmass**
- **int kfr [3]**

9.67.1 Detailed Description

Definition at line 200 of file PythiaWrapper6_4.h.

9.67.2 Member Data Documentation

9.67.2.1 double prvnv::ab[2][16][2]

Definition at line 201 of file PythiaWrapper6_4.h.

9.67.2.2 double prvnv::dcmass

Definition at line 206 of file PythiaWrapper6_4.h.

9.67.2.3 int prvnv::idr

Definition at line 204 of file PythiaWrapper6_4.h.

9.67.2.4 int prvnv::idr2

Definition at line 205 of file PythiaWrapper6_4.h.

9.67.2.5 int prvnv::kfr[3]

Definition at line 207 of file PythiaWrapper6_4.h.

9.67.2.6 double prvnv::res[5][6]

Definition at line 203 of file PythiaWrapper6_4.h.

9.67.2.7 double prvnv::rms[4]

Definition at line 202 of file PythiaWrapper6_4.h.

The documentation for this struct was generated from the following file:

- **PythiaWrapper6_4.h**

9.68 prvpm Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `double` **rm [4]**
- **double a [2]**
- **double b [2]**
- **double resm [2]**
- **double resw [2]**
- **bool mflag**

9.68.1 Detailed Description

Definition at line 213 of file PythiaWrapper6_4.h.

9.68.2 Member Data Documentation

9.68.2.1 double prvpm::a[2]

Definition at line 215 of file PythiaWrapper6_4.h.

9.68.2.2 double prvpm::b[2]

Definition at line 216 of file PythiaWrapper6_4.h.

9.68.2.3 bool prvpm::mflag

Definition at line 219 of file PythiaWrapper6_4.h.

9.68.2.4 double prvpm::resm[2]

Definition at line 217 of file PythiaWrapper6_4.h.

9.68.2.5 double prvpm::resw[2]

Definition at line 218 of file PythiaWrapper6_4.h.

9.68.2.6 double prvpm::rm[4]

Definition at line 214 of file PythiaWrapper6_4.h.

The documentation for this struct was generated from the following file:

- **PythiaWrapper6_4.h**

9.69 pssm Struct Reference

`#include <PythiaWrapper6_4.h>`

Public Attributes

- `int` **imss [100]**
- **double rmss [100]**

9.69.1 Detailed Description

`Definition at line 168 of file PythiaWrapper6_4.h.`

9.69.2 Member Data Documentation

9.69.2.1 int pssm::imss[100]

`Definition at line 169 of file PythiaWrapper6_4.h.`

9.69.2.2 double pssm::rmss[100]

`Definition at line 170 of file PythiaWrapper6_4.h.`

`The documentation for this struct was generated from the following file:`

- **PythiaWrapper6_4.h**

9.70 HepMC::StreamInfo Class Reference

StreamInfo (p. 247) contains extra information needed when using streaming IO.

`#include <StreamInfo.h>`

Public Member Functions

- **StreamInfo ()**
  default constructor
- **~StreamInfo ()**
  destructor
- **std::string IO_GenEvent_Key () const**
  IO_GenEvent (p. 186) begin event block key.
- **std::string IO_GenEvent_End () const**
  IO_GenEvent (p. 186) end event block key.
- **std::string IO_Ascii_Key () const**
- **std::string IO_Ascii_End () const**
  IO_Ascii end event block key.
- **std::string IO_Ascii_PDT_Key () const**
  IO_Ascii begin particle data block key.
- **std::string IO_Ascii_PDT_End () const**
  IO_Ascii end particle data block key.
- **std::string IO_ExtendedAscii_Key () const**
- **std::string IO_ExtendedAscii_End () const**
  IO_ExtendedAscii end event block key.
- **std::string IO_ExtendedAscii_PDT_Key () const**
  IO_ExtendedAscii begin particle data block key.
- **std::string IO_ExtendedAscii_PDT_End () const**
  IO_ExtendedAscii end particle data block key.
- **int io_type () const**
  get IO type
- **void set_io_type (int)**
  set IO type
- **bool has_key () const**
- **void set_has_key (bool)**
  set to false if the stream does not have a file type key

- **Units::MomentumUnit io_momentum_unit () const**

  get the I/O momentum units

- **Units::LengthUnit io_position_unit () const**

  get the I/O length units

- **int stream_id () const**
- **bool finished_first_event () const**

  Special information is processed the first time we use the IO.

- **void set_finished_first_event (bool b)**

  Special information is processed the first time we use the IO.

- **void use_input_units (Units::MomentumUnit, Units::LengthUnit)**
- **bool reading_event_header ()**
- **void set_reading_event_header (bool)**

  set the reading_event_header flag

9.70.1 Detailed Description

StreamInfo (p. 247) contains extra information needed when using streaming IO.

This class contains the extra information needed when using streaming IO to process **HepMC** (p. 25) GenEvents

Definition at line 26 of file StreamInfo.h.

9.70.2 Constructor & Destructor Documentation

9.70.2.1 HepMC::StreamInfo::StreamInfo ()

default constructor

Definition at line 13 of file StreamInfo.cc.

9.70.2.2 HepMC::StreamInfo::~StreamInfo () [inline]

destructor

Definition at line 31 of file StreamInfo.h.

9.70.3 Member Function Documentation

9.70.3.1 bool HepMC::StreamInfo::finished_first_event () const [inline]

Special information is processed the first time we use the IO.

Definition at line 81 of file StreamInfo.h.

Referenced by HepMC::detail::establish_input_stream_info(), HepMC::establish_input_stream_info(), HepMC::detail::establish_output_stream_info(), HepMC::establish_output_stream_info(), HepMC::GenEvent::read(), HepMC::GenEvent::write(), HepMC::write_HepMC_IO_block_begin(), and HepMC::write_HepMC_IO_block_end().

9.70.3.2 bool HepMC::StreamInfo::has_key () const [inline]

true if the stream has a file type key has_key is true by default

Definition at line 67 of file StreamInfo.h.

Referenced by HepMC::GenEvent::read().

9.70.3.3 std::string HepMC::StreamInfo::IO_Ascii_End () const [inline]

IO_Ascii end event block key.

Definition at line 43 of file StreamInfo.h.

9.70.3.4 std::string HepMC::StreamInfo::IO_Ascii_Key () const [inline]

IO_Ascii begin event block key IO_Ascii has been removed, but we want to be able to read existing files written by IO_Ascii

Definition at line 41 of file StreamInfo.h.

9.70.3.5 std::string HepMC::StreamInfo::IO_Ascii_PDT_End () const [inline]

IO_Ascii end particle data block key.

Definition at line 47 of file StreamInfo.h.

9.70.3.6 std::string HepMC::StreamInfo::IO_Ascii_PDT_Key () const [inline]

IO_Ascii begin particle data block key.

Definition at line 45 of file StreamInfo.h.

9.70.3.7 std::string HepMC::StreamInfo::IO_ExtendedAscii_End () const [inline]

IO_ExtendedAscii end event block key.

Definition at line 54 of file StreamInfo.h.

9.70.3.8 std::string HepMC::StreamInfo::IO_ExtendedAscii_Key () const [inline]

IO_ExtendedAscii begin event block key IO_ExtendedAscii has been removed, but we want to be able to read existing files written by IO_ExtendedAscii

Definition at line 52 of file StreamInfo.h.

9.70.3.9 std::string HepMC::StreamInfo::IO_ExtendedAscii_PDT_End () const [inline]

IO_ExtendedAscii end particle data block key.

Definition at line 58 of file StreamInfo.h.

9.70.3.10 std::string HepMC::StreamInfo::IO_ExtendedAscii_PDT_Key () const [inline]

IO_ExtendedAscii begin particle data block key.

Definition at line 56 of file StreamInfo.h.

9.70.3.11 std::string HepMC::StreamInfo::IO_GenEvent_End () const [inline]

IO_GenEvent (p. 186) end event block key.

Definition at line 36 of file StreamInfo.h.

Referenced by HepMC::write_HepMC_IO_block_end().

9.70.3.12 std::string HepMC::StreamInfo::IO_GenEvent_Key () const [inline]

IO_GenEvent (p. 186) begin event block key.

Definition at line 34 of file StreamInfo.h.

Referenced by HepMC::write_HepMC_IO_block_begin().

9.70.3.13 Units::MomentumUnit HepMC::StreamInfo::io_momentum_unit () const [inline]

get the I/O momentum units

Definition at line 72 of file StreamInfo.h.

Referenced by HepMC::GenEvent::read().

9.70.3.14 Units::LengthUnit HepMC::StreamInfo::io_position_unit () const [inline]

get the I/O length units

Definition at line 74 of file StreamInfo.h.

Referenced by HepMC::GenEvent::read().

9.70.3.15 int HepMC::StreamInfo::io_type () const [inline]

get IO type

Definition at line 61 of file StreamInfo.h.

Referenced by HepMC::GenEvent::read(), and HepMC::detail::read_particle().

9.70.3.16 bool HepMC::StreamInfo::reading_event_header ()

reading_event_header will return true when streaming input is processing the **GenEvent** (p. 75) header information

Definition at line 51 of file StreamInfo.cc.

Referenced by HepMC::GenEvent::read().

9.70.3.17 void HepMC::StreamInfo::set_finished_first_event (bool *b*) [inline]

Special information is processed the first time we use the IO.

Definition at line 83 of file StreamInfo.h.

Referenced by HepMC::GenEvent::read(), and HepMC::GenEvent::write().

9.70.3.18 void HepMC::StreamInfo::set_has_key (bool)

set to false if the stream does not have a file type key

Definition at line 47 of file StreamInfo.cc.

9.70.3.19 void HepMC::StreamInfo::set_io_type (int)

set IO type

Definition at line 43 of file StreamInfo.cc.

9.70.3.20 void HepMC::StreamInfo::set_reading_event_header (bool)

set the reading_event_header flag

Definition at line 55 of file StreamInfo.cc.

Referenced by HepMC::GenEvent::read().

9.70.3.21 int HepMC::StreamInfo::stream_id () const [inline]

get the I/O stream id This is used for sanity checking.

Definition at line 78 of file StreamInfo.h.

Referenced by HepMC::HepMCStreamCallback().

9.70.3.22 void HepMC::StreamInfo::use_input_units (Units::MomentumUnit, Units::LengthUnit)

needed when reading a file without units if those units are different than the declared default units (e.g., the default units are MeV, but the file was written with GeV) This method is not necessary if the units are written in the file

Definition at line 38 of file StreamInfo.cc.

Referenced by HepMC::set_input_units().

The documentation for this class was generated from the following files:

- **StreamInfo.h**
- **StreamInfo.cc**

9.71 HepMC::TempParticleMap Class Reference

TempParticleMap (p. 253) is a temporary GenParticle* container used during input.

#include <TempParticleMap.h>

Public Types

- typedef std::map< **HepMC::GenParticle** *, **int** > **TempMap**
- **typedef std::map< int, HepMC::GenParticle * > TempOrderMap**
- **typedef TempMap::iterator TempMapIterator**
- **typedef TempOrderMap::iterator orderIterator**

Public Member Functions

- **TempParticleMap ()**
- **~TempParticleMap ()**
- **TempMapIterator begin ()**
- **TempMapIterator end ()**
- **orderIterator order_begin ()**
- **orderIterator order_end ()**
- **int end_vertex (GenParticle *)**
- **void addEndParticle (GenParticle *, int &)**

9.71.1 Detailed Description

TempParticleMap (p. 253) is a temporary GenParticle* container used during input.

Used by IO classes for recoverable particle ordering. Map GenParticle* against both outgoing vertex and particle order.

Definition at line 24 of file TempParticleMap.h.

9.71.2 Member Typedef Documentation

9.71.2.1 typedef TempOrderMap::iterator HepMC::TempParticleMap::orderIterator

Definition at line 29 of file TempParticleMap.h.

9.71.2.2 typedef std::map<HepMC::GenParticle*,int> HepMC::TempParticleMap::TempMap

Definition at line 26 of file TempParticleMap.h.

9.71.2.3 typedef TempMap::iterator HepMC::TempParticleMap::TempMapIterator

Definition at line 28 of file TempParticleMap.h.

9.71.2.4 typedef std::map<int,HepMC::GenParticle*> HepMC::TempParticleMap::TempOrderMap

Definition at line 27 of file TempParticleMap.h.

9.71.3 Constructor & Destructor Documentation

9.71.3.1 HepMC::TempParticleMap::TempParticleMap () [inline]

Definition at line 31 of file TempParticleMap.h.

9.71.3.2 HepMC::TempParticleMap::~TempParticleMap () [inline]

Definition at line 34 of file TempParticleMap.h.

9.71.4 Member Function Documentation

9.71.4.1 void HepMC::TempParticleMap::addEndParticle (GenParticle *, int &) [inline]

Definition at line 58 of file TempParticleMap.h.

References p.

Referenced by HepMC::detail::read_particle().

9.71.4.2 TempMapIterator HepMC::TempParticleMap::begin () [inline]

Definition at line 36 of file TempParticleMap.h.

9.71.4.3 TempMapIterator HepMC::TempParticleMap::end () [inline]

Definition at line 37 of file TempParticleMap.h.

Referenced by end_vertex().

9.71.4.4 int HepMC::TempParticleMap::end_vertex (GenParticle *) [inline]

Definition at line 50 of file TempParticleMap.h.

References end(), and p.

9.71.4.5 orderIterator HepMC::TempParticleMap::order_begin () [inline]

Definition at line 38 of file TempParticleMap.h.

9.71.4.6 orderIterator HepMC::TempParticleMap::order_end () [inline]

Definition at line 39 of file TempParticleMap.h.

The documentation for this class was generated from the following file:

- **TempParticleMap.h**

9.72 HepMC::ThreeVector Class Reference

ThreeVector (p. 256) is a simple representation of a position or displacement 3 vector.

#include <SimpleVector.h>

Public Member Functions

- **ThreeVector (double xin, double yin=0, double zin=0)**
  construct using x, y, and z (only x is required)
- **ThreeVector ()**
- **template<class T> ThreeVector (const T &v, typename detail::disable_if< detail::is_arithmetic< T >::value, void >::type *=0)**
- **ThreeVector (const ThreeVector &v)**
  copy constructor
- **void swap (ThreeVector &other)**
  swap
- **double x () const**
  return x
- **double y () const**
  return y
- **double z () const**
  return z
- **void setX (double xin)**
  set x
- **void setY (double yin)**
  set y
- **void setZ (double zin)**
  set z
- **void set (double x, double y, double z)**
  set x, y, and z
- **double phi () const**
  The azimuth angle.
- **double theta () const**
  The polar angle.
- **double r () const**
  The magnitude.

- **void setPhi (double)**
  Set phi keeping magnitude and theta constant (BaBar).
- **void setTheta (double)**
  Set theta keeping magnitude and phi constant (BaBar).
- **double perp2 () const**
  The transverse component squared (rho^2 in cylindrical coordinate system).
- **double perp () const**
  The transverse component (rho in cylindrical coordinate system).
- **ThreeVector & operator= (const ThreeVector &)**
  make a copy
- **bool operator== (const ThreeVector &) const**
  equality
- **bool operator!= (const ThreeVector &) const**
  inequality

9.72.1 Detailed Description

ThreeVector (p. 256) is a simple representation of a position or displacement 3 vector.

For compatibility with existing code, the basic expected geometrical access methods are povided. Also, there is a templated constructor that will take another vector (HepLorentzVector, GenVector, ...) which must have the following methods: **x()** (p. 261), **y()** (p. 261), **z()** (p. 261).

Examples:

testSimpleVector.cc, and VectorConversion.h.

Definition at line 131 of file SimpleVector.h.

9.72.2 Constructor & Destructor Documentation

9.72.2.1 HepMC::ThreeVector::ThreeVector (double *xin*, double *yin* = 0, double *zin* = 0) [inline]

construct using x, y, and z (only x is required)

Definition at line 136 of file SimpleVector.h.

9.72.2.2 HepMC::ThreeVector::ThreeVector () [inline]

Definition at line 139 of file SimpleVector.h.

9.72.2.3 template<class T> HepMC::ThreeVector::ThreeVector (const T & *v*, typename detail::disable_if< detail::is_arithmetic< T >::value, void >::type * = 0) [inline]

templated constructor this is used ONLY if T is not arithmetic

Definition at line 145 of file SimpleVector.h.

9.72.2.4 HepMC::ThreeVector::ThreeVector (const ThreeVector & *v*) [inline]

copy constructor

Definition at line 150 of file SimpleVector.h.

9.72.3 Member Function Documentation

9.72.3.1 bool HepMC::ThreeVector::operator!= (const ThreeVector &) const

inequality

9.72.3.2 ThreeVector& HepMC::ThreeVector::operator= (const ThreeVector &)

make a copy

9.72.3.3 bool HepMC::ThreeVector::operator== (const ThreeVector &) const

equality

9.72.3.4 double HepMC::ThreeVector::perp () const

The transverse component (rho in cylindrical coordinate system).

Examples:

testSimpleVector.cc.

Referenced by main().

9.72.3.5 double HepMC::ThreeVector::perp2 () const

The transverse component squared (rho^2 in cylindrical coordinate system).

Examples:

testSimpleVector.cc.

Referenced by main().

9.72.3.6 double HepMC::ThreeVector::phi () const

The azimuth angle.

Examples:

testSimpleVector.cc.

Referenced by main(), and HepMC::Polarization::set_normal3d().

9.72.3.7 double HepMC::ThreeVector::r () const

The magnitude.

Examples:

testSimpleVector.cc.

Referenced by main().

9.72.3.8 void HepMC::ThreeVector::set (double *x*, double *y*, double *z*)

set x, y, and z

Examples:

testSimpleVector.cc.

Referenced by main().

9.72.3.9 void HepMC::ThreeVector::setPhi (double)

Set phi keeping magnitude and theta constant (BaBar).

Examples:

testSimpleVector.cc.

Referenced by main(), and HepMC::Polarization::normal3d().

9.72.3.10 void HepMC::ThreeVector::setTheta (double)

Set theta keeping magnitude and phi constant (BaBar).

Examples:

testSimpleVector.cc.

Referenced by main(), and HepMC::Polarization::normal3d().

9.72.3.11 void HepMC::ThreeVector::setX (double *xin*) [inline]

set x

Examples:

testSimpleVector.cc.

Definition at line 159 of file SimpleVector.h.
Referenced by main().

9.72.3.12 void HepMC::ThreeVector::setY (double *yin*) [inline]

set y

Examples:

testSimpleVector.cc.

Definition at line 160 of file SimpleVector.h.
Referenced by main().

9.72.3.13 void HepMC::ThreeVector::setZ (double *zin*) [inline]

set z

Examples:

testSimpleVector.cc.

Definition at line 161 of file SimpleVector.h.
Referenced by main().

9.72.3.14 void HepMC::ThreeVector::swap (ThreeVector & *other*)

swap

9.72.3.15 double HepMC::ThreeVector::theta () const

The polar angle.

Examples:

testSimpleVector.cc.

Referenced by main(), and HepMC::Polarization::set_normal3d().

9.72.3.16 double HepMC::ThreeVector::x () const [inline]

return x

Examples:

testSimpleVector.cc.

Definition at line 155 of file SimpleVector.h.

Referenced by main().

9.72.3.17 double HepMC::ThreeVector::y () const [inline]

return y

Examples:

testSimpleVector.cc.

Definition at line 156 of file SimpleVector.h.

Referenced by main().

9.72.3.18 double HepMC::ThreeVector::z () const [inline]

return z

Examples:

testSimpleVector.cc.

Definition at line 157 of file SimpleVector.h.

Referenced by main().

The documentation for this class was generated from the following file:

- **SimpleVector.h**

9.73 HepMC::WeightContainer Class Reference

`Container for the Weights associated with an event or vertex.`

`#include <WeightContainer.h>`

Public Types

- `typedef std::size_t` **size_type**
  defining the size type used by vector and map
- **typedef std::vector< double >::iterator iterator**
  iterator for the weight container
- **typedef std::vector< double >::const_iterator const_iterator**
  const iterator for the weight container

Public Member Functions

- **WeightContainer (size_type n=0, double value=0.)**
  default constructor
- **WeightContainer (const std::vector< double > &weights)**
  construct from a vector of weights
- **WeightContainer (const WeightContainer &in)**
  copy
- **~WeightContainer ()**
- **void swap (WeightContainer &other)**
  swap
- **WeightContainer & operator= (const WeightContainer &)**
  copy assignment
- **WeightContainer & operator= (const std::vector< double > &in)**
  alternate assignment using a vector of doubles
- **void print (std::ostream &ostr=std::cout) const**
  print weights
- **void write (std::ostream &ostr=std::cout) const**
  write weights in a readable table
- **size_type size () const**
  size of weight container
- **bool empty () const**
  return true if weight container is empty

- **void push_back (const double &)**
  push onto weight container
- **void pop_back ()**
  pop from weight container
- **void clear ()**
  clear the weight container
- **bool has_key (const std::string &s) const**
  check to see if a name exists in the map
- **double & operator[ ] (size_type n)**
  access the weight container
- **const double & operator[ ] (size_type n) const**
  access the weight container
- **double & operator[ ] (const std::string &s)**
  access the weight container
- **const double & operator[ ] (const std::string &s) const**
  access the weight container
- **bool operator== (const WeightContainer &) const**
  equality
- **bool operator!= (const WeightContainer &) const**
  inequality
- **double & front ()**
  returns the first element
- **const double & front () const**
  returns the first element
- **double & back ()**
  returns the last element
- **const double & back () const**
  returns the last element
- **iterator begin ()**
  begining of the weight container
- **iterator end ()**
  end of the weight container
- **const_iterator begin () const**

begining of the weight container

- **const_iterator end () const**
  end of the weight container

Friends

- class **GenEvent**

9.73.1 Detailed Description

Container for the Weights associated with an event or vertex.

This class has both map-like and vector-like functionality. Named weights are now supported.

Definition at line 29 of file WeightContainer.h.

9.73.2 Member Typedef Documentation

9.73.2.1 typedef std::vector<double>::const_iterator HepMC::WeightContainer::const_iterator

const iterator for the weight container

Definition at line 38 of file WeightContainer.h.

9.73.2.2 typedef std::vector<double>::iterator HepMC::WeightContainer::iterator

iterator for the weight container

Definition at line 36 of file WeightContainer.h.

9.73.2.3 typedef std::size_t HepMC::WeightContainer::size_type

defining the size type used by vector and map

Definition at line 34 of file WeightContainer.h.

9.73.3 Constructor & Destructor Documentation

9.73.3.1 HepMC::WeightContainer::WeightContainer (size_type *n* = 0, double *value* = 0.) [explicit]

default constructor

Definition at line 22 of file WeightContainer.cc.

9.73.3.2 HepMC::WeightContainer::WeightContainer (const std::vector< double > & *weights*)

construct from a vector of weights

Definition at line 26 of file WeightContainer.cc.
References size().

9.73.3.3 HepMC::WeightContainer::WeightContainer (const WeightContainer & *in*) [inline]

copy
Definition at line 141 of file WeightContainer.h.

9.73.3.4 HepMC::WeightContainer::~WeightContainer () [inline]

Definition at line 145 of file WeightContainer.h.

9.73.4 Member Function Documentation

9.73.4.1 const double & HepMC::WeightContainer::back () const [inline]

returns the last element
Definition at line 192 of file WeightContainer.h.

9.73.4.2 double & HepMC::WeightContainer::back () [inline]

returns the last element
Definition at line 190 of file WeightContainer.h.

9.73.4.3 WeightContainer::const_iterator HepMC::WeightContainer::begin () const [inline]

begining of the weight container
Definition at line 201 of file WeightContainer.h.

9.73.4.4 WeightContainer::iterator HepMC::WeightContainer::begin () [inline]

begining of the weight container
Definition at line 195 of file WeightContainer.h.
Referenced by write(), and HepMC::IO_AsciiParticles::write_event().

9.73.4.5 void HepMC::WeightContainer::clear () [inline]

clear the weight container
Definition at line 173 of file WeightContainer.h.

9.73.4.6 bool HepMC::WeightContainer::empty () const [inline]

return true if weight container is empty
Definition at line 171 of file WeightContainer.h.

Referenced by main().

9.73.4.7 WeightContainer::const_iterator HepMC::WeightContainer::end () const [inline]

end of the weight container

Definition at line 204 of file WeightContainer.h.

9.73.4.8 WeightContainer::iterator HepMC::WeightContainer::end () [inline]

end of the weight container

Definition at line 198 of file WeightContainer.h.

Referenced by HepMC::GenVertex::print(), write(), and HepMC::IO_Ascii-Particles::write_event().

9.73.4.9 const double & HepMC::WeightContainer::front () const [inline]

returns the first element

Definition at line 187 of file WeightContainer.h.

9.73.4.10 double & HepMC::WeightContainer::front () [inline]

returns the first element

Definition at line 185 of file WeightContainer.h.

9.73.4.11 bool HepMC::WeightContainer::has_key (const std::string & *s*) const

check to see if a name exists in the map

Definition at line 105 of file WeightContainer.cc.

Referenced by main().

9.73.4.12 bool HepMC::WeightContainer::operator!= (const WeightContainer &) const

inequality

Definition at line 100 of file WeightContainer.cc.

9.73.4.13 WeightContainer & HepMC::WeightContainer::operator= (const std::vector< double > & *in*) [inline]

alternate assignment using a vector of doubles

best practices implementation

Definition at line 162 of file WeightContainer.h.

9.73.4.14 WeightContainer & HepMC::WeightContainer::operator= (const WeightContainer &) [inline]

copy assignment

best practices implementation

Definition at line 154 of file WeightContainer.h.

9.73.4.15 bool HepMC::WeightContainer::operator== (const WeightContainer &) const

equality

Definition at line 92 of file WeightContainer.cc.

References m_names, m_weights, and size().

9.73.4.16 const double & HepMC::WeightContainer::operator[ ] (const std::string & *s*) const

access the weight container

Definition at line 80 of file WeightContainer.cc.

9.73.4.17 double & HepMC::WeightContainer::operator[ ] (const std::string & *s*)

access the weight container

Definition at line 66 of file WeightContainer.cc.

9.73.4.18 const double & HepMC::WeightContainer::operator[ ] (size_type *n*) const [inline]

access the weight container

Definition at line 182 of file WeightContainer.h.

9.73.4.19 double & HepMC::WeightContainer::operator[ ] (size_type *n*) [inline]

access the weight container

Definition at line 179 of file WeightContainer.h.

9.73.4.20 void HepMC::WeightContainer::pop_back ()

pop from weight container

Definition at line 51 of file WeightContainer.cc.

References size().

Referenced by main().

9.73.4.21 void HepMC::WeightContainer::print (std::ostream & *ostr* = std::cout) const

print weights

Definition at line 111 of file WeightContainer.cc.

Referenced by HepMC::GenEvent::print().

9.73.4.22 void HepMC::WeightContainer::push_back (const double &)

push onto weight container

Definition at line 42 of file WeightContainer.cc.

Referenced by main().

9.73.4.23 WeightContainer::size_type HepMC::WeightContainer::size () const [inline]

size of weight container

Definition at line 169 of file WeightContainer.h.

Referenced by main(), operator==(), pop_back(), HepMC::GenVertex::print(), HepMC::GenEvent::print(), WeightContainer(), HepMC::GenEvent::write(), and HepMC::IO_AsciiParticles::write_event().

9.73.4.24 void HepMC::WeightContainer::swap (WeightContainer & *other*) [inline]

swap

Definition at line 147 of file WeightContainer.h.

References m_names, and m_weights.

Referenced by HepMC::GenVertex::swap(), and HepMC::GenEvent::swap().

9.73.4.25 void HepMC::WeightContainer::write (std::ostream & *ostr* = std::cout) const

write weights in a readable table

Definition at line 121 of file WeightContainer.cc.

References begin(), and end().

Referenced by main().

9.73.5 Friends And Related Function Documentation

9.73.5.1 friend class GenEvent [friend]

Definition at line 30 of file WeightContainer.h.

The documentation for this class was generated from the following files:

- **WeightContainer.h**
- **WeightContainer.cc**

Chapter 10

HepMC File Documentation

10.1 CompareGenEvent.cc File Reference

```
#include <iostream>
#include "HepMC/CompareGenEvent.h"
#include "HepMC/GenEvent.h"
```

Namespaces

- namespace **HepMC**

Functions

- bool **HepMC::compareGenEvent (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareSignalProcessVertex (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareBeamParticles (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareWeights (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareParticles (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareVertices (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareVertex (GenVertex ∗v1, GenVertex ∗v2)**

10.2 CompareGenEvent.h File Reference

```
#include <iostream>
#include "HepMC/GenEvent.h"
```

Namespaces

- namespace **HepMC**

Functions

- bool **HepMC::compareGenEvent (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareSignalProcessVertex (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareBeamParticles (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareWeights (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareVertices (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareParticles (GenEvent ∗, GenEvent ∗)**
- **bool HepMC::compareVertex (GenVertex ∗v1, GenVertex ∗v2)**

10.3 enable_if.h File Reference

Namespaces

- namespace **HepMC**
- **namespace HepMC::detail**

Classes

- struct **HepMC::detail::enable_if<, >**
  internal - used to decide if a class is arithmetic
- **struct HepMC::detail::enable_if< true, T >**
  internal - use if class T is arithmetic
- **struct HepMC::detail::disable_if<, >**
  internal - used by SimpleVector to decide if a class is arithmetic
- **struct HepMC::detail::disable_if< false, T >**
  internal - used by SimpleVector to decide if a class is arithmetic

10.4 example_BuildEventFromScratch.cc File Reference

```
#include <iostream>
#include "HepMC/GenEvent.h"
```

Functions

- int **main** ()

10.4.1 Function Documentation

10.4.1.1 int main ()

Examples:

example_BuildEventFromScratch.cc, example_EventSelection.cc, example_MyPythiaOnlyToHepMC.cc, example_UsingIterators.cc, example_VectorConversion.cc, fio/example_MyHerwig.cc, fio/example_MyPythia.cc, fio/example_PythiaStreamIO.cc, fio/testHerwigCopies.cc, fio/testPythiaCopies.cc, testFlow.cc, testHepMC.cc.in, testHepMCIteration.cc.in, testMass.cc.in, testMultipleCopies.cc.in, testPrintBug.cc, testSimpleVector.cc, testStreamIO.cc.in, and testUnits.cc.

Definition at line 22 of file example_BuildEventFromScratch.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::Units::GEV, HepMC::Units::MM, HepMC::GenEvent::print(), HepMC::GenEvent::set_signal_process_vertex(), and HepMC::GenEvent::use_units().

10.5 example_EventSelection.cc File Reference

```
#include "HepMC/IO_GenEvent.h"
#include "HepMC/GenEvent.h"
```

Classes

- class **IsEventGood**
  example class

Functions

- int **main** ()

10.5.1 Function Documentation

10.5.1.1 int main ()

Definition at line 37 of file example_EventSelection.cc.

References HepMC::GenEvent::event_number(), and HepMC::IO_Base-Class::read_next_event().

10.6 example_MyHerwig.cc File Reference

```
#include <iostream>
#include "HepMC/HerwigWrapper.h"
#include "HepMC/IO_HERWIG.h"
#include "HepMC/IO_GenEvent.h"
#include "HepMC/GenEvent.h"
#include "HepMC/HEPEVT_Wrapper.h"
```

Functions

- int **main** ()

10.6.1 Function Documentation

10.6.1.1 int main ()

To Compile: go to the **HepMC** (p. 25) directory and type: gmake examples/example_MyHerwig.exe

In this example the precision and number of entries for the HEPEVT fortran common block are explicitly defined to correspond to those used in the Herwig version of the HEPEVT common block. If you get funny output from HEPEVT in your own code, probably you have set these values incorrectly!

Definition at line 24 of file example_MyHerwig.cc.

References HepMC::getHerwigCrossSection(), HepMC::Units::GEV, hwbgen, hwbmch, hwcdec, hwcfor, hwdhad, hwdhob, hwdhvy, hwefin, hweini, hwepro, hwevnt, hwigin, hwmevt, hwproc, hwufne, hwuinc, hwuine, HepMC::Units::MM, HepMC::GenEvent::print(), HepMC::HEPEVT_Wrapper::print_hepevt(), HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

10.7 example_MyPythia.cc File Reference

```
#include <iostream>
#include "HepMC/PythiaWrapper.h"
#include "HepMC/IO_HEPEVT.h"
#include "HepMC/IO_GenEvent.h"
#include "HepMC/IO_AsciiParticles.h"
#include "HepMC/GenEvent.h"
#include "PythiaHelper.h"
```

Classes

- class **IsGoodEventMyPythia**
  example class

Functions

- void **pythia_out** ()
- **void pythia_in** ()
- **void pythia_in_out** ()
- **void event_selection** ()
- **void pythia_particle_out** ()
- **int main** ()

10.7.1 Function Documentation

10.7.1.1 void event_selection ()

Examples:

fio/example_MyPythia.cc.

Definition at line 152 of file example_MyPythia.cc.

References HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, pypars, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_mpi(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

Referenced by main().

10.7.1.2 int main ()

Definition at line 85 of file example_MyPythia.cc.

References event_selection(), pythia_in(), pythia_in_out(), and pythia_out().

10.7.1.3 void pythia_in ()

Examples:

fio/example_MyPythia.cc.

Definition at line 205 of file example_MyPythia.cc.

References HepMC::GenEvent::event_number(), and HepMC::IO_BaseClass::read_next_event().

Referenced by main().

10.7.1.4 void pythia_in_out ()

Examples:

fio/example_MyPythia.cc.

Definition at line 239 of file example_MyPythia.cc.

References HepMC::GenEvent::event_number(), HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

Referenced by main().

10.7.1.5 void pythia_out ()

Examples:

fio/example_MyPythia.cc.

Definition at line 99 of file example_MyPythia.cc.

References HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, pypars, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_mpi(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

Referenced by main().

10.7.1.6 void pythia_particle_out ()

Examples:

fio/example_MyPythia.cc.

Definition at line 311 of file example_MyPythia.cc.

References HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

10.8 example_MyPythiaOnlyToHepMC.cc File Reference

```
#include <iostream>
#include "HepMC/PythiaWrapper.h"
#include "HepMC/IO_HEPEVT.h"
#include "HepMC/GenEvent.h"
#include "PythiaHelper.h"
```

Functions

- int **main** ()

10.8.1 Function Documentation

10.8.1.1 int main ()

Definition at line 23 of file example_MyPythiaOnlyToHepMC.cc.

References HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, pypars, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_mpi(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

10.9 example_PythiaStreamIO.cc File Reference

```
#include <fstream>
#include <iostream>
#include "HepMC/PythiaWrapper.h"
#include "HepMC/IO_HEPEVT.h"
#include "HepMC/GenEvent.h"
#include "PythiaHelper.h"
```

Functions

- void **writePythiaStreamIO** ()
- **void readPythiaStreamIO** ()
- **int main** ()

10.9.1 Function Documentation

10.9.1.1 int main ()

Definition at line 31 of file example_PythiaStreamIO.cc.

References readPythiaStreamIO(), and writePythiaStreamIO().

10.9.1.2 void readPythiaStreamIO ()

Examples:

fio/example_PythiaStreamIO.cc.

Definition at line 103 of file example_PythiaStreamIO.cc.

References HepMC::GenEvent::cross_section(), HepMC::GenEvent::is_valid(), HepMC::GenEvent::read(), HepMC::GenEvent::write(), HepMC::write_HepMC_IO_block_begin(), and HepMC::write_HepMC_IO_block_end().

Referenced by main().

10.9.1.3 void writePythiaStreamIO ()

example of generating events with Pythia using **HepMC/PythiaWrapper.h** (p. 344) Events are read into the **HepMC** (p. 25) event record from the FORTRAN HEPEVT common block using the IO_HEPEVT strategy

To Compile: go to the **HepMC** (p. 25) example directory and type: make example_PythiaStreamIO.exe

This example uses streaming I/O **writePythiaStreamIO()** (p. 279) sets the cross section in GenRun **readPythiaStreamIO()** (p. 279) reads the file written by **writePythiaStreamIO()** (p. 279)

Examples:

fio/example_PythiaStreamIO.cc.

Definition at line 40 of file example_PythiaStreamIO.cc.

References HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, pypars, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_mpi(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), HepMC::GenEvent::use_units(), HepMC::write_HepMC_IO_block_begin(), and HepMC::write_HepMC_IO_block_end().

Referenced by main().

10.10 example_UsingIterators.cc File Reference

```
#include "HepMC/IO_GenEvent.h"
#include "HepMC/GenEvent.h"
#include <math.h>
#include <algorithm>
#include <list>
```

Classes

- class **IsPhoton**
  example class
- **class IsW_Boson**
  example class
- **class IsStateFinal**
  example class

Functions

- int **main ()**

10.10.1 Function Documentation

10.10.1.1 int main ()

Definition at line 56 of file example_UsingIterators.cc.

References HepMC::copy_if(), HepMC::descendants, p, HepMC::parents, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::IO_GenEvent::rdstate(), HepMC::IO_BaseClass::read_next_event(), v, HepMC::GenEvent::vertices_begin(), and HepMC::GenEvent::vertices_end().

10.11 example_VectorConversion.cc File Reference

```
#include <iostream>
#include "VectorConversion.h"
#include "HepMC/GenEvent.h"
#include "CLHEP/Vector/LorentzVector.h"
```

Functions

- int **main ()**

10.11.1 Function Documentation

10.11.1.1 int main ()

Definition at line 25 of file example_VectorConversion.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), convertTo(), HepMC::Units::GEV, HepMC::Units::MM, p, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::GenEvent::print(), HepMC::GenEvent::set_signal_process_vertex(), and HepMC::GenEvent::use_units().

10.12 filterEvent.cc File Reference

```
#include "HepMC/GenEvent.h"
```

Functions

- void **filterEvent (HepMC::GenEvent ∗ge)**

10.12.1 Function Documentation

10.12.1.1 void filterEvent (HepMC::GenEvent ∗ *ge*)

Definition at line 5 of file filterEvent.cc.

References HepMC::GenEvent::beam_particles(), HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::GenVertex::particles_out_const_begin(), HepMC::GenVertex::particles_out_const_end(), HepMC::GenVertex::particles_out_size(), HepMC::GenVertex::remove_particle(), HepMC::GenEvent::vertices_begin(), and HepMC::GenEvent::vertices_end().

10.13 Flow.cc File Reference

```
#include "HepMC/Flow.h"
#include "HepMC/GenParticle.h"
#include "HepMC/GenVertex.h"
#include "HepMC/SearchVector.h"
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &ostr, const Flow &f)**
  for printing

10.14 Flow.h File Reference

```
#include <iostream>
#include <map>
#include <vector>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::Flow**
  The flow object.

10.15 GenCrossSection.cc File Reference

```
#include <iostream>
#include <sstream>
#include "HepMC/GenCrossSection.h"
#include "HepMC/IO_Exception.h"
```

Namespaces

- namespace **HepMC**

10.16 GenCrossSection.h File Reference

```
#include <iostream>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::GenCrossSection**

  The GenCrossSection (p. ***71) class stores the generated cross section.***

Functions

- std::ostream & **HepMC::operator<< (std::ostream &os, GenCrossSection &xs)**
- **std::istream & HepMC::operator>> (std::istream &is, GenCrossSection &xs)**

10.17 GenEvent.cc File Reference

```
#include <iomanip>
#include "HepMC/GenEvent.h"
#include "HepMC/GenCrossSection.h"
#include "HepMC/Version.h"
#include "HepMC/StreamHelpers.h"
```

Namespaces

- namespace **HepMC**

10.18 GenEvent.h File Reference

```
#include "HepMC/GenVertex.h"
#include "HepMC/GenParticle.h"
#include "HepMC/WeightContainer.h"
#include "HepMC/GenCrossSection.h"
#include "HepMC/HeavyIon.h"
#include "HepMC/PdfInfo.h"
#include "HepMC/Units.h"
#include "HepMC/HepMCDefs.h"
#include <map>
#include <string>
#include <vector>
#include <algorithm>
#include <iostream>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::GenEvent**
  *The GenEvent (*p. *75) class is the core of HepMC (*p. *25).*
- **class HepMC::GenEvent::vertex_const_iterator**
  const vertex iterator
- **class HepMC::GenEvent::vertex_iterator**
  non-const vertex iterator
- **class HepMC::GenEvent::particle_const_iterator**
  const particle iterator
- **class HepMC::GenEvent::particle_iterator**
  non-const particle iterator

Functions

- template<class InputIterator, class OutputIterator, class Predicate> void **HepMC::copy_if (InputIterator first, InputIterator last, OutputIterator out, Predicate pred)**
  define the type of iterator to use

- **std::ostream & HepMC::operator<< (std::ostream &, GenEvent &)**
  standard streaming IO output operator

- **std::istream & HepMC::operator>> (std::istream &, GenEvent &)**
  standard streaming IO input operator

- **std::istream & HepMC::set_input_units (std::istream &, Units::MomentumUnit, Units::LengthUnit)**
  set the units for this input stream

- **std::ostream & HepMC::write_HepMC_IO_block_begin (std::ostream &)**
  *Explicitly write the begin block lines that IO_GenEvent (*p. *186) uses.*

- **std::ostream & HepMC::write_HepMC_IO_block_end (std::ostream &)**
  *Explicitly write the end block line that IO_GenEvent (*p. *186) uses.*

- **GenEvent & HepMC::convert_units (GenEvent &evt, Units::MomentumUnit m, Units::LengthUnit l)**

10.19 GenEventStreamIO.cc File Reference

```
#include <iostream>
#include <ostream>
#include <istream>
#include <sstream>
#include "HepMC/GenEvent.h"
#include "HepMC/GenCrossSection.h"
#include "HepMC/StreamInfo.h"
#include "HepMC/StreamHelpers.h"
#include "HepMC/Version.h"
#include "HepMC/IO_Exception.h"
```

Namespaces

- namespace **HepMC**
- **namespace HepMC::detail**

Functions

- void **HepMC::HepMCStreamCallback (std::ios_base::event e, std::ios_base &b, int i)**
- **template<class IO> StreamInfo & HepMC::get_stream_info (IO &iost)**
- **std::ostream & HepMC::operator<< (std::ostream &, GenEvent &)**

  standard streaming IO output operator

- **std::istream & HepMC::operator>> (std::istream &, GenEvent &)**

  standard streaming IO input operator

- **std::istream & HepMC::set_input_units (std::istream &, Units::MomentumUnit, Units::LengthUnit)**

  set the units for this input stream

- **std::ostream & HepMC::write_HepMC_IO_block_begin (std::ostream &)**

  Explicitly write the begin block lines that IO_GenEvent (p. 186) uses.

- **std::ostream & HepMC::write_HepMC_IO_block_end (std::ostream &)**

  Explicitly write the end block line that IO_GenEvent (p. 186) uses.

- **std::ostream & HepMC::establish_output_stream_info (std::ostream &os)**

  used by IO_GenEvent (p. 186) constructor

- **std::istream & HepMC::establish_input_stream_info (std::istream &is)**

  used by IO_GenEvent (p. 186) constructor

- **std::istream & HepMC::detail::read_particle (std::istream &, TempParticleMap &, GenParticle *)**

- **std::ostream & HepMC::detail::establish_output_stream_info (std::ostream &)**
  *used by IO_GenEvent (*p. *186) constructor*

- **std::istream & HepMC::detail::establish_input_stream_info (std::istream &)**
  *used by IO_GenEvent (*p. *186) constructor*

10.20 GenParticle.cc File Reference

```
#include "HepMC/GenEvent.h"
#include "HepMC/GenVertex.h"
#include "HepMC/GenParticle.h"
#include <iomanip>
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &ostr, const GenParticle &part)**
  print particle

10.21 GenParticle.h File Reference

```
#include "HepMC/Flow.h"
#include "HepMC/Polarization.h"
#include "HepMC/SimpleVector.h"
#include "HepMC/IteratorRange.h"
#include <iostream>
#include <stdint.h>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::GenParticle**
  *The GenParticle (*p. *113) class contains information about generated particles.*

Defines

- #define **hepmc_uint64_t uint64_t**

10.21.1 Define Documentation

10.21.1.1 #define hepmc_uint64_t uint64_t

Definition at line 38 of file GenParticle.h.

10.22 GenRanges.cc File Reference

```
#include <iostream>
#include "HepMC/GenRanges.h"
#include "HepMC/GenEvent.h"
#include "HepMC/GenVertex.h"
```

Namespaces

- namespace **HepMC**

10.23 GenRanges.h File Reference

```
#include <stdexcept>
#include "HepMC/GenEvent.h"
#include "HepMC/GenVertex.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::GenEventVertexRange**
  GenEventVertexRange (p. *112*) *acts like a collection of vertices.*
- **class HepMC::ConstGenEventVertexRange**
  ConstGenEventVertexRange (p. *47*) *acts like a collection of vertices.*
- **class HepMC::GenEventParticleRange**
  GenEventParticleRange (p. *111*) *acts like a collection of particles.*
- **class HepMC::ConstGenEventParticleRange**
  ConstGenEventParticleRange (p. *45*) *acts like a collection of particles.*
- **class HepMC::GenVertexParticleRange**
  GenVertexParticleRange (p. *153*) *acts like a collection of particles.*
- **class HepMC::GenParticleProductionRange**
  GenParticleProductionRange (p. *126*) *acts like a collection of particles.*
- **class HepMC::ConstGenParticleProductionRange**
- **class HepMC::GenParticleEndRange**
  GenParticleEndRange (p. *124*) *acts like a collection of particles.*
- **class HepMC::ConstGenParticleEndRange**

10.24 GenVertex.cc File Reference

```
#include "HepMC/GenParticle.h"
#include "HepMC/GenVertex.h"
#include "HepMC/GenEvent.h"
#include "HepMC/SearchVector.h"
#include <iomanip>
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &ostr, const GenVertex &vtx)**
  print vertex information

10.25 GenVertex.h File Reference

```
#include "HepMC/WeightContainer.h"
#include "HepMC/SimpleVector.h"
#include "HepMC/IteratorRange.h"
#include <iostream>
#include <iterator>
#include <vector>
#include <set>
#include <algorithm>
#include <cstddef>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::GenVertex**
  *GenVertex (*p. *128) contains information about decay vertices.*
- **class HepMC::GenVertex::edge_iterator**
  edge iterator
- **class HepMC::GenVertex::vertex_iterator**
  vertex iterator
- **class HepMC::GenVertex::particle_iterator**
  particle iterator

10.26 HeavyIon.cc File Reference

```
#include <iostream>
#include <ostream>
#include <istream>
#include <sstream>
#include "HepMC/HeavyIon.h"
#include "HepMC/StreamHelpers.h"
#include "HepMC/IO_Exception.h"
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &, HeavyIon const *)**
  Write the contents of HeavyIon (p. ***154***) ***to an output stream.***
- **std::istream & HepMC::operator>> (std::istream &, HeavyIon *)**
  Read the contents of HeavyIon (p. ***154***) ***from an input stream.***

10.27 HeavyIon.h File Reference

Namespaces

- `namespace` **HepMC**

Classes

- `class` **HepMC::HeavyIon**

  The HeavyIon (p. ***154) class stores information about heavy ions.***

Functions

- `std::ostream &` **HepMC::operator<< (std::ostream &, HeavyIon const *)**

  Write the contents of HeavyIon (p. ***154) to an output stream.***

- **std::istream & HepMC::operator>> (std::istream &, HeavyIon *)**

  Read the contents of HeavyIon (p. ***154) from an input stream.***

10.28 HEPEVT_Wrapper.cc File Reference

```
#include "HepMC/HEPEVT_Wrapper.h"
```

Namespaces

- namespace **HepMC**

10.29 HEPEVT_Wrapper.h File Reference

```
#include <ctype.h>
#include <iostream>
#include <cstdio>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::HEPEVT_Wrapper**
  Generic Wrapper for the fortran HEPEVT common block.

Defines

- #define **HEPEVT_EntriesAllocation 10000**
- **#define hepevt hepevt_**

Variables

- const unsigned int **hepevt_bytes_allocation**
- **struct {**
  char data [hepevt_bytes_allocation]
  } hepevt_

10.29.1 Define Documentation

10.29.1.1 #define hepevt hepevt_

Definition at line 84 of file HEPEVT_Wrapper.h.

Referenced by HepMC::HEPEVT_Wrapper::byte_num_to_double(), HepMC::HEPEVT_Wrapper::byte_num_to_int(), and HepMC::HEPEVT_Wrapper::write_byte_num().

10.29.1.2 #define HEPEVT_EntriesAllocation 10000

Definition at line 4 of file HEPEVT_Wrapper.h.

10.29.2 Variable Documentation

10.29.2.1 char data[hepevt_bytes_allocation]

Definition at line 81 of file HEPEVT_Wrapper.h.

10.29.2.2 struct { ... } hepevt_

10.29.2.3 const unsigned int hepevt_bytes_allocation

Initial value:

```
sizeof(long int) * ( 2 + 6 * HEPEVT_EntriesAllocation )
+ sizeof(double) * ( 9 * HEPEVT_EntriesAllocation )
```

Definition at line 66 of file HEPEVT_Wrapper.h.

Referenced by HepMC::HEPEVT_Wrapper::byte_num_to_double(), HepMC::HEPEVT_Wrapper::byte_num_to_int(), and HepMC::HEPEVT_Wrapper::write_byte_num().

10.30 HepMCDefs.h File Reference

Defines

- #define **HEPMC_VERSION "2.06.09"**

10.30.1 Define Documentation

10.30.1.1 #define HEPMC_VERSION "2.06.09"

Definition at line 65 of file HepMCDefs.h.

Referenced by HepMC::versionName().

10.31 HerwigWrapper.cc File Reference

```
#include <cmath>
#include "HepMC/HerwigWrapper.h"
#include "HepMC/GenCrossSection.h"
```

Namespaces

- namespace **HepMC**

Functions

- GenCrossSection **HepMC::getHerwigCrossSection (int ngen)**

Variables

- **hwgev hwevnt_**

10.31.1 Variable Documentation

10.31.1.1 struct hwgev hwevnt_

Definition at line 20 of file HerwigWrapper.cc.

10.32 HerwigWrapper.h File Reference

```
#include <ctype.h>
#include "HepMC/GenCrossSection.h"
```

Classes

- struct **hwgev**

Defines

- #define **hwproc hwproc_**
- **#define hwbeam hwbeam_**
- **#define hwbmch hwbmch_**
- **#define hwevnt hwevnt_**
- **#define hwpram hwpram_**
- **#define hwigin hwigin_**
- **#define hwigup hwigup_**
- **#define hwuinc hwuinc_**
- **#define hwusta hwusta_**
- **#define hweini hweini_**
- **#define hwuine hwuine_**
- **#define hwepro hwepro_**
- **#define hwupro hwupro_**
- **#define hwbgen hwbgen_**
- **#define hwdhob hwdhob_**
- **#define hwcfor hwcfor_**
- **#define hwcdec hwcdec_**
- **#define hwdhad hwdhad_**
- **#define hwdhvy hwdhvy_**
- **#define hwmevt hwmevt_**
- **#define hwufne hwufne_**
- **#define hwefin hwefin_**
- **#define hwudpr hwudpr_**
- **#define hwuepr hwuepr_**
- **#define hwupup hwupup_**
- **#define hwegup hwegup_**
- **#define hwudat hwudat_**

Variables

- struct {
    double **EBEAM1**
    double **EBEAM2**
    double **PBEAM1**
    double **PBEAM2**
    int **IPROC**
    int **MAXEV**
  } **hwproc_**

- struct {
    int **IPART1**
    int **IPART2**
  } **hwbeam_**

- struct {
    char **PART1 [8]**
    char **PART2 [8]**
  } **hwbmch_**

- const int **herwig_hepevt_size = 4000**
- **hwgev hwevnt_**
- **struct {**
  double AFCH [2][16]
  double ALPHEM
  double B1LIM
  double BETAF
  double BTCLM
  double CAFAC
  double CFFAC
  double CLMAX
  double CLPOW
  double CLSMR [2]
  double CSPEED
  double ENSOF
  double ETAMIX
  double F0MIX
  double F1MIX
  double F2MIX
  double GAMH
  double GAMW
  double GAMZ
  double GAMZP
  double GEV2NB
  double H1MIX
  double PDIQK
  double PGSMX
  double PGSPL [4]
  double PHIMIX
  double PIFAC
  double PRSOF
  double PSPLT [2]
  double PTRMS
  double PXRMS
  double QCDL3
  double QCDL5
  double QCDLAM
  double QDIQK
  double QFCH [16]
  double QG
  double QSPAC
  double QV
  double SCABI
  double SWEIN

double TMTOP
double VFCH [2][16]
double VCKM [3][3]
double VGCUT
double VQCUT
double VPCUT
double ZBINM
double EFFMIN
double OMHMIX
double ET2MIX
double PH3MIX
double GCUTME
int IOPREM
int IPRINT
int ISPAC
int LRSUD
int LWSUD
int MODPDF [2]
int NBTRY
int NCOLO
int NCTRY
int NDTRY
int NETRY
int NFLAV
int NGSPL
int NSTRU
int NSTRY
int NZBIN
int IOP4JT [2]
int NPRFMT
int AZSOFT
int AZSPIN
int CLDIR [2]
int HARDME
int NOSPAC
int PRNDEC
int PRVTX
int SOFTME
int ZPRIME
int PRNDEF
int PRNTEX
int PRNWEB
} hwpram_

10.32.1 Define Documentation

10.32.1.1 #define hwbeam hwbeam_

Definition at line 40 of file HerwigWrapper.h.

10.32.1.2 #define hwbgen hwbgen_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 104 of file HerwigWrapper.h.

Referenced by main().

10.32.1.3 #define hwbmch hwbmch_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 49 of file HerwigWrapper.h.

Referenced by main().

10.32.1.4 #define hwcdec hwcdec_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 107 of file HerwigWrapper.h.

Referenced by main().

10.32.1.5 #define hwcfor hwcfor_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 106 of file HerwigWrapper.h.

Referenced by main().

10.32.1.6 #define hwdhad hwdhad_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 108 of file HerwigWrapper.h.

Referenced by main().

10.32.1.7 #define hwdhob hwdhob_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 105 of file HerwigWrapper.h.

Referenced by main().

10.32.1.8 #define hwdhvy hwdhvy_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 109 of file HerwigWrapper.h.

Referenced by main().

10.32.1.9 #define hwefin hwefin_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 112 of file HerwigWrapper.h.

Referenced by main().

10.32.1.10 #define hwegup hwegup_

Definition at line 117 of file HerwigWrapper.h.

10.32.1.11 #define hweini hweini_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 100 of file HerwigWrapper.h.

Referenced by main().

10.32.1.12 #define hwepro hwepro_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 102 of file HerwigWrapper.h.

Referenced by main().

10.32.1.13 #define hwevnt hwevnt_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 63 of file HerwigWrapper.h.

Referenced by HepMC::getHerwigCrossSection(), and main().

10.32.1.14 #define hwigin hwigin_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 96 of file HerwigWrapper.h.

Referenced by main().

10.32.1.15 #define hwigup hwigup_

Definition at line 97 of file HerwigWrapper.h.

10.32.1.16 #define hwmevt hwmevt_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 110 of file HerwigWrapper.h.

Referenced by main().

10.32.1.17 #define hwpram hwpram_

Definition at line 91 of file HerwigWrapper.h.

10.32.1.18 #define hwproc hwproc_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 32 of file HerwigWrapper.h.

Referenced by main().

10.32.1.19 #define hwudat hwudat_

10.32.1.20 #define hwudpr hwudpr_

Definition at line 114 of file HerwigWrapper.h.

10.32.1.21 #define hwuepr hwuepr_

Definition at line 115 of file HerwigWrapper.h.

10.32.1.22 #define hwufne hwufne_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 111 of file HerwigWrapper.h.
Referenced by main().

10.32.1.23 #define hwuinc hwuinc_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 98 of file HerwigWrapper.h.
Referenced by main().

10.32.1.24 #define hwuine hwuine_

Examples:

fio/example_MyHerwig.cc, and fio/testHerwigCopies.cc.

Definition at line 101 of file HerwigWrapper.h.
Referenced by main().

10.32.1.25 #define hwupro hwupro_

Definition at line 103 of file HerwigWrapper.h.

10.32.1.26 #define hwupup hwupup_

Definition at line 116 of file HerwigWrapper.h.

10.32.1.27 #define hwusta hwusta_

Definition at line 99 of file HerwigWrapper.h.

10.32.2 Variable Documentation

10.32.2.1 double AFCH[2][16]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.2 double ALPHEM

Definition at line 79 of file HerwigWrapper.h.

10.32.2.3 int AZSOFT

Definition at line 87 of file HerwigWrapper.h.

10.32.2.4 int AZSPIN

Definition at line 87 of file HerwigWrapper.h.

10.32.2.5 double B1LIM

Definition at line 79 of file HerwigWrapper.h.

10.32.2.6 double BETAF

Definition at line 79 of file HerwigWrapper.h.

10.32.2.7 double BTCLM

Definition at line 79 of file HerwigWrapper.h.

10.32.2.8 double CAFAC

Definition at line 79 of file HerwigWrapper.h.

10.32.2.9 double CFFAC

Definition at line 79 of file HerwigWrapper.h.

10.32.2.10 int CLDIR[2]

Definition at line 87 of file HerwigWrapper.h.

10.32.2.11 double CLMAX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.12 double CLPOW

Definition at line 79 of file HerwigWrapper.h.

10.32.2.13 double CLSMR[2]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.14 double CSPEED

Definition at line 79 of file HerwigWrapper.h.

10.32.2.15 double EBEAM1

Definition at line 28 of file HerwigWrapper.h.

10.32.2.16 double EBEAM2

Definition at line 28 of file HerwigWrapper.h.

10.32.2.17 double EFFMIN

Definition at line 79 of file HerwigWrapper.h.

10.32.2.18 double ENSOF

Definition at line 79 of file HerwigWrapper.h.

10.32.2.19 double ET2MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.20 double ETAMIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.21 double F0MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.22 double F1MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.23 double F2MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.24 double GAMH

Definition at line 79 of file HerwigWrapper.h.

10.32.2.25 double GAMW

Definition at line 79 of file HerwigWrapper.h.

10.32.2.26 double GAMZ

Definition at line 79 of file HerwigWrapper.h.

10.32.2.27 double GAMZP

Definition at line 79 of file HerwigWrapper.h.

10.32.2.28 double GCUTME

Definition at line 79 of file HerwigWrapper.h.

10.32.2.29 double GEV2NB

Definition at line 79 of file HerwigWrapper.h.

10.32.2.30 double H1MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.31 int HARDME

Definition at line 87 of file HerwigWrapper.h.

10.32.2.32 const int herwig_hepevt_size = 4000

Definition at line 54 of file HerwigWrapper.h.

10.32.2.33 struct { ... } hwbeam_

10.32.2.34 struct { ... } hwbmch_

10.32.2.35 struct hwgev hwevnt_

10.32.2.36 struct { ... } hwpram_

10.32.2.37 struct { ... } hwproc_

10.32.2.38 int IOP4JT[2]

Definition at line 85 of file HerwigWrapper.h.

10.32.2.39 int IOPREM

Definition at line 85 of file HerwigWrapper.h.

10.32.2.40 int IPART1

Definition at line 37 of file HerwigWrapper.h.

10.32.2.41 int IPART2

Definition at line 37 of file HerwigWrapper.h.

10.32.2.42 int IPRINT

Definition at line 85 of file HerwigWrapper.h.

10.32.2.43 int IPROC

Definition at line 29 of file HerwigWrapper.h.

10.32.2.44 int ISPAC

Definition at line 85 of file HerwigWrapper.h.

10.32.2.45 int LRSUD

Definition at line 85 of file HerwigWrapper.h.

10.32.2.46 int LWSUD

Definition at line 85 of file HerwigWrapper.h.

10.32.2.47 int MAXEV

Definition at line 29 of file HerwigWrapper.h.

10.32.2.48 int MODPDF[2]

Definition at line 85 of file HerwigWrapper.h.

10.32.2.49 int NBTRY

Definition at line 85 of file HerwigWrapper.h.

10.32.2.50 int NCOLO

Definition at line 85 of file HerwigWrapper.h.

10.32.2.51 int NCTRY

Definition at line 85 of file HerwigWrapper.h.

10.32.2.52 int NDTRY

Definition at line 85 of file HerwigWrapper.h.

10.32.2.53 int NETRY

Definition at line 85 of file HerwigWrapper.h.

10.32.2.54 int NFLAV

Definition at line 85 of file HerwigWrapper.h.

10.32.2.55 int NGSPL

Definition at line 85 of file HerwigWrapper.h.

10.32.2.56 int NOSPAC

Definition at line 87 of file HerwigWrapper.h.

10.32.2.57 int NPRFMT

Definition at line 85 of file HerwigWrapper.h.

10.32.2.58 int NSTRU

Definition at line 85 of file HerwigWrapper.h.

10.32.2.59 int NSTRY

Definition at line 85 of file HerwigWrapper.h.

10.32.2.60 int NZBIN

Definition at line 85 of file HerwigWrapper.h.

10.32.2.61 double OMHMIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.62 char PART1[8]

Definition at line 46 of file HerwigWrapper.h.

10.32.2.63 char PART2[8]

Definition at line 46 of file HerwigWrapper.h.

10.32.2.64 double PBEAM1

Definition at line 28 of file HerwigWrapper.h.

10.32.2.65 double PBEAM2

Definition at line 28 of file HerwigWrapper.h.

10.32.2.66 double PDIQK

Definition at line 79 of file HerwigWrapper.h.

10.32.2.67 double PGSMX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.68 double PGSPL[4]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.69 double PH3MIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.70 double PHIMIX

Definition at line 79 of file HerwigWrapper.h.

10.32.2.71 double PIFAC

Definition at line 79 of file HerwigWrapper.h.

10.32.2.72 int PRNDEC

Definition at line 87 of file HerwigWrapper.h.

10.32.2.73 int PRNDEF

Definition at line 87 of file HerwigWrapper.h.

10.32.2.74 int PRNTEX

Definition at line 87 of file HerwigWrapper.h.

10.32.2.75 int PRNWEB

Definition at line 87 of file HerwigWrapper.h.

10.32.2.76 double PRSOF

Definition at line 79 of file HerwigWrapper.h.

10.32.2.77 int PRVTX

Definition at line 87 of file HerwigWrapper.h.

10.32.2.78 double PSPLT[2]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.79 double PTRMS

Definition at line 79 of file HerwigWrapper.h.

10.32.2.80 double PXRMS

Definition at line 79 of file HerwigWrapper.h.

10.32.2.81 double QCDL3

Definition at line 79 of file HerwigWrapper.h.

10.32.2.82 double QCDL5

Definition at line 79 of file HerwigWrapper.h.

10.32.2.83 double QCDLAM

Definition at line 79 of file HerwigWrapper.h.

10.32.2.84 double QDIQK

Definition at line 79 of file HerwigWrapper.h.

10.32.2.85 double QFCH[16]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.86 double QG

Definition at line 79 of file HerwigWrapper.h.

10.32.2.87 double QSPAC

Definition at line 79 of file HerwigWrapper.h.

10.32.2.88 double QV

Definition at line 79 of file HerwigWrapper.h.

10.32.2.89 double SCABI

Definition at line 79 of file HerwigWrapper.h.

10.32.2.90 int SOFTME

Definition at line 87 of file HerwigWrapper.h.

10.32.2.91 double SWEIN

Definition at line 79 of file HerwigWrapper.h.

10.32.2.92 double TMTOP

Definition at line 79 of file HerwigWrapper.h.

10.32.2.93 double VCKM[3][3]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.94 double VFCH[2][16]

Definition at line 79 of file HerwigWrapper.h.

10.32.2.95 double VGCUT

Definition at line 79 of file HerwigWrapper.h.

10.32.2.96 double VPCUT

Definition at line 79 of file HerwigWrapper.h.

10.32.2.97 double VQCUT

Definition at line 79 of file HerwigWrapper.h.

10.32.2.98 double ZBINM

Definition at line 79 of file HerwigWrapper.h.

10.32.2.99 int ZPRIME

Definition at line 87 of file HerwigWrapper.h.

10.33 initPythia.cc File Reference

```
#include "HepMC/PythiaWrapper.h"
#include "PythiaHelper.h"
```

Functions

- void **initPythia** ()

10.33.1 Function Documentation

10.33.1.1 void initPythia ()

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyPythia.cc, fio/example_PythiaStream-IO.cc, and fio/testPythiaCopies.cc.

Definition at line 12 of file initPythia.cc.

References pydat2, pydatr, pypars, and pysubs.

Referenced by event_selection(), main(), pythia_in_out(), pythia_-out(), pythia_particle_out(), and writePythiaStreamIO().

10.34 IO_AsciiParticles.cc File Reference

```
#include "HepMC/IO_AsciiParticles.h"
#include "HepMC/GenEvent.h"
#include "HepMC/Version.h"
```

Namespaces

- namespace **HepMC**

10.35 IO_AsciiParticles.h File Reference

```
#include <fstream>
#include <string>
#include <map>
#include <vector>
#include "HepMC/IO_BaseClass.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_AsciiParticles**
  event input/output in ascii format for eye and machine reading

10.36 IO_BaseClass.h File Reference

```
#include <iostream>
#include "HepMC/GenEvent.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_BaseClass**
  all input/output classes inherit from IO_BaseClass (p. *181*)

10.37 IO_Exception.h File Reference

```
#include <stdexcept>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_Exception**
  IO exception handling.

10.38 IO_GenEvent.cc File Reference

```
#include "HepMC/IO_GenEvent.h"
#include "HepMC/IO_Exception.h"
#include "HepMC/GenEvent.h"
#include "HepMC/StreamHelpers.h"
```

Namespaces

- namespace **HepMC**

10.39 IO_GenEvent.h File Reference

```
#include <fstream>
#include <string>
#include <map>
#include <vector>
#include "HepMC/IO_BaseClass.h"
#include "HepMC/IO_Exception.h"
#include "HepMC/Units.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_GenEvent**

  IO_GenEvent (p. ***186***) ***also deals with HeavyIon*** (p. ***154***) ***and PdfInfo*** (p. ***222***)***.***

10.40 IO_HEPEVT.cc File Reference

```
#include "HepMC/IO_HEPEVT.h"
#include "HepMC/GenEvent.h"
#include <cstdio>
```

Namespaces

- namespace **HepMC**

10.41 IO_HEPEVT.h File Reference

```
#include <map>
#include <vector>
#include "HepMC/IO_BaseClass.h"
#include "HepMC/HEPEVT_Wrapper.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_HEPEVT**
  HEPEVT IO class.

10.42 IO_HERWIG.cc File Reference

```
#include "HepMC/IO_HERWIG.h"
#include "HepMC/GenEvent.h"
#include <cstdio>
```

Namespaces

- namespace **HepMC**

10.43 IO_HERWIG.h File Reference

```
#include <set>
#include <vector>
#include "HepMC/IO_BaseClass.h"
#include "HepMC/HEPEVT_Wrapper.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::IO_HERWIG**
  IO_HERWIG (p. ***195***) ***is used to get Herwig information.***

10.44 is_arithmetic.h File Reference

Namespaces

- namespace **HepMC**
- **namespace detail**
- **namespace HepMC::detail**

Classes

- struct **HepMC::detail::is_arithmetic< T >**
  undefined and therefore non-arithmetic
- **struct HepMC::detail::is_arithmetic< char >**
  character is arithmetic
- **struct HepMC::detail::is_arithmetic< unsigned char >**
  unsigned character is arithmetic
- **struct HepMC::detail::is_arithmetic< signed char >**
  signed character is arithmetic
- **struct HepMC::detail::is_arithmetic< short >**
  short is arithmetic
- **struct HepMC::detail::is_arithmetic< unsigned short >**
  unsigned short is arithmetic
- **struct HepMC::detail::is_arithmetic< int >**
  int is arithmetic
- **struct HepMC::detail::is_arithmetic< unsigned int >**
  unsigned int is arithmetic
- **struct HepMC::detail::is_arithmetic< long >**
  long is arithmetic
- **struct HepMC::detail::is_arithmetic< unsigned long >**
  unsigned long is arithmetic
- **struct HepMC::detail::is_arithmetic< float >**
  float is arithmetic
- **struct HepMC::detail::is_arithmetic< double >**
  double is arithmetic
- **struct HepMC::detail::is_arithmetic< long double >**
  long double is arithmetic

10.45 IsGoodEvent.h File Reference

Classes

- `class` **IsGoodEvent**
  used in the tests

10.46 IteratorRange.h File Reference

Namespaces

- namespace **HepMC**

Enumerations

- enum **HepMC::IteratorRange {**
  HepMC::parents, HepMC::children, HepMC::family, HepMC::ancestors,
  HepMC::descendants, HepMC::relatives }
  type of iteration

10.47 list_of_examples.cc File Reference

10.48 list_of_examples.cc File Reference

10.49 main31.cc File Reference

```
#include "Pythia.h"
#include "HepMCInterface.h"
#include "HepMC/GenEvent.h"
#include "HepMC/IO_GenEvent.h"
```

Namespaces

- namespace **Pythia8**

Functions

- int **main** ()

10.49.1 Function Documentation

10.49.1.1 int main ()

Definition at line 32 of file main31.cc.

References HepMC::Units::GEV, and HepMC::Units::MM.

10.50 main32.cc File Reference

```
#include "Pythia.h"
#include "HepMCInterface.h"
#include "HepMC/GenEvent.h"
#include "HepMC/IO_GenEvent.h"
```

Functions

- int **main (int argc, char ∗argv[ ])**

10.50.1 Function Documentation

10.50.1.1 int main (int *argc*, char ∗ *argv*[ ])

Definition at line 33 of file main32.cc.

References HepMC::Units::GEV, and HepMC::Units::MM.

10.51 PdfInfo.cc File Reference

```
#include <iostream>
#include <ostream>
#include <istream>
#include <sstream>
#include "HepMC/PdfInfo.h"
#include "HepMC/StreamHelpers.h"
#include "HepMC/IO_Exception.h"
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &, PdfInfo const *)**
- **std::istream & HepMC::operator>> (std::istream &, PdfInfo *)**

10.52 PdfInfo.h File Reference

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::PdfInfo**

  The PdfInfo (p. 222) class stores PDF information.

Functions

- std::ostream & **HepMC::operator<< (std::ostream &, PdfInfo const *)**
- **std::istream & HepMC::operator>> (std::istream &, PdfInfo *)**

10.53 Polarization.cc File Reference

```
#include "HepMC/Polarization.h"
```

Namespaces

- namespace **HepMC**

Functions

- std::ostream & **HepMC::operator<< (std::ostream &ostr, const Polarization &polar)**
  print polarization information

10.54 Polarization.h File Reference

```
#include "HepMC/SimpleVector.h"
#include <iostream>
#include <cmath>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::Polarization**
  The Polarization (p. *234*) *class stores theta and phi for a GenParticle* (p. *113*).

Variables

- static const double **HepMC::HepMC_pi = 3.14159265358979323846**

10.55 PythiaHelper.h File Reference

```
#include "HepMC/PythiaWrapper.h"
```

Functions

- void **initPythia** ()

10.55.1 Function Documentation

10.55.1.1 void initPythia ()

Definition at line 12 of file initPythia.cc.

References pydat2, pydatr, pypars, and pysubs.

Referenced by event_selection(), main(), pythia_in_out(), pythia_-out(), pythia_particle_out(), and writePythiaStreamIO().

10.56 PythiaWrapper.h File Reference

```
#include "HepMC/PythiaWrapper6_4.h"
#include <cmath>
#include "HepMC/GenCrossSection.h"
```

Namespaces

- namespace **HepMC**

Functions

- GenCrossSection **HepMC::getPythiaCrossSection** ()
  calculate the Pythia cross section and statistical error

10.57 PythiaWrapper6_4.h File Reference

```
#include <ctype.h>
#include <cstring>
```

Classes

- struct **pin3**
- **struct pin5**
- **struct pin7**
- **struct pin8**
- **struct pin9**
- **struct pssm**
- **struct prvnv**
- **struct prvpm**

Defines

- #define **initpydata initpydata_**
- **#define pyjets pyjets_**
- **#define pydat1 pydat1_**
- **#define pydat2 pydat2_**
- **#define pydat3 pydat3_**
- **#define pydatr pydatr_**
- **#define pysubs pysubs_**
- **#define pypars pypars_**
- **#define pyint1 pyint1_**
- **#define pyint2 pyint2_**
- **#define pyint3 pyint3_**
- **#define pyint4 pyint4_**
- **#define pyint5 pyint5_**
- **#define pyint7 pyint7_**
- **#define pyint8 pyint8_**
- **#define pyint9 pyint9_**
- **#define pyssm pyssm_**
- **#define pyssmt pyssmt_**
- **#define pymsrv pymsrv_**
- **#define pyrvnv pyrvnv_**
- **#define pyrvpm pyrvpm_**
- **#define pyints pyints_**
- **#define pyg2dx pyg2dx_**
- **#define pyhepc pyhepc_**
- **#define pyinit pyinit_**
- **#define pylist pylist_**
- **#define pystat pystat_**
- **#define pyevnt pyevnt_**
- **#define upinit upinit_**
- **#define upevnt upevnt_**
- **#define pydata pydata_**

Functions

- void **initpydata (void)**

Variables

- const int **pyjets_maxn = 4000**
- **struct {**
  int n
  int npad
  int k [5][pyjets_maxn]
  double p [5][pyjets_maxn]
  double v [5][pyjets_maxn]
  } pyjets_

- **struct {**
  int mstu [200]
  double paru [200]
  int mstj [200]
  double parj [200]
  } pydat1_

- **struct {**
  int kchg [4][500]
  double pmas [4][500]
  double parf [2000]
  double vckm [4][4]
  } pydat2_

- **struct {**
  int mdcy [3][500]
  int mdme [2][8000]
  double brat [8000]
  int kfdp [5][8000]
  } pydat3_

- **struct {**
  int mrpy [6]
  double rrpy [100]
  } pydatr_

- **struct {**
  int msel
  int mselpd
  int msub [500]
  int kfin [81][2]
  double ckin [200]
  } pysubs_

- **struct {**
  int mstp [200]
  double parp [200]
  int msti [200]

double pari [200]
} pypars_

- struct {
  int mint [400]
  double vint [400]
  } pyint1_

- struct {
  int iset [500]
  int kfpr [2][500]
  double coef [20][500]
  int icol [2][4][40]
  } pyint2_

- pin3 pyint3_
- struct {
  int mwid [500]
  double wids [5][500]
  } pyint4_

- pin5 pyint5_
- pin7 pyint7_
- pin8 pyint8_
- pin9 pyint9_
- pssm pyssm_
- struct {
  double zmix [4][4]
  double umix [2][2]
  double vmix [2][2]
  double smz [4]
  double smw [2]
  double sfmix [4][16]
  double zmixi [4][4]
  double umixi [2][2]
  double vmixi [2][2]
  } pyssmt_

- struct {
  double rvlam [3][3][3]
  double rvlamp [3][3][3]
  double rvlamb [3][3][3]
  } pymsrv_

- prvnv pyrvnv_
- prvpm pyrvpm_
- struct {
  double xxm [20]
  } pyints_

- struct {
  double x1
  } pyg2dx_

10.57.1 Define Documentation

10.57.1.1 #define initpydata initpydata_

Definition at line 30 of file PythiaWrapper6_4.h.

10.57.1.2 #define pydat1 pydat1_

Definition at line 52 of file PythiaWrapper6_4.h.

10.57.1.3 #define pydat2 pydat2_

Definition at line 60 of file PythiaWrapper6_4.h.
Referenced by initPythia().

10.57.1.4 #define pydat3 pydat3_

Definition at line 69 of file PythiaWrapper6_4.h.

10.57.1.5 #define pydata pydata_

10.57.1.6 #define pydatr pydatr_

Definition at line 77 of file PythiaWrapper6_4.h.
Referenced by initPythia().

10.57.1.7 #define pyevnt pyevnt_

Definition at line 245 of file PythiaWrapper6_4.h.

10.57.1.8 #define pyg2dx pyg2dx_

Definition at line 236 of file PythiaWrapper6_4.h.

10.57.1.9 #define pyhepc pyhepc_

Definition at line 241 of file PythiaWrapper6_4.h.

10.57.1.10 #define pyinit pyinit_

Definition at line 242 of file PythiaWrapper6_4.h.

10.57.1.11 #define pyint1 pyint1_

Definition at line 103 of file PythiaWrapper6_4.h.

10.57.1.12 #define pyint2 pyint2_

Definition at line 112 of file PythiaWrapper6_4.h.

10.57.1.13 #define pyint3 pyint3_

Definition at line 121 of file PythiaWrapper6_4.h.

10.57.1.14 #define pyint4 pyint4_

Definition at line 129 of file PythiaWrapper6_4.h.

10.57.1.15 #define pyint5 pyint5_

Definition at line 137 of file PythiaWrapper6_4.h.
Referenced by HepMC::getPythiaCrossSection().

10.57.1.16 #define pyint7 pyint7_

Definition at line 144 of file PythiaWrapper6_4.h.

10.57.1.17 #define pyint8 pyint8_

Definition at line 155 of file PythiaWrapper6_4.h.

10.57.1.18 #define pyint9 pyint9_

Definition at line 165 of file PythiaWrapper6_4.h.

10.57.1.19 #define pyints pyints_

Definition at line 229 of file PythiaWrapper6_4.h.

10.57.1.20 #define pyjets pyjets_

Definition at line 42 of file PythiaWrapper6_4.h.

10.57.1.21 #define pylist pylist_

Definition at line 243 of file PythiaWrapper6_4.h.

10.57.1.22 #define pymsrv pymsrv_

Definition at line 197 of file PythiaWrapper6_4.h.

10.57.1.23 #define pypars pypars_

Examples:

example_MyPythiaOnlyToHepMC.cc, fio/example_MyPythia.cc, fio/example_PythiaStream-IO.cc, and fio/testPythiaCopies.cc.

Definition at line 95 of file PythiaWrapper6_4.h.

Referenced by event_selection(), initPythia(), main(), pythia_out(), and writePythiaStreamIO().

10.57.1.24 #define pyrvnv pyrvnv_

Definition at line 210 of file PythiaWrapper6_4.h.

10.57.1.25 #define pyrvpm pyrvpm_

Definition at line 222 of file PythiaWrapper6_4.h.

10.57.1.26 #define pyssm pyssm_

Definition at line 173 of file PythiaWrapper6_4.h.

10.57.1.27 #define pyssmt pyssmt_

Definition at line 188 of file PythiaWrapper6_4.h.

10.57.1.28 #define pystat pystat_

Definition at line 244 of file PythiaWrapper6_4.h.

10.57.1.29 #define pysubs pysubs_

Definition at line 85 of file PythiaWrapper6_4.h.

Referenced by initPythia().

10.57.1.30 #define upevnt upevnt_

Definition at line 247 of file PythiaWrapper6_4.h.

10.57.1.31 #define upinit upinit_

Definition at line 246 of file PythiaWrapper6_4.h.

10.57.2 Function Documentation

10.57.2.1 void initpydata (void)

10.57.3 Variable Documentation

10.57.3.1 double brat[8000]

Definition at line 65 of file PythiaWrapper6_4.h.

10.57.3.2 double ckin[200]

Definition at line 82 of file PythiaWrapper6_4.h.

10.57.3.3 double coef[20][500]

Definition at line 108 of file PythiaWrapper6_4.h.

10.57.3.4 int icol[2][4][40]

Definition at line 109 of file PythiaWrapper6_4.h.

10.57.3.5 int iset[500]

Definition at line 107 of file PythiaWrapper6_4.h.

10.57.3.6 int k[5][pyjets_maxn]

Definition at line 38 of file PythiaWrapper6_4.h.

10.57.3.7 int kchg[4][500]

Definition at line 56 of file PythiaWrapper6_4.h.

10.57.3.8 int kfdp[5][8000]

Definition at line 66 of file PythiaWrapper6_4.h.

10.57.3.9 int kfin[81][2]

Definition at line 81 of file PythiaWrapper6_4.h.

10.57.3.10 int kfpr[2][500]

Definition at line 107 of file PythiaWrapper6_4.h.

10.57.3.11 int mdcy[3][500]

Definition at line 64 of file PythiaWrapper6_4.h.

10.57.3.12 int mdme[2][8000]

Definition at line 64 of file PythiaWrapper6_4.h.

10.57.3.13 int mint[400]

Definition at line 99 of file PythiaWrapper6_4.h.

10.57.3.14 int mrpy[6]

Definition at line 73 of file PythiaWrapper6_4.h.

10.57.3.15 int msel

Definition at line 81 of file PythiaWrapper6_4.h.

10.57.3.16 int mselpd

Definition at line 81 of file PythiaWrapper6_4.h.

10.57.3.17 int msti[200]

Definition at line 91 of file PythiaWrapper6_4.h.

10.57.3.18 int mstj[200]

Definition at line 48 of file PythiaWrapper6_4.h.

10.57.3.19 int mstp[200]

Definition at line 89 of file PythiaWrapper6_4.h.

10.57.3.20 int mstu[200]

Definition at line 46 of file PythiaWrapper6_4.h.

10.57.3.21 int msub[500]

Definition at line 81 of file PythiaWrapper6_4.h.

10.57.3.22 int mwid[500]

Definition at line 125 of file PythiaWrapper6_4.h.

10.57.3.23 int n

Definition at line 38 of file PythiaWrapper6_4.h.

10.57.3.24 int npad

Definition at line 38 of file PythiaWrapper6_4.h.

10.57.3.25 double p[5][pyjets_maxn]

Examples:

example_EventSelection.cc, example_UsingIterators.cc, and testHepMCIteration.cc.in.

Definition at line 39 of file PythiaWrapper6_4.h.

Referenced by HepMC::TempParticleMap::addEndParticle(), HepMC::already_in_vector(), HepMC::IO_HERWIG::build_end_vertex(), HepMC::IO_HEPEVT::build_end_vertex(), HepMC::IO_HERWIG::build_particle(), HepMC::IO_HEPEVT::build_particle(), HepMC::IO_HERWIG::build_production_vertex(), HepMC::IO_HEPEVT::build_production_vertex(), HepMC::Flow::connected_partners(), HepMC::Flow::dangling_connected_partners(), HepMC::GenVertex::edge_iterator::edge_iterator(), HepMC::TempParticleMap::end_vertex(), HepMC::IO_HERWIG::find_in_map(), HepMC::IO_HEPEVT::find_in_map(), findPiZero(), HepMC::GenEvent::GenEvent(), IsPhoton(), IsWBoson(), main(), HepMC::not_in_vector(), PrintDescendants::operator()(), PrintChildren::operator()(), PrintParticle::operator()(), PrintPhoton::operator()(), IsFinalState::operator()(), IsGoodEvent::operator()(), IsStateFinal::operator()(), IsW_Boson::operator()(), IsPhoton::operator()(), IsGoodEventMyPythia::operator()(), IsEventGood::operator()(), HepMC::GenVertex::edge_iterator::operator=(), HepMC::GenVertex::particles_in(), HepMC::GenVertex::particles_out(), particleTypes(), HepMC::GenEvent::read(), HepMC::detail::read_particle(), HepMC::GenEvent::remove_barcode(), repairUnits(), HepMC::GenEvent::set_barcode(), HepMC::GenEvent::set_pdf_info(), HepMC::GenEvent::valid_beam_particles(), and HepMC::IO_HEPEVT::write_event().

10.57.3.26 double parf[2000]

Definition at line 57 of file PythiaWrapper6_4.h.

10.57.3.27 double pari[200]

Definition at line 92 of file PythiaWrapper6_4.h.

10.57.3.28 double parj[200]

Definition at line 49 of file PythiaWrapper6_4.h.

10.57.3.29 double parp[200]

Definition at line 90 of file PythiaWrapper6_4.h.

10.57.3.30 double paru[200]

Definition at line 47 of file PythiaWrapper6_4.h.

10.57.3.31 double pmas[4][500]

Definition at line 57 of file PythiaWrapper6_4.h.

10.57.3.32 struct { ... } pydat1_

10.57.3.33 struct { ... } pydat2_

10.57.3.34 struct { ... } pydat3_

10.57.3.35 struct { ... } pydatr_

10.57.3.36 struct { ... } pyg2dx_

10.57.3.37 struct { ... } pyint1_

10.57.3.38 struct { ... } pyint2_

10.57.3.39 struct pin3 pyint3_

10.57.3.40 struct { ... } pyint4_

10.57.3.41 struct pin5 pyint5_

10.57.3.42 struct pin7 pyint7_

10.57.3.43 struct pin8 pyint8_

10.57.3.44 struct pin9 pyint9_

10.57.3.45 struct { ... } pyints_

10.57.3.46 struct { ... } pyjets_

10.57.3.47 const int pyjets_maxn = 4000

Definition at line 35 of file PythiaWrapper6_4.h.

10.57.3.48 struct { ... } pymsrv_

10.57.3.49 struct { ... } pypars_

10.57.3.50 struct prvnv pyrvnv_

10.57.3.51 struct prvpm pyrvpm_

10.57.3.52 struct pssm pyssm_

10.57.3.53 struct { ... } pyssmt_

10.57.3.54 struct { ... } pysubs_

10.57.3.55 double rrpy[100]

Definition at line 74 of file PythiaWrapper6_4.h.

10.57.3.56 double rvlam[3][3][3]

Definition at line 192 of file PythiaWrapper6_4.h.

10.57.3.57 double rvlamb[3][3][3]

Definition at line 194 of file PythiaWrapper6_4.h.

10.57.3.58 double rvlamp[3][3][3]

Definition at line 193 of file PythiaWrapper6_4.h.

10.57.3.59 double sfmix[4][16]

Definition at line 182 of file PythiaWrapper6_4.h.

10.57.3.60 double smw[2]

Definition at line 181 of file PythiaWrapper6_4.h.

10.57.3.61 double smz[4]

Definition at line 180 of file PythiaWrapper6_4.h.

10.57.3.62 double umix[2][2]

Definition at line 178 of file PythiaWrapper6_4.h.

10.57.3.63 double umixi[2][2]

Definition at line 184 of file PythiaWrapper6_4.h.

10.57.3.64 double v[5][pyjets_maxn]

Examples:

example_UsingIterators.cc, testHepMCIteration.cc.in, and VectorConversion.h.

Definition at line 39 of file PythiaWrapper6_4.h.

Referenced by HepMC::compareVertices(), convertTo(), HepMC::GenEvent::GenEvent(), main(), HepMC::GenEvent::read(), HepMC::detail::read_vertex(), HepMC::GenEvent::remove_barcode(), HepMC::GenEvent::set_barcode(), HepMC::GenEvent::write(), and HepMC::IO_HEPEVT::write_event().

10.57.3.65 double vckm[4][4]

Definition at line 57 of file PythiaWrapper6_4.h.

10.57.3.66 double vint[400]

Definition at line 100 of file PythiaWrapper6_4.h.

10.57.3.67 double vmix[2][2]

Definition at line 179 of file PythiaWrapper6_4.h.

10.57.3.68 double vmixi[2][2]

Definition at line 185 of file PythiaWrapper6_4.h.

10.57.3.69 double wids[5][500]

Definition at line 126 of file PythiaWrapper6_4.h.

10.57.3.70 double x1

Examples:

testMass.cc.in.

Definition at line 233 of file PythiaWrapper6_4.h.

Referenced by HepMC::operator>>().

10.57.3.71 double xxm[20]

Definition at line 226 of file PythiaWrapper6_4.h.

10.57.3.72 double zmix[4][4]

Definition at line 177 of file PythiaWrapper6_4.h.

10.57.3.73 double zmixi[4][4]

Definition at line 183 of file PythiaWrapper6_4.h.

10.58 PythiaWrapper6_4_WIN32.h File Reference

10.59 SearchVector.cc File Reference

```
#include "HepMC/SearchVector.h"
```

Namespaces

- namespace **HepMC**

Functions

- bool **HepMC::not_in_vector (std::vector< HepMC::GenParticle * > *, GenParticle *)**

  returns true if it cannot find GenParticle in the vector*

- **std::vector< HepMC::GenParticle * >::iterator HepMC::already_in_vector (std::vector< GenParticle * > *v, GenParticle *p)**

  returns true if GenParticle (p. *113) is in the vector*

10.60 SearchVector.h File Reference

```
#include "HepMC/GenVertex.h"
#include "HepMC/GenParticle.h"
```

Namespaces

- namespace **HepMC**

Functions

- bool **HepMC::not_in_vector (std::vector< HepMC::GenParticle * > *, GenParticle *)**

  returns true if it cannot find GenParticle* in the vector

- **std::vector< HepMC::GenParticle * >::iterator HepMC::already_in_vector (std::vector< GenParticle * > *v, GenParticle *p)**

  returns true if GenParticle (p. 113) is in the vector

10.61 SimpleVector.h File Reference

```
#include "HepMC/enable_if.h"
#include "HepMC/is_arithmetic.h"
#include "HepMC/SimpleVector.icc"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::FourVector**

  FourVector (p. ***61***) ***is a simple representation of a physics 4 vector.***

- **class HepMC::ThreeVector**

  ThreeVector (p. ***256***) ***is a simple representation of a position or displacement 3 vector.***

10.62 StreamHelpers.cc File Reference

```
#include <ostream>
#include <istream>
#include <sstream>
#include "HepMC/GenVertex.h"
#include "HepMC/GenParticle.h"
#include "HepMC/StreamHelpers.h"
#include "HepMC/IO_Exception.h"
```

Namespaces

- namespace **HepMC**
- **namespace HepMC::detail**

Functions

- std::istream & **HepMC::detail::read_vertex (std::istream &, TempParticleMap &, GenVertex ∗)**
- **std::istream & HepMC::detail::find_event_end (std::istream &)**

  used to read to the end of a bad event

10.63 StreamHelpers.h File Reference

```
#include <ostream>
#include <istream>
#include "HepMC/GenEvent.h"
#include "HepMC/TempParticleMap.h"
```

Namespaces

- namespace **HepMC**
- **namespace HepMC::detail**

Functions

- std::ostream & **HepMC::detail::establish_output_stream_info (std::ostream &)**
  *used by IO_GenEvent (*p. *186) constructor*
- **std::istream & HepMC::detail::establish_input_stream_info (std::istream &)**
  *used by IO_GenEvent (*p. *186) constructor*
- **std::istream & HepMC::detail::read_vertex (std::istream &, TempParticleMap &, GenVertex ∗)**
- **std::istream & HepMC::detail::read_particle (std::istream &, TempParticleMap &, GenParticle ∗)**
- **std::ostream & HepMC::detail::output (std::ostream &os, const double &d)**
  write a double - for internal use by streaming IO
- **std::ostream & HepMC::detail::output (std::ostream &os, const float &d)**
  write a float - for internal use by streaming IO
- **std::ostream & HepMC::detail::output (std::ostream &os, const int &i)**
  write an int - for internal use by streaming IO
- **std::ostream & HepMC::detail::output (std::ostream &os, const long &i)**
  write a long - for internal use by streaming IO
- **std::ostream & HepMC::detail::output (std::ostream &os, const char &c)**
  write a single char - for internal use by streaming IO
- **std::istream & HepMC::detail::find_event_end (std::istream &)**
  used to read to the end of a bad event

10.64 StreamInfo.cc File Reference

```
#include <string>
#include "HepMC/StreamInfo.h"
```

Namespaces

- namespace **HepMC**

10.65 StreamInfo.h File Reference

```
#include <string>
#include "HepMC/Units.h"
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::StreamInfo**

  StreamInfo (p. ***247***) ***contains extra information needed when using streaming IO.***

Enumerations

- enum **HepMC::known_io {**
  HepMC::gen = 1, HepMC::ascii, HepMC::extascii, HepMC::ascii_pdt,
  HepMC::extascii_pdt }

  The known_io enum is used to track which type of input is being read.

10.66 TempParticleMap.h File Reference

```
#include <map>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::TempParticleMap**

  TempParticleMap (p. ***253***) ***is a temporary GenParticle∗ container used during input.***

10.67 testFlow.cc File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include "HepMC/GenEvent.h"
#include "HepMC/IO_GenEvent.h"
```

Typedefs

- typedef std::vector< **HepMC::GenParticle** * > **FlowVec**

Functions

- int **main** ()

10.67.1 Typedef Documentation

10.67.1.1 typedef std::vector<HepMC::GenParticle*> FlowVec

Definition at line 15 of file testFlow.cc.

10.67.2 Function Documentation

10.67.2.1 int main ()

Definition at line 17 of file testFlow.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::GenParticle::barcode(), HepMC::Flow::erase(), HepMC::GenParticle::flow(), HepMC::Units::GEV, HepMC::Units::MM, HepMC::GenEvent::print(), HepMC::GenParticle::set_flow(), HepMC::GenEvent::set_signal_process_vertex(), HepMC::GenEvent::use_units(), and HepMC::GenEvent::write().

10.68 testHepMCIteration.h File Reference

Classes

- class **IsFinalState**
- **class PrintPhoton**
- **class PrintParticle**
- **class PrintChildren**

  test class

- **class PrintDescendants**

  test class

Functions

- bool **IsPhoton (const HepMC::GenParticle ∗p)**

  returns true if the GenParticle particle is a photon with pT > 10 GeV

- **bool IsWBoson (const HepMC::GenParticle ∗p)**

  returns true if the GenParticle is a W+/W-

10.68.1 Function Documentation

10.68.1.1 bool IsPhoton (const HepMC::GenParticle ∗ *p*)

returns true if the GenParticle particle is a photon with pT > 10 GeV

Examples:

testHepMCIteration.cc.in.

Definition at line 10 of file testHepMCIteration.h.
References p.
Referenced by PrintPhoton::operator()().

10.68.1.2 bool IsWBoson (const HepMC::GenParticle ∗ *p*)

returns true if the GenParticle is a W+/W-

Examples:

testHepMCIteration.cc.in.

Definition at line 17 of file testHepMCIteration.h.
References p.

10.69 testHepMCMethods.cc File Reference

```
#include "testHepMCMethods.h"
```

Functions

- double **findPiZero (HepMC::GenEvent ∗evt)**
- **void particleTypes (HepMC::GenEvent ∗evt, std::ostream &os)**
- **void repairUnits (HepMC::GenEvent ∗evt, HepMC::Units::MomentumUnit from, HepMC::Units::MomentumUnit to)**

10.69.1 Function Documentation

10.69.1.1 double findPiZero (HepMC::GenEvent ∗ *evt*)

Examples:

testHepMC.cc.in, and testStreamIO.cc.in.

Definition at line 11 of file testHepMCMethods.cc.

References p, HepMC::GenEvent::particles_begin(), and HepMC::GenEvent::particles_end().

10.69.1.2 void particleTypes (HepMC::GenEvent ∗ *evt*, std::ostream & *os*)

Examples:

testHepMC.cc.in, and testStreamIO.cc.in.

Definition at line 22 of file testHepMCMethods.cc.

References HepMC::GenEvent::event_number(), p, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), and HepMC::GenEvent::particles_size().

10.69.1.3 void repairUnits (HepMC::GenEvent ∗ *evt*, HepMC::Units::MomentumUnit *from*, HepMC::Units::MomentumUnit *to*)

Examples:

testHepMC.cc.in.

Definition at line 78 of file testHepMCMethods.cc.

References HepMC::Units::conversion_factor(), HepMC::FourVector::e(), p, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::FourVector::py(), and HepMC::FourVector::pz().

10.70 testHepMCMethods.h File Reference

```
#include "HepMC/GenEvent.h"
```

Functions

- double **findPiZero (HepMC::GenEvent ∗)**
- **void particleTypes (HepMC::GenEvent ∗, std::ostream &os=std::cout)**
- **void repairUnits (HepMC::GenEvent ∗, HepMC::Units::MomentumUnit, HepMC::Units::MomentumUnit)**

10.70.1 Function Documentation

10.70.1.1 double findPiZero (HepMC::GenEvent ∗)

Definition at line 11 of file testHepMCMethods.cc.

References p, HepMC::GenEvent::particles_begin(), and HepMC::GenEvent::particles_end().

10.70.1.2 void particleTypes (HepMC::GenEvent ∗, std::ostream & *os* = std::cout)

Definition at line 22 of file testHepMCMethods.cc.

References HepMC::GenEvent::event_number(), p, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), and HepMC::GenEvent::particles_size().

10.70.1.3 void repairUnits (HepMC::GenEvent ∗, HepMC::Units::MomentumUnit, HepMC::Units::MomentumUnit)

Definition at line 78 of file testHepMCMethods.cc.

References HepMC::Units::conversion_factor(), HepMC::FourVector::e(), p, HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::FourVector::py(), and HepMC::FourVector::pz().

10.71 testHerwigCopies.cc File Reference

```
#include <fstream>
#include <iostream>
#include "HepMC/HerwigWrapper.h"
#include "HepMC/IO_HERWIG.h"
#include "HepMC/GenEvent.h"
#include "HepMC/CompareGenEvent.h"
#include "HepMC/HEPEVT_Wrapper.h"
```

Functions

- int **main** ()

10.71.1 Function Documentation

10.71.1.1 int main ()

Definition at line 16 of file testHerwigCopies.cc.

References HepMC::compareGenEvent(), HepMC::GenEvent::event_number(), HepMC::getHerwigCrossSection(), HepMC::Units::GEV, hwbgen, hwbmch, hwcdec, hwcfor, hwdhad, hwdhob, hwdhvy, hwefin, hweini, hwepro, hwevnt, hwigin, hwmevt, hwproc, hwufne, hwuinc, hwuine, HepMC::Units::MM, HepMC::GenEvent::print(), HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::GenEvent::set_event_number(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_signal_process_id(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), and HepMC::GenEvent::use_units().

10.72 testPolarization.cc File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include "HepMC/GenEvent.h"
#include "HepMC/IO_GenEvent.h"
```

Functions

- int **main** ()

10.72.1 Function Documentation

10.72.1.1 int main ()

Definition at line 14 of file testPolarization.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::GenEvent::particles_begin(), HepMC::GenEvent::particles_end(), HepMC::GenEvent::print(), HepMC::GenParticle::set_flow(), HepMC::GenParticle::set_polarization(), HepMC::GenEvent::set_signal_process_vertex(), and HepMC::GenEvent::write().

10.73 testPrintBug.cc File Reference

```
#include <fstream>
#include "HepMC/GenEvent.h"
#include "HepMC/SimpleVector.h"
```

Functions

- int **main** ()

10.73.1 Function Documentation

10.73.1.1 int main ()

Definition at line 10 of file testPrintBug.cc.

References HepMC::GenVertex::add_particle_in(), HepMC::GenVertex::add_particle_out(), HepMC::GenEvent::add_vertex(), HepMC::Units::GEV, HepMC::Units::MM, HepMC::GenEvent::print(), and HepMC::GenEvent::use_units().

10.74 testPythiaCopies.cc File Reference

```
#include <fstream>
#include <iostream>
#include "HepMC/PythiaWrapper.h"
#include "HepMC/IO_HEPEVT.h"
#include "HepMC/GenEvent.h"
#include "HepMC/CompareGenEvent.h"
#include "PythiaHelper.h"
```

Functions

- int **main** ()

10.74.1 Function Documentation

10.74.1.1 int main ()

Definition at line 16 of file testPythiaCopies.cc.

References HepMC::compareGenEvent(), HepMC::GenEvent::event_number(), HepMC::getPythiaCrossSection(), HepMC::Units::GEV, initPythia(), HepMC::Units::MM, HepMC::GenEvent::print(), pypars, HepMC::IO_BaseClass::read_next_event(), HepMC::GenEvent::set_cross_section(), HepMC::HEPEVT_Wrapper::set_max_number_entries(), HepMC::GenEvent::set_mpi(), HepMC::HEPEVT_Wrapper::set_sizeof_real(), HepMC::GenEvent::use_units(), and HepMC::GenEvent::weights().

10.75 testSimpleVector.cc File Reference

```
#include <iostream>
#include "HepMC/SimpleVector.h"
```

Functions

- int **main** ()

10.75.1 Function Documentation

10.75.1.1 int main ()

Definition at line 8 of file testSimpleVector.cc.

References HepMC::FourVector::e(), HepMC::FourVector::eta(), HepMC::FourVector::m(), HepMC::FourVector::m2(), HepMC::FourVector::perp(), HepMC::ThreeVector::perp(), HepMC::FourVector::perp2(), HepMC::ThreeVector::perp2(), HepMC::FourVector::phi(), HepMC::ThreeVector::phi(), HepMC::FourVector::pseudoRapidity(), HepMC::FourVector::px(), HepMC::FourVector::py(), HepMC::FourVector::pz(), HepMC::ThreeVector::r(), HepMC::FourVector::rho(), HepMC::FourVector::set(), HepMC::ThreeVector::set(), HepMC::FourVector::setE(), HepMC::ThreeVector::setPhi(), HepMC::FourVector::setPx(), HepMC::FourVector::setPy(), HepMC::FourVector::setPz(), HepMC::FourVector::setT(), HepMC::ThreeVector::setTheta(), HepMC::FourVector::setX(), HepMC::ThreeVector::setX(), HepMC::FourVector::setY(), HepMC::ThreeVector::setY(), HepMC::FourVector::setZ(), HepMC::ThreeVector::setZ(), HepMC::FourVector::t(), HepMC::FourVector::theta(), HepMC::ThreeVector::theta(), HepMC::FourVector::x(), HepMC::ThreeVector::x(), HepMC::FourVector::y(), HepMC::ThreeVector::y(), HepMC::FourVector::z(), and HepMC::ThreeVector::z().

10.76 testUnits.cc File Reference

```
#include <iostream>
#include "HepMC/Units.h"
```

Functions

- int **main** ()

10.76.1 Function Documentation

10.76.1.1 int main ()

Definition at line 8 of file testUnits.cc.

References HepMC::Units::CM, HepMC::Units::conversion_factor(), HepMC::Units::default_length_unit(), HepMC::Units::default_momentum_unit(), HepMC::Units::GEV, HepMC::Units::MEV, HepMC::Units::MM, and HepMC::Units::name().

10.77 testWeights.cc File Reference

```
#include <assert.h>
#include <iostream>
#include <string>
#include <vector>
#include "HepMC/WeightContainer.h"
#include <stdexcept>
```

Functions

- int **main** ()

10.77.1 Function Documentation

10.77.1.1 int main ()

Definition at line 16 of file testWeights.cc.

References HepMC::WeightContainer::empty(), HepMC::WeightContainer::has_key(), HepMC::WeightContainer::pop_back(), HepMC::WeightContainer::push_back(), HepMC::WeightContainer::size(), and HepMC::WeightContainer::write().

10.78 Units.h File Reference

```
#include <iostream>
#include <string>
```

Namespaces

- namespace **HepMC**
- **namespace Units**
- **namespace HepMC::Units**

Enumerations

- enum **HepMC::Units::MomentumUnit { HepMC::Units::MEV, HepMC::Units::GEV }**
- **enum HepMC::Units::LengthUnit { HepMC::Units::MM, HepMC::Units::CM }**

Functions

- **LengthUnit HepMC::Units::default_length_unit ()**
  default unit is defined by configure
- **MomentumUnit HepMC::Units::default_momentum_unit ()**
  default unit is defined by configure
- **std::string HepMC::Units::name (MomentumUnit)**
  convert enum to string
- **std::string HepMC::Units::name (LengthUnit)**
  convert enum to string
- **double HepMC::Units::conversion_factor (MomentumUnit from, MomentumUnit to)**
  scaling factor relative to MeV
- **double HepMC::Units::conversion_factor (LengthUnit from, LengthUnit to)**

10.79 VectorConversion.h File Reference

```
#include "HepMC/SimpleVector.h"
#include "CLHEP/Vector/LorentzVector.h"
```

Namespaces

- namespace **CLHEP**

Functions

- CLHEP::Hep3Vector **convertTo (const HepMC::ThreeVector &v)**
  Convert from HepMC::ThreeVector (p. ***256) to CLHEP::Hep3Vector.***
- **CLHEP::HepLorentzVector convertTo (const HepMC::FourVector &v)**
  Convert from HepMC::FourVector (p. ***61) to CLHEP::HepLorentzVector.***

10.79.1 Function Documentation

10.79.1.1 CLHEP::HepLorentzVector convertTo (const HepMC::FourVector & *v*) [inline]

Convert from **HepMC::FourVector** (p. 61) to CLHEP::HepLorentzVector.

Definition at line 25 of file VectorConversion.h.

References v.

10.79.1.2 CLHEP::Hep3Vector convertTo (const HepMC::ThreeVector & *v*) [inline]

Convert from **HepMC::ThreeVector** (p. 256) to CLHEP::Hep3Vector.

Examples:

example_VectorConversion.cc, and VectorConversion.h.

Definition at line 21 of file VectorConversion.h.

References v.

Referenced by main().

10.80 Version.h File Reference

```
#include <string>
#include <iostream>
#include "HepMC/HepMCDefs.h"
```

Namespaces

- namespace **HepMC**

Functions

- void **HepMC::version (std::ostream &os=std::cout)**
  *print HepMC (*p. *25) version*
- **void HepMC::writeVersion (std::ostream &os)**
  *write HepMC (*p. *25) version to os*
- **std::string HepMC::versionName ()**
  *return HepMC (*p. *25) version*

10.81 WeightContainer.cc File Reference

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <vector>
#include <string>
#include <map>
#include <stdexcept>
#include "HepMC/WeightContainer.h"
```

Namespaces

- namespace **HepMC**

10.82 WeightContainer.h File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
```

Namespaces

- namespace **HepMC**

Classes

- class **HepMC::WeightContainer**

  Container for the Weights associated with an event or vertex.

Chapter 11

HepMC Example Documentation

11.1 example_BuildEventFromScratch.cc

Example of building an event and a particle data table from scratch
This is meant to be of use for persons implementing **HepMC** (p.25) inside a MC event generator

```
1
2 // Matt.Dobbs@Cern.CH, Feb 2000
3 // Example of building an event and a particle data table from scratch
4 // This is meant to be of use for persons implementing HepMC inside a MC
5 // event generator
7 // To Compile: go to the HepMC directory and type:
8 // gmake examples/example_BuildEventFromScratch.exe
9 //
10
11 #include <iostream>
12
13 #include "HepMC/GenEvent.h"
14
15 // in this example we use the HepMC namespace, so that we do not have to
16 // precede all HepMC classes with HepMC::
17
18 // This example also shows how to use the CLHEP Lorentz vector with HepMC2
19
20 using namespace HepMC;
21
22 int main() {
23     //
24     // In this example we will place the following event into HepMC "by hand"
25     //
26     //     name status pdg_id  parent Px       Py    Pz       Energy      Mass
27     //  1  !p+!    3   2212    0,0    0.000    0.000 7000.000 7000.000    0.938
28     //  2  !p+!    3   2212    0,0    0.000    0.000-7000.000 7000.000    0.938
29     //=============================================================================
30     //  3  !d!     3      1    1,1    0.750   -1.569   32.191   32.238    0.000
31     //  4  !u~!    3     -2    2,2   -3.047  -19.000  -54.629   57.920    0.000
32     //  5  !W-!    3    -24    1,2    1.517   -20.68  -20.605   85.925   80.799
33     //  6  !gamma! 1     22    1,2   -3.813    0.113   -1.833    4.233    0.000
34     //  7  !d!     1      1    5,5   -2.445   28.816    6.082   29.552    0.010
35     //  8  !u~!    1     -2    5,5    3.962  -49.498  -26.687   56.373    0.006
36
37     // now we build the graph, which will look like
38     //                        p7                          #
39     // p1                    /                            #
40     //   \v1__p3      p5---v4                             #
```

```
41     //         \_v3_/       \                          #
42     //         /    \        p8                        #
43     //    v2__p4     \                                  #
44     //   /            p6                                #
45     // p2                                               #
46     //                                                  #
47
48     // First create the event container, with Signal Process 20, event number 1
49     //
50     GenEvent* evt = new GenEvent( 20, 1 );
51     // define the units
52     evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
53     //
54     // create vertex 1 and vertex 2, together with their inparticles
55     GenVertex* v1 = new GenVertex();
56     evt->add_vertex( v1 );
57     v1->add_particle_in( new GenParticle( FourVector(0,0,7000,7000),
58                                       2212, 3 ) );
59     GenVertex* v2 = new GenVertex();
60     evt->add_vertex( v2 );
61     v2->add_particle_in( new GenParticle( FourVector(0,0,-7000,7000),
62                                       2212, 3 ) );
63     //
64     // create the outgoing particles of v1 and v2
65     GenParticle* p3 =
66         new GenParticle( FourVector(.750,-1.569,32.191,32.238), 1, 3 );
67     v1->add_particle_out( p3 );
68     GenParticle* p4 =
69         new GenParticle( FourVector(-3.047,-19.,-54.629,57.920), -2, 3 );
70     v2->add_particle_out( p4 );
71     //
72     // create v3
73     GenVertex* v3 = new GenVertex();
74     evt->add_vertex( v3 );
75     v3->add_particle_in( p3 );
76     v3->add_particle_in( p4 );
77     v3->add_particle_out(
78         new GenParticle( FourVector(-3.813,0.113,-1.833,4.233 ), 22, 1 )
79         );
80     GenParticle* p5 =
81         new GenParticle( FourVector(1.517,-20.68,-20.605,85.925), -24,3);
82     v3->add_particle_out( p5 );
83     //
84     // create v4
85     GenVertex* v4 = new GenVertex(FourVector(0.12,-0.3,0.05,0.004));
86     evt->add_vertex( v4 );
87     v4->add_particle_in( p5 );
88     v4->add_particle_out(
89         new GenParticle( FourVector(-2.445,28.816,6.082,29.552), 1,1 )
90         );
91     v4->add_particle_out(
92         new GenParticle( FourVector(3.962,-49.498,-26.687,56.373), -2,1 )
93         );
94     //
95     // tell the event which vertex is the signal process vertex
96     evt->set_signal_process_vertex( v3 );
97     // the event is complete, we now print it out to the screen
98     evt->print();
99
100     // now clean-up by deleteing all objects from memory
101     //
102     // deleting the event deletes all contained vertices, and all particles
103     // contained in those vertices
104     delete evt;
105
106     return 0;
107 }
```

11.2 example_EventSelection.cc

Example of applying an event selection to the events written to file using example_MyPythia.cxx Events containing a photon of pT > 25 GeV pass the selection and are written to "example_EventSelection.dat"

```
1
2 // Matt.Dobbs@Cern.CH, Feb 2000
3 // Example of applying an event selection to the events written to file
4 // using example_MyPythia.cxx
5 // Events containing a photon of pT > 25 GeV pass the selection and are
6 // written to "example_EventSelection.dat"
8 // To Compile: go to the HepMC directory and type:
9 // gmake examples/example_EventSelection.exe
10 //
11
12 #include "HepMC/IO_GenEvent.h"
13 #include "HepMC/GenEvent.h"
14
16
20 class IsEventGood {
21 public:
23     bool operator()( const HepMC::GenEvent* evt ) {
24         for ( HepMC::GenEvent::particle_const_iterator p
25                   = evt->particles_begin(); p != evt->particles_end(); ++p ){
26             if ( (*p)->pdg_id() == 22 && (*p)->momentum().perp() > 25. ) {
27                 //std::cout << "Event " << evt->event_number()
28                 //     << " is a good event." << std::endl;
29                 //(*p)->print();
30                 return 1;
31             }
32         }
33         return 0;
34     }
35 };
36
37 int main() {
38     // declare an input strategy to read the data produced with the
39     // example_MyPythia
40     { // begin scope of ascii_in and ascii_out
41         HepMC::IO_GenEvent ascii_in("example_MyPythia.dat",std::ios::in);
42         // declare another IO_GenEvent for writing out the good events
43         HepMC::IO_GenEvent ascii_out("example_EventSelection.dat",std::ios::out);
44         // declare an instance of the event selection predicate
45         IsEventGood is_good_event;
46         //.....................................EVENT LOOP
47         int icount=0;
48         int num_good_events=0;
49         HepMC::GenEvent* evt = ascii_in.read_next_event();
50         while ( evt ) {
51             icount++;
52             if ( icount%50==1 ) std::cout << "Processing Event Number " << icount
53                                           << " its # " << evt->event_number()
54                                           << std::endl;
55             if ( is_good_event(evt) ) {
56                 ascii_out << evt;
57                 ++num_good_events;
58             }
59             delete evt;
60             ascii_in >> evt;
61         }
62         //.....................................PRINT RESULT
63         std::cout << num_good_events << " out of " << icount
64                   << " processed events passed the cuts. Finished." << std::endl;
65     } // end scope of ascii_in and ascii_out
66     return 0;
```

```
67 }
68
69
70
71
72
73
74
75
76
77
```

11.3 example_MyPythiaOnlyToHepMC.cc

Example of generating events with Pythia using **HepMC/PythiaWrapper.h** (p. 344) Events are read into the **HepMC** (p. 25) event record from the FORTRAN HEPEVT common block using the IO_HEPEVT strategy - nothing is done with them. This program is just used to find the total time required to transfer from HEPEVT into the **HepMC** (p. 25) event record.

```
1
2 // Matt.Dobbs@Cern.CH, December 1999
3 // November 2000, updated to use Pythia 6.1
4 // example of generating events with Pythia
5 // using HepMC/PythiaWrapper.h
6 // Events are read into the HepMC event record from the FORTRAN HEPEVT
7 // common block using the IO_HEPEVT strategy -- nothing is done with them.
8 // This program is just used to find the total time required to transfer
9 // from HEPEVT into the HepMC event record.
11 // To Compile: go to the HepMC directory and type:
12 // gmake examples/example_MyPythiaOnlyTo HepMC.exe
13 //
14 // See comments in examples/example_MyPythia.cxx regarding the HEPEVT wrapper.
15 //
16
17 #include <iostream>
18 #include "HepMC/PythiaWrapper.h"
19 #include "HepMC/IO_HEPEVT.h"
20 #include "HepMC/GenEvent.h"
21 #include "PythiaHelper.h"
22
23 int main() {
24     //
25     //.....................................HEPEVT
26     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
27     //  numbers. We need to explicitly pass this information to the
28     //  HEPEVT_Wrapper.
29     //
30     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
31     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
32     //
33     //.....................................PYTHIA INITIALIZATIONS
34     initPythia();
35     //
36     //.....................................HepMC INITIALIZATIONS
37     //
38     // Instantiate an IO strategy for reading from HEPEVT.
39     HepMC::IO_HEPEVT hepevtio;
40     //
41     //.....................................EVENT LOOP
42     for ( int i = 1; i <= 100; i++ ) {
43         if ( i%50==1 ) std::cout << "Processing Event Number "
44                                  << i << std::endl;
45         call_pyevnt();      // generate one event with Pythia
46         // pythia pyhepc routine convert common PYJETS in common HEPEVT
47         call_pyhepc( 1 );
48         HepMC::GenEvent* evt = hepevtio.read_next_event();
49         // define the units (Pythia uses GeV and mm)
50         evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
51         // set number of multi parton interactions
52         evt->set_mpi( pypars.msti[31-1] );
53         // set cross section information
54         evt->set_cross_section( HepMC::getPythiaCrossSection() );
55         //
56         //....................USER WOULD PROCESS EVENT HERE
57         //
58         // we also need to delete the created event from memory
```

```
59         delete evt;
60     }
61     //.....................................TERMINATION
62     // write out some information from Pythia to the screen
63     call_pystat( 1 );
64
65     return 0;
66 }
67
68
69
```

11.4 example_UsingIterators.cc

This example shows low to use the particle and vertex iterators

```
1
2 // Matt.Dobbs@Cern.CH, Feb 2000
3 // This example shows low to use the particle and vertex iterators
5 // To Compile: go to the HepMC directory and type:
6 // gmake examples/example_UsingIterators.exe
7 //
8
9 #include "HepMC/IO_GenEvent.h"
10 #include "HepMC/GenEvent.h"
11 #include <math.h>
12 #include <algorithm>
13 #include <list>
14
16
20 class IsPhoton {
21 public:
23     bool operator()( const HepMC::GenParticle* p ) {
24         if ( p->pdg_id() == 22
25              && p->momentum().perp() > 10. ) return 1;
26         return 0;
27     }
28 };
29
31
34 class IsW_Boson {
35 public:
37     bool operator()( const HepMC::GenParticle* p ) {
38         if ( abs(p->pdg_id()) == 24 ) return 1;
39         return 0;
40     }
41 };
42
44
47 class IsStateFinal {
48 public:
50     bool operator()( const HepMC::GenParticle* p ) {
51         if ( !p->end_vertex() && p->status()==1 ) return 1;
52         return 0;
53     }
54 };
55
56 int main() {
57     { // begin scope of ascii_in
58         // an event has been prepared in advance for this example, read it
59         // into memory using the IO_GenEvent input strategy
60         HepMC::IO_GenEvent ascii_in("example_UsingIterators.txt",std::ios::in);
61         if ( ascii_in.rdstate() == std::ios::failbit ) {
62             std::cerr << "ERROR input file example_UsingIterators.txt is needed "
63                       << "and does not exist. "
64                       << "\n Look for it in HepMC/examples, Exit." << std::endl;
65             return 1;
66         }
67
68         HepMC::GenEvent* evt = ascii_in.read_next_event();
69
70         // if you wish to have a look at the event, then use evt->print();
71
72         // use GenEvent::vertex_iterator to fill a list of all
73         // vertices in the event
74         std::list<HepMC::GenVertex*> allvertices;
75         for ( HepMC::GenEvent::vertex_iterator v = evt->vertices_begin();
76               v != evt->vertices_end(); ++v ) {
```

```
77              allvertices.push_back(*v);
78          }
79
80          // we could do the same thing with the STL algorithm copy
81          std::list<HepMC::GenVertex*> allvertices2;
82          copy( evt->vertices_begin(), evt->vertices_end(),
83                back_inserter(allvertices2) );
84
85          // fill a list of all final state particles in the event, by requiring
86          // that each particle satisfyies the IsStateFinal predicate
87          IsStateFinal isfinal;
88          std::list<HepMC::GenParticle*> finalstateparticles;
89          for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
90                p != evt->particles_end(); ++p ) {
91              if ( isfinal(*p) ) finalstateparticles.push_back(*p);
92          }
93
94          // an STL-like algorithm called HepMC::copy_if is provided in the
95          // GenEvent.h header to do this sort of operation more easily,
96          // you could get the identical results as above by using:
97          std::list<HepMC::GenParticle*> finalstateparticles2;
98          HepMC::copy_if( evt->particles_begin(), evt->particles_end(),
99                          back_inserter(finalstateparticles2), IsStateFinal() );
100
101          // lets print all photons in the event that satisfy the IsPhoton criteria
102          IsPhoton isphoton;
103          for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
104                p != evt->particles_end(); ++p ) {
105              if ( isphoton(*p) ) (*p)->print();
106          }
107
108          // the GenVertex::particle_iterator and GenVertex::vertex_iterator
109          // are slightly different from the GenEvent:: versions, in that
110          // the iterator starts at the given vertex, and walks through the attached
111          // vertex returning particles/vertices.
112          // Thus only particles/vertices which are in the same graph as the given
113          // vertex will be returned. A range is specified with these iterators,
114          // the choices are:
115          //    parents, children, family, ancestors, descendants, relatives
116          // here are some examples.
117
118          // use GenEvent::particle_iterator to find all W's in the event,
119          // then
120          // (1) for each W user the GenVertex::particle_iterator with a range of
121          //     parents to return and print the immediate mothers of these W's.
122          // (2) for each W user the GenVertex::particle_iterator with a range of
123          //     descendants to return and print all descendants of these W's.
124          IsW_Boson isw;
125          for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
126                p != evt->particles_end(); ++p ) {
127              if ( isw(*p) ) {
128                  std::cout << "A W boson has been found: " << std::endl;
129                  (*p)->print();
130                  // return all parents
131                  // we do this by pointing to the production vertex of the W
132                  // particle and asking for all particle parents of that vertex
133                  std::cout << "\t Its parents are: " << std::endl;
134                  if ( (*p)->production_vertex() ) {
135                      for ( HepMC::GenVertex::particle_iterator mother
136                                = (*p)->production_vertex()->
137                                  particles_begin(HepMC::parents);
138                            mother != (*p)->production_vertex()->
139                                  particles_end(HepMC::parents);
140                            ++mother ) {
141                          std::cout << "\t";
142                          (*mother)->print();
143                      }
```

```
144                 }
145                 // return all descendants
146                 // we do this by pointing to the end vertex of the W
147                 // particle and asking for all particle descendants of that vertex
148                 std::cout << "\t\t Its descendants are: " << std::endl;
149                 if ( (*p)->end_vertex() ) {
150                     for ( HepMC::GenVertex::particle_iterator des
151                               =(*p)->end_vertex()->
152                               particles_begin(HepMC::descendants);
153                           des != (*p)->end_vertex()->
154                               particles_end(HepMC::descendants);
155                           ++des ) {
156                         std::cout << "\t\t";
157                         (*des)->print();
158                     }
159                 }
160             }
161         }
162         // cleanup
163         delete evt;
164         // in analogy to the above, similar use can be made of the
165         // HepMC::GenVertex::vertex_iterator, which also accepts a range.
166     } // end scope of ascii_in
167
168     return 0;
169 }
```

11.5 example_VectorConversion.cc

Example of how to convert from another vector class to a SimpleVector. This example uses CLHEP::HepLorentzVector

```
1
2 // Matt.Dobbs@Cern.CH, Feb 2000
3 // Example of building an event and a particle data table from scratch
4 // This is meant to be of use for persons implementing HepMC inside a MC
5 // event generator
7 // To Compile: go to the HepMC directory and type:
8 // gmake examples/example_BuildEventFromScratch.exe
9 //
10
11 #include <iostream>
12
13 #include "VectorConversion.h"
14 #include "HepMC/GenEvent.h"
15 #include "CLHEP/Vector/LorentzVector.h"
16
17 // in this example we use the HepMC namespace, so that we do not have to
18 // precede all HepMC classes with HepMC::
19
20 // This example also shows how to use the CLHEP Lorentz vector with HepMC2
21
22 using namespace HepMC;
23 using namespace CLHEP;
24
25 int main() {
26     //
27     // In this example we will place the following event into HepMC "by hand"
28     //
29     //     name status pdg_id  parent Px       Py    Pz       Energy      Mass
30     //  1  !p+!    3   2212    0,0    0.000    0.000 7000.000 7000.000    0.938
31     //  2  !p+!    3   2212    0,0    0.000    0.000-7000.000 7000.000    0.938
32     //=============================================================================
33     //  3  !d!     3      1    1,1    0.750   -1.569   32.191   32.238    0.000
34     //  4  !u~!    3     -2    2,2   -3.047  -19.000  -54.629   57.920    0.000
35     //  5  !W-!    3    -24    1,2    1.517   -20.68  -20.605   85.925   80.799
36     //  6  !gamma! 1     22    1,2   -3.813    0.113   -1.833    4.233    0.000
37     //  7  !d!     1      1    5,5   -2.445   28.816    6.082   29.552    0.010
38     //  8  !u~!    1     -2    5,5    3.962  -49.498  -26.687   56.373    0.006
39
40     // now we build the graph, which will look like
41     //                        p7                          #
42     // p1                    /                            #
43     //   \v1__p3      p5---v4                             #
44     //         \_v3_/       \                             #
45     //         /    \        p8                          #
46     //    v2__p4     \                                    #
47     //   /            p6                                  #
48     // p2                                                 #
49     //                                                    #
50
51     // First create the event container, with Signal Process 20, event number 1
52     //
53     // Note that the HepLorentzVectors will be automatically converted to
54     // HepMC::FourVector within GenParticle and GenVertex
55     GenEvent* evt = new GenEvent( 20, 1 );
56     // define the units
57     evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
58     //
59     // create vertex 1 and vertex 2, together with their inparticles
60     GenVertex* v1 = new GenVertex();
61     evt->add_vertex( v1 );
62     v1->add_particle_in( new GenParticle( HepLorentzVector(0,0,7000,7000),
```

```
63                                      2212, 3 ) );
64     GenVertex* v2 = new GenVertex();
65     evt->add_vertex( v2 );
66     v2->add_particle_in( new GenParticle( HepLorentzVector(0,0,-7000,7000),
67                                      2212, 3 ) );
68     //
69     // create the outgoing particles of v1 and v2
70     GenParticle* p3 =
71         new GenParticle( HepLorentzVector(.750,-1.569,32.191,32.238), 1, 3 );
72     v1->add_particle_out( p3 );
73     GenParticle* p4 =
74         new GenParticle( HepLorentzVector(-3.047,-19.,-54.629,57.920), -2, 3 );
75     v2->add_particle_out( p4 );
76     //
77     // create v3
78     GenVertex* v3 = new GenVertex();
79     evt->add_vertex( v3 );
80     v3->add_particle_in( p3 );
81     v3->add_particle_in( p4 );
82     v3->add_particle_out(
83         new GenParticle( HepLorentzVector(-3.813,0.113,-1.833,4.233 ), 22, 1 )
84         );
85     GenParticle* p5 =
86         new GenParticle( HepLorentzVector(1.517,-20.68,-20.605,85.925), -24,3);
87     v3->add_particle_out( p5 );
88     //
89     // create v4
90     GenVertex* v4 = new GenVertex(HepLorentzVector(0.12,-0.3,0.05,0.004));
91     evt->add_vertex( v4 );
92     v4->add_particle_in( p5 );
93     v4->add_particle_out(
94         new GenParticle( HepLorentzVector(-2.445,28.816,6.082,29.552), 1,1 )
95         );
96     v4->add_particle_out(
97         new GenParticle( HepLorentzVector(3.962,-49.498,-26.687,56.373), -2,1 )
98         );
99     //
100     // tell the event which vertex is the signal process vertex
101     evt->set_signal_process_vertex( v3 );
102     // the event is complete, we now print it out to the screen
103     evt->print();
104
105     // example conversion back to Lorentz vector
106     // add all outgoing momenta
107     std::cout << std::endl;
108     std::cout << " Add output momenta " << std::endl;
109     HepLorentzVector sum;
110     for ( GenEvent::particle_const_iterator p = evt->particles_begin();
111               p != evt->particles_end(); ++p ){
112         if( (*p)->status() == 1 ) {
113             sum += convertTo( (*p)->momentum() );
114             (*p)->print();
115         }
116     }
117     std::cout << "Vector Sum: " << sum << std::endl;
118
119     // now clean-up by deleteing all objects from memory
120     //
121     // deleting the event deletes all contained vertices, and all particles
122     // contained in those vertices
123     delete evt;
124
125     return 0;
126 }
```

11.6 fio/example_MyHerwig.cc

```
1
2 // Matt.Dobbs@Cern.CH, October 2002
3 // example of generating events with Herwig using HepMC/HerwigWrapper.h
4 // Events are read into the HepMC event record from the FORTRAN HEPEVT
5 // common block using the IO_HERWIG strategy.
16
17 #include <iostream>
18 #include "HepMC/HerwigWrapper.h"
19 #include "HepMC/IO_HERWIG.h"
20 #include "HepMC/IO_GenEvent.h"
21 #include "HepMC/GenEvent.h"
22 #include "HepMC/HEPEVT_Wrapper.h"
23
24 int main() {
25     //
26     //.....................................HEPEVT
27     // Herwig 6.4 uses HEPEVT with 4000 entries and 8-byte floating point
28     //  numbers. We need to explicitly pass this information to the
29     //  HEPEVT_Wrapper.
30     //
31     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
32     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
33     //
34     //.................................INITIALIZATIONS
35
36     hwproc.PBEAM1 = 7000.; // energy of beam1
37     hwproc.PBEAM2 = 7000.; // energy of beam2
38     // 1610 = gg->H--> WW, 1706 = qq-->ttbar, 2510 = ttH -> ttWW
39     hwproc.IPROC = 1706; // qq -> ttbar production
40     hwproc.MAXEV = 100; // number of events
41     // tell it what the beam particles are:
42     for ( unsigned int i = 0; i < 8; ++i ) {
43         hwbmch.PART1[i] = (i < 1) ? 'P' : ' ';
44         hwbmch.PART2[i] = (i < 1) ? 'P' : ' ';
45     }
46     hwigin();    // INITIALISE OTHER COMMON BLOCKS
47     hwevnt.MAXPR = 1; // number of events to print
48     hwuinc(); // compute parameter-dependent constants
49     hweini(); // initialise elementary process
50
51     //.....................................HepMC INITIALIZATIONS
52     //
53     // Instantiate an IO strategy for reading from HEPEVT.
54     HepMC::IO_HERWIG hepevtio;
55     // Instantiate an IO strategy to write the data to file
56     HepMC::IO_GenEvent ascii_io("example_MyHerwig.dat",std::ios::out);
57     //
58     //.....................................EVENT LOOP
59     for ( int i = 1; i <= hwproc.MAXEV; i++ ) {
60         if ( i%50==1 ) std::cout << "Processing Event Number "
61                                  << i << std::endl;
62         // initialise event
63         hwuine();
64         // generate hard subprocess
65         hwepro();
66         // generate parton cascades
67         hwbgen();
68         // do heavy object decays
69         hwdhob();
70         // do cluster formation
71         hwcfor();
72         // do cluster decays
73         hwcdec();
74         // do unstable particle decays
75         hwdhad();
```

```
76         // do heavy flavour hadron decays
77         hwdhvy();
78         // add soft underlying event if needed
79         hwmevt();
80         // finish event
81         hwufne();
82         HepMC::GenEvent* evt = hepevtio.read_next_event();
83         // define the units (Herwig uses GeV and mm)
84         evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
85         // set cross section information
86         evt->set_cross_section( HepMC::getHerwigCrossSection(i) );
87         // add some information to the event
88         evt->set_event_number(i);
89         evt->set_signal_process_id(20);
90         if (i<=hwevnt.MAXPR) {
91             std::cout << "\n\n This is the FIXED version of HEPEVT as "
92                       << "coded in IO_HERWIG " << std::endl;
93             HepMC::HEPEVT_Wrapper::print_hepevt();
94             evt->print();
95         }
96         // write the event to the ascii file
97         ascii_io << evt;
98
99         // we also need to delete the created event from memory
100         delete evt;
101     }
102     //.....................................TERMINATION
103     hwefin();
104
105     return 0;
106 }
```

11.7 fio/example_MyPythia.cc

example to generate events and write output example to generate events and perform simple event selection example to read the file written by pythia_out example to generate events, write them, and read them back

```
1
2 // Matt.Dobbs@Cern.CH, December 1999
3 // November 2000, updated to use Pythia 6.1
4 //
46
47
48 #include <iostream>
49 #include "HepMC/PythiaWrapper.h"
50 #include "HepMC/IO_HEPEVT.h"
51 #include "HepMC/IO_GenEvent.h"
52 #include "HepMC/IO_AsciiParticles.h"
53 #include "HepMC/GenEvent.h"
54 #include "PythiaHelper.h"
55
57
61 class IsGoodEventMyPythia {
62 public:
64     bool operator()( const HepMC::GenEvent* evt ) {
65         for ( HepMC::GenEvent::particle_const_iterator p
66                   = evt->particles_begin(); p != evt->particles_end(); ++p ){
67             if ( (*p)->pdg_id() == 22 && (*p)->momentum().perp() > 25. ) {
68                 //std::cout << "Event " << evt->event_number()
69                 //     << " is a good event." << std::endl;
70                 //(*p)->print();
71                 return 1;
72             }
73         }
74         return 0;
75     }
76 };
77
78
79 void pythia_out();
80 void pythia_in();
81 void pythia_in_out();
82 void event_selection();
83 void pythia_particle_out();
84
85 int main() {
86     // example to generate events and write output
87     pythia_out();
88     // example to generate events and perform simple event selection
89     event_selection();
90     // example to read the file written by pythia_out
91     pythia_in();
92     // example to generate events, write them, and read them back
93     pythia_in_out();
94
95     return 0;
96 }
97
98
99 void pythia_out()
100 {
101     std::cout << std::endl;
102     std::cout << "Begin pythia_out()" << std::endl;
103    //.....................................HEPEVT
104     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
105     //  numbers. We need to explicitly pass this information to the
106     //  HEPEVT_Wrapper.
```

```
107     //
108     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
109     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
110     //
111     //.....................................PYTHIA INITIALIZATIONS
112     initPythia();
113
114     //.....................................HepMC INITIALIZATIONS
115     //
116     // Instantiate an IO strategy for reading from HEPEVT.
117     HepMC::IO_HEPEVT hepevtio;
118     //
119     { // begin scope of ascii_io
120         // Instantiate an IO strategy to write the data to file
121         HepMC::IO_GenEvent ascii_io("example_MyPythia.dat",std::ios::out);
122         //
123         //.....................................EVENT LOOP
124         for ( int i = 1; i <= 100; i++ ) {
125             if ( i%50==1 ) std::cout << "Processing Event Number "
126                                      << i << std::endl;
127             call_pyevnt();      // generate one event with Pythia
128             // pythia pyhepc routine converts common PYJETS in common HEPEVT
129             call_pyhepc( 1 );
130             HepMC::GenEvent* evt = hepevtio.read_next_event();
131             // define the units (Pythia uses GeV and mm)
132             evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
133             // add some information to the event
134             evt->set_event_number(i);
135             evt->set_signal_process_id(20);
136             // set number of multi parton interactions
137             evt->set_mpi( pypars.msti[31-1] );
138             // set cross section information
139             evt->set_cross_section( HepMC::getPythiaCrossSection() );
140             // write the event out to the ascii files
141             ascii_io << evt;
142             // we also need to delete the created event from memory
143             delete evt;
144         }
145         //.....................................TERMINATION
146         // write out some information from Pythia to the screen
147         call_pystat( 1 );
148     } // end scope of ascii_io
149 }
150
151
152 void event_selection()
153 {
154     std::cout << std::endl;
155     std::cout << "Begin event_selection()" << std::endl;
156     //.....................................HEPEVT
157     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
158     //  numbers. We need to explicitly pass this information to the
159     //  HEPEVT_Wrapper.
160     //
161     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
162     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
163     //
164     //.....................................PYTHIA INITIALIZATIONS
165     initPythia();
166     //
167     //.....................................HepMC INITIALIZATIONS
168     // Instantiate an IO strategy for reading from HEPEVT.
169     HepMC::IO_HEPEVT hepevtio;
170     // declare an instance of the event selection predicate
171     IsGoodEventMyPythia is_good_event;
172     //.....................................EVENT LOOP
173     int icount=0;
```

```
174     int num_good_events=0;
175     for ( int i = 1; i <= 100; i++ ) {
176         icount++;
177         if ( i%50==1 ) std::cout << "Processing Event Number "
178                                  << i << std::endl;
179         call_pyevnt(); // generate one event with Pythia
180         // pythia pyhepc routine convert common PYJETS in common HEPEVT
181         call_pyhepc( 1 );
182         HepMC::GenEvent* evt = hepevtio.read_next_event();
183         // define the units (Pythia uses GeV and mm)
184         evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
185         // set number of multi parton interactions
186         evt->set_mpi( pypars.msti[31-1] );
187         // set cross section information
188         evt->set_cross_section( HepMC::getPythiaCrossSection() );
189         // do event selection
190         if ( is_good_event(evt) ) {
191             std::cout << "Good Event Number " << i << std::endl;
192             ++num_good_events;
193         }
194         // we also need to delete the created event from memory
195         delete evt;
196     }
197     //.....................................TERMINATION
198     // write out some information from Pythia to the screen
199     call_pystat( 1 );
200     //.....................................PRINT RESULTS
201     std::cout << num_good_events << " out of " << icount
202               << " processed events passed the cuts. Finished." << std::endl;
203 }
204
205 void pythia_in()
206 {
207     std::cout << std::endl;
208     std::cout << "Begin pythia_in()" << std::endl;
209     std::cout << "reading example_MyPythia.dat" << std::endl;
210     //.....................................define an input scope
211     {
212         // open input stream
213         std::ifstream istr( "example_MyPythia.dat" );
214         if( !istr ) {
215           std::cerr << "example_ReadMyPythia: cannot open example_MyPythia.dat" << std::endl;
216           exit(-1);
217         }
218         HepMC::IO_GenEvent ascii_in(istr);
219         // open output stream (alternate method)
220         HepMC::IO_GenEvent ascii_out("example_MyPythia2.dat",std::ios::out);
221         // now read the file
222         int icount=0;
223         HepMC::GenEvent* evt = ascii_in.read_next_event();
224         while ( evt ) {
225             icount++;
226             if ( icount%50==1 ) std::cout << "Processing Event Number " << icount
227                                           << " its # " << evt->event_number()
228                                           << std::endl;
229             // write the event out to the ascii file
230             ascii_out << evt;
231             delete evt;
232             ascii_in >> evt;
233         }
234         //.....................................PRINT RESULT
235         std::cout << icount << " events found. Finished." << std::endl;
236     } // ascii_out and istr destructors are called here
237 }
238
239 void pythia_in_out()
240 {
```

```
241     std::cout << std::endl;
242     std::cout << "Begin pythia_in_out()" << std::endl;
243     //.....................................HEPEVT
244     // Pythia 6.3 uses HEPEVT with 4000 entries and 8-byte floating point
245     //  numbers. We need to explicitly pass this information to the
246     //  HEPEVT_Wrapper.
247     //
248     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
249     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
250     //
251     //.....................................PYTHIA INITIALIZATIONS
252     initPythia();
253
254     //.....................................HepMC INITIALIZATIONS
255     //
256     // Instantiate an IO strategy for reading from HEPEVT.
257     HepMC::IO_HEPEVT hepevtio;
258     //
259     //.....................................define the output scope
260     {
261         // Instantial an IO strategy to write the data to file
262         HepMC::IO_GenEvent ascii_io("example_MyPythiaRead.dat",std::ios::out);
263         //
264         //.....................................EVENT LOOP
265         for ( int i = 1; i <= 100; i++ ) {
266             if ( i%50==1 ) std::cout << "Processing Event Number "
267                                      << i << std::endl;
268             call_pyevnt();      // generate one event with Pythia
269             // pythia pyhepc routine converts common PYJETS in common HEPEVT
270             call_pyhepc( 1 );
271             HepMC::GenEvent* evt = hepevtio.read_next_event();
272             // define the units (Pythia uses GeV and mm)
273             evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
274             // set cross section information
275             evt->set_cross_section( HepMC::getPythiaCrossSection() );
276             // add some information to the event
277             evt->set_event_number(i);
278             evt->set_signal_process_id(20);
279             // write the event out to the ascii file
280             ascii_io << evt;
281             // we also need to delete the created event from memory
282             delete evt;
283         }
284         //.....................................TERMINATION
285         // write out some information from Pythia to the screen
286         call_pystat( 1 );
287     }  // ascii_io destructor is called here
288     //
289     //.....................................define an input scope
290     {
291         // now read the file we wrote
292         HepMC::IO_GenEvent ascii_in("example_MyPythiaRead.dat",std::ios::in);
293         HepMC::IO_GenEvent ascii_io2("example_MyPythiaRead2.dat",std::ios::out);
294         int icount=0;
295         HepMC::GenEvent* evt = ascii_in.read_next_event();
296         while ( evt ) {
297             icount++;
298             if ( icount%50==1 ) std::cout << "Processing Event Number " << icount
299                                           << " its # " << evt->event_number()
300                                           << std::endl;
301             // write the event out to the ascii file
302             ascii_io2 << evt;
303             delete evt;
304             ascii_in >> evt;
305         }
306         //.....................................PRINT RESULT
307         std::cout << icount << " events found. Finished." << std::endl;
```

```
308     } // ascii_io2 and ascii_in destructors are called here
309 }
310
311 void pythia_particle_out()
312 {
313     std::cout << std::endl;
314     std::cout << "Begin pythia_particle_out()" << std::endl;
315     //.....................................HEPEVT
316     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
317     //  numbers. We need to explicitly pass this information to the
318     //  HEPEVT_Wrapper.
319     //
320     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
321     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
322     //
323     //.....................................PYTHIA INITIALIZATIONS
324     initPythia();
325
326     //.....................................HepMC INITIALIZATIONS
327     //
328     // Instantiate an IO strategy for reading from HEPEVT.
329     HepMC::IO_HEPEVT hepevtio;
330     //
331     { // begin scope of ascii_io
332         // Instantiate an IO strategy to write the data to file
333         HepMC::IO_AsciiParticles ascii_io("example_PythiaParticle.dat",std::ios::out);
334         //
335         //.....................................EVENT LOOP
336         for ( int i = 1; i <= 100; i++ ) {
337             if ( i%50==1 ) std::cout << "Processing Event Number "
338                                      << i << std::endl;
339             call_pyevnt();      // generate one event with Pythia
340             // pythia pyhepc routine converts common PYJETS in common HEPEVT
341             call_pyhepc( 1 );
342             HepMC::GenEvent* evt = hepevtio.read_next_event();
343             // define the units (Pythia uses GeV and mm)
344             evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
345             // set cross section information
346             evt->set_cross_section( HepMC::getPythiaCrossSection() );
347             // add some information to the event
348             evt->set_event_number(i);
349             evt->set_signal_process_id(20);
350             // write the event out to the ascii file
351             ascii_io << evt;
352             // we also need to delete the created event from memory
353             delete evt;
354         }
355         //.....................................TERMINATION
356         // write out some information from Pythia to the screen
357         call_pystat( 1 );
358     } // end scope of ascii_io
359 }
360
```

11.8 fio/example_PythiaStreamIO.cc

This example generates Pythia events and fills cross section information from pyint5. The example uses streaming I/O to write a file and then read it.

```
1
2 // example_PythiaStreamIO.cc
3 //
4 // garren@fnal.gov, May 2009
5 //
19
20
21 #include <fstream>
22 #include <iostream>
23 #include "HepMC/PythiaWrapper.h"
24 #include "HepMC/IO_HEPEVT.h"
25 #include "HepMC/GenEvent.h"
26 #include "PythiaHelper.h"
27
28 void writePythiaStreamIO();
29 void readPythiaStreamIO();
30
31 int main() {
32
33     writePythiaStreamIO();
34     readPythiaStreamIO();
35
36     return 0;
37 }
38
39
40 void writePythiaStreamIO() {
41     // example to generate events and write output
42     std::cout << std::endl;
43     std::cout << "Begin pythia_out()" << std::endl;
44    //.....................................HEPEVT
45     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
46     //  numbers. We need to explicitly pass this information to the
47     //  HEPEVT_Wrapper.
48     //
49     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
50     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
51     //
52     //.....................................PYTHIA INITIALIZATIONS
53     initPythia();
54
55     //.....................................HepMC INITIALIZATIONS
56     //
57     // Instantiate an IO strategy for reading from HEPEVT.
58     HepMC::IO_HEPEVT hepevtio;
59     //
60     { // begin scope of ascii_io
61         // declare an output stream
62         const char outfile[] = "example_PythiaStreamIO_write.dat";
63         std::ofstream ascii_io( outfile );
64         if( !ascii_io ) {
65           std::cerr << "cannot open " << outfile << std::endl;
66           exit(-1);
67         }
68         // use the default IO_GenEvent precision
69         ascii_io.precision(16);
70         // write the line that defines the beginning of a GenEvent block
71         HepMC::write_HepMC_IO_block_begin( ascii_io );
72         //
73         //.....................................EVENT LOOP
```

```
74         for ( int i = 1; i <= 100; i++ ) {
75             if ( i%50==1 ) std::cout << "Processing Event Number "
76                                      << i << std::endl;
77             call_pyevnt();      // generate one event with Pythia
78             // pythia pyhepc routine converts common PYJETS in common HEPEVT
79             call_pyhepc( 1 );
80             HepMC::GenEvent* evt = hepevtio.read_next_event();
81             // define the units (Pythia uses GeV and mm)
82             evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
83             // add some information to the event
84             evt->set_event_number(i);
85             evt->set_signal_process_id(20);
86             // set number of multi parton interactions
87             evt->set_mpi( pypars.msti[31-1] );
88             // set cross section information
89             evt->set_cross_section( HepMC::getPythiaCrossSection() );
90             // write the event out to the ascii files
91             ascii_io << (*evt);;
92             // we also need to delete the created event from memory
93             delete evt;
94         }
95         // write the line that defines the end of a GenEvent block
96         HepMC::write_HepMC_IO_block_end( ascii_io );
97         //.....................................TERMINATION
98         // write out some information from Pythia to the screen
99         call_pystat( 1 );
100    } // end scope of ascii_io
101 }
102
103 void readPythiaStreamIO() {
104    // example to read events written by writePythiaStreamIO
105    // and write them back out
106    std::cout << std::endl;
107    // input units are GeV and mm
108    const char infile[] = "example_PythiaStreamIO_write.dat";
109    std::ifstream is( infile );
110    if( !is ) {
111      std::cerr << "cannot open " << infile << std::endl;
112      exit(-1);
113    }
114    //
115    { // begin scope of ascii_io
116        // declare an output stream
117        const char outfile[] = "example_PythiaStreamIO_read.dat";
118        std::ofstream ascii_io( outfile );
119        if( !ascii_io ) {
120          std::cerr << "cannot open " << outfile << std::endl;
121          exit(-1);
122        }
123        ascii_io.precision(16);
124        HepMC::write_HepMC_IO_block_begin( ascii_io );
125        //
126        //.....................................EVENT LOOP
127        HepMC::GenEvent evt;
128        int i = 0;
129        while ( is ) {
130            evt.read( is );
131            // make sure we have a valid event
132            if( evt.is_valid() ) {
133                ++i;
134                if ( i%50==1 ) std::cout << "Processing Event Number "
135                                         << i << std::endl;
136                if ( i%25==2 ) {
137                    // write the cross section if it exists
138                    if( evt.cross_section() ) {
139                        std::cout << "cross section at event " << i << " is "
140                                  << evt.cross_section()->cross_section()
```

```
141                                     << std::endl;
142                     }
143                 }
144                 // write the event out to the ascii files
145                 evt.write( ascii_io );
146             }
147         }
148         //.......................................TERMINATION
149         HepMC::write_HepMC_IO_block_end( ascii_io );
150     } // end scope of ascii_io
151 }
```

11.9 fio/testHerwigCopies.cc

Multiple events in memory at the same time

```
1
2 // testHerwigCopies.cc
3 //
4 // garren@fnal.gov, January 2008
5 // Multiple events in memory at the same time
7
8 #include <fstream>
9 #include <iostream>
10 #include "HepMC/HerwigWrapper.h"
11 #include "HepMC/IO_HERWIG.h"
12 #include "HepMC/GenEvent.h"
13 #include "HepMC/CompareGenEvent.h"
14 #include "HepMC/HEPEVT_Wrapper.h"
15
16 int main() {
17     //
18     //.....................................HEPEVT
19     // Herwig 6.4 uses HEPEVT with 4000 entries and 8-byte floating point
20     //  numbers. We need to explicitly pass this information to the
21     //  HEPEVT_Wrapper.
22     //
23     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
24     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
25     //
26     //.....................................INITIALIZATIONS
27
28     hwproc.PBEAM1 = 7000.; // energy of beam1
29     hwproc.PBEAM2 = 7000.; // energy of beam2
30     // 1610 = gg->H--> WW, 1706 = qq-->ttbar, 2510 = ttH -> ttWW
31     hwproc.IPROC = 1706; // qq -> ttbar production
32     hwproc.MAXEV = 50; // number of events
33     // tell it what the beam particles are:
34     for ( unsigned int i = 0; i < 8; ++i ) {
35         hwbmch.PART1[i] = (i < 1) ? 'P' : ' ';
36         hwbmch.PART2[i] = (i < 1) ? 'P' : ' ';
37     }
38     hwigin();    // INITIALISE OTHER COMMON BLOCKS
39     hwevnt.MAXPR = 0; // number of events to print
40     hwuinc(); // compute parameter-dependent constants
41     hweini(); // initialise elementary process
42
43     //.....................................HepMC INITIALIZATIONS
44     //
45     // Instantiate an IO strategy for reading from HEPEVT.
46     HepMC::IO_HERWIG hepevtio;
47     //
48     // open some output files
49     std::ofstream out1( "testHerwigOriginals.dat" );
50     std::ofstream out2( "testHerwigCopies1.dat" );
51     std::ofstream out3( "testHerwigCopies2.dat" );
52     //
53     //.....................................EVENT LOOP
54     for ( int i = 1; i <= hwproc.MAXEV; i++ ) {
55         if ( i%50==1 ) std::cout << "Processing Event Number "
56                                  << i << std::endl;
57         // initialise event
58         hwuine();
59         // generate hard subprocess
60         hwepro();
61         // generate parton cascades
62         hwbgen();
63         // do heavy object decays
```

```
64          hwdhob();
65          // do cluster formation
66          hwcfor();
67          // do cluster decays
68          hwcdec();
69          // do unstable particle decays
70          hwdhad();
71          // do heavy flavour hadron decays
72          hwdhvy();
73          // add soft underlying event if needed
74          hwmevt();
75          // finish event
76          hwufne();
77          HepMC::GenEvent* evt = hepevtio.read_next_event();
78          // herwig uses GeV and mm
79          evt->use_units( HepMC::Units::GEV, HepMC::Units::MM);
80          // set cross section information
81          evt->set_cross_section( HepMC::getHerwigCrossSection(i) );
82          // add some information to the event
83          evt->set_event_number(i);
84          evt->set_signal_process_id(20);
85          //
86          //.....................make some copies
87          evt->print(out1);
88          HepMC::GenEvent ec = (*evt);
89          ec.print(out2);
90          HepMC::GenEvent* evt4 = new HepMC::GenEvent(*evt);
91          evt4->print(out3);
92          if( !compareGenEvent(evt,evt4) ) {
93             std::cerr << "testHerwigCopies: GenEvent comparison fails at event "
94                       << evt->event_number() << std::endl;
95             return -1;
96          }
97
98          // we also need to delete the created event from memory
99          delete evt;
100          delete evt4;
101      }
102      //....................................TERMINATION
103      hwefin();
104      std::cout << "testHerwigCopies: event comparison is successful" << std::endl;
105
106      return 0;
107  }
```

11.10 fio/testPythiaCopies.cc

Multiple events in memory at the same time

```
1
2 // testPythiaCopies.cc
3 //
4 // garren@fnal.gov, January 2008
5 // Multiple events in memory at the same time
7
8 #include <fstream>
9 #include <iostream>
10 #include "HepMC/PythiaWrapper.h"
11 #include "HepMC/IO_HEPEVT.h"
12 #include "HepMC/GenEvent.h"
13 #include "HepMC/CompareGenEvent.h"
14 #include "PythiaHelper.h"
15
16 int main() {
17     //
18     //.....................................HEPEVT
19     // Pythia 6.1 uses HEPEVT with 4000 entries and 8-byte floating point
20     //  numbers. We need to explicitly pass this information to the
21     //  HEPEVT_Wrapper.
22     //
23     HepMC::HEPEVT_Wrapper::set_max_number_entries(4000);
24     HepMC::HEPEVT_Wrapper::set_sizeof_real(8);
25     //
26     //.....................................PYTHIA INITIALIZATIONS
27     initPythia();
28     //
29     //.....................................HepMC INITIALIZATIONS
30     //
31     // Instantiate an IO strategy for reading from HEPEVT.
32     HepMC::IO_HEPEVT hepevtio;
33     //
34     // open some output files
35     std::ofstream out1( "testPythiaOriginals.dat" );
36     std::ofstream out2( "testPythiaCopies1.dat" );
37     std::ofstream out3( "testPythiaCopies2.dat" );
38     //
39     //.....................................EVENT LOOP
40     for ( int i = 1; i <= 50; i++ ) {
41         if ( i%50==1 ) std::cout << "Processing Event Number "
42                                  << i << std::endl;
43         call_pyevnt();      // generate one event with Pythia
44         // pythia pyhepc routine convert common PYJETS in common HEPEVT
45         call_pyhepc( 1 );
46         HepMC::GenEvent* evt = hepevtio.read_next_event();
47         // pythia uses GeV and mm
48         evt->use_units( HepMC::Units::GEV, HepMC::Units::MM);
49         // set a couple of arbitrary weights
50         evt->weights().push_back(0.456);
51         evt->weights()["test2"] = 0.8956;
52         // set number of multi parton interactions
53         evt->set_mpi( pypars.msti[31-1] );
54         // set cross section information
55         evt->set_cross_section( HepMC::getPythiaCrossSection() );
56         //
57         //.....................make some copies
58         evt->print(out1);
59         HepMC::GenEvent ec = (*evt);
60         ec.print(out2);
61         HepMC::GenEvent* evt4 = new HepMC::GenEvent(*evt);
62         evt4->print(out3);
63         if( !compareGenEvent(evt,evt4) ) {
```

```
64            std::cerr << "testPythiaCopies: GenEvent comparison fails at event "
65                      << evt->event_number() << std::endl;
66            return -1;
67         }
68         //
69         // now delete the created events from memory
70         delete evt;
71         delete evt4;
72     }
73     //.....................................TERMINATION
74     // write out some information from Pythia to the screen
75     call_pystat( 1 );
76     std::cout << "testPythiaCopies: event comparison is successful" << std::endl;
77
78     return 0;
79 }
80
81
82
```

11.11 testFlow.cc

Use a modified example_BuildEventFromScratch to test Flow

```
1
2 // testFlow.cc
3 //
4 // garren@fnal.gov, June 2009
5 // based on example_BuildEventFromScratch.cc
7
8 #include <iostream>
9 #include <fstream>
10 #include <vector>
11
12 #include "HepMC/GenEvent.h"
13 #include "HepMC/IO_GenEvent.h"
14
15 typedef std::vector<HepMC::GenParticle*> FlowVec;
16
17 int main() {
18     //
19     // In this example we will place the following event into HepMC "by hand"
20     //
21     //     name status pdg_id  parent Px       Py    Pz       Energy      Mass
22     //  1  !p+!    3   2212    0,0    0.000    0.000 7000.000 7000.000    0.938
23     //  2  !p+!    3   2212    0,0    0.000    0.000-7000.000 7000.000    0.938
24     //=============================================================================
25     //  3  !d!     3      1    1,1    0.750   -1.569   32.191   32.238    0.000
26     //  4  !u~!    3     -2    2,2   -3.047  -19.000  -54.629   57.920    0.000
27     //  5  !W-!    3    -24    1,2    1.517   -20.68  -20.605   85.925   80.799
28     //  6  !gamma! 1     22    1,2   -3.813    0.113   -1.833    4.233    0.000
29     //  7  !d!     1      1    5,5   -2.445   28.816    6.082   29.552    0.010
30     //  8  !u~!    1     -2    5,5    3.962  -49.498  -26.687   56.373    0.006
31
32     // open an output file
33     const char outfile[] = "testFlow.out";
34     std::ofstream os( outfile );
35     if( !os ) {
36       std::cerr << "cannot open " << outfile << std::endl;
37       exit(-1);
38     }
39     // declare several IO_GenEvent instances for comparison
40     HepMC::IO_GenEvent xout1("testFlow.out1",std::ios::out);
41     HepMC::IO_GenEvent xout2("testFlow.out2",std::ios::out);
42     HepMC::IO_GenEvent xout3("testFlow.out3",std::ios::out);
43     // output streams for copy test
44     std::ofstream xout4( "testFlow.out4" );
45     std::ofstream xout5( "testFlow.out5" );
46
47     int numbad = 0;
48
49
50     // build the graph, which will look like
51     //                        p7                  #
52     // p1                    /                    #
53     //   \v1__p3      p5---v4                     #
54     //         \_v3_/       \                     #
55     //         /    \        p8                  #
56     //    v2__p4     \                           #
57     //   /            p6                         #
58     // p2                                        #
59     //
60     // define a flow pattern as  p1 -> p3 -> p6
61     //                       and p2 -> p4 -> p5
62     //
63
```

```
64     // First create the event container, with Signal Process 20, event number 1
65     //
66     HepMC::GenEvent* evt = new HepMC::GenEvent( 20, 1 );
67     evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
68     //
69     // create vertex 1 and vertex 2, together with their inparticles
70     HepMC::GenVertex* v1 = new HepMC::GenVertex();
71     evt->add_vertex( v1 );
72     HepMC::GenParticle* p1 = new HepMC::GenParticle( HepMC::FourVector(0,0,7000,7000),
73                                       2212, 3 );
74     p1->set_flow(1,231);
75     v1->add_particle_in( p1 );
76     HepMC::GenVertex* v2 = new HepMC::GenVertex();
77     evt->add_vertex( v2 );
78     HepMC::GenParticle* p2 = new HepMC::GenParticle( HepMC::FourVector(0,0,-7000,7000),
79                                       2212, 3 );
80     p2->set_flow(1,243);
81     v2->add_particle_in( p2 );
82     //
83     // create the outgoing particles of v1 and v2
84     HepMC::GenParticle* p3 =
85         new HepMC::GenParticle( HepMC::FourVector(.750,-1.569,32.191,32.238),
86                                 1, 3 );
87     p3->set_flow(1,231);
88     v1->add_particle_out( p3 );
89     HepMC::GenParticle* p4 =
90         new HepMC::GenParticle( HepMC::FourVector(-3.047,-19.,-54.629,57.920),
91                                 -2, 3 );
92     p4->set_flow(1,243);
93     v2->add_particle_out( p4 );
94     //
95     // create v3
96     HepMC::GenVertex* v3 = new HepMC::GenVertex();
97     evt->add_vertex( v3 );
98     v3->add_particle_in( p3 );
99     v3->add_particle_in( p4 );
100     HepMC::GenParticle* p6 =
101           new HepMC::GenParticle( HepMC::FourVector(-3.813,0.113,-1.833,4.233 ),
102                                   22, 1 );
103    p6->set_flow(1,231);
104    v3->add_particle_out( p6 );
105    HepMC::GenParticle* p5 =
106        new HepMC::GenParticle( HepMC::FourVector(1.517,-20.68,-20.605,85.925),
107                                -24, 3 );
108    p5->set_flow(1,243);
109    v3->add_particle_out( p5 );
110    //
111    // create v4
112    HepMC::GenVertex* v4 = new HepMC::GenVertex(HepMC::FourVector(0.12,-0.3,0.05,0.004));
113    evt->add_vertex( v4 );
114    v4->add_particle_in( p5 );
115    HepMC::GenParticle* p7 = new HepMC::GenParticle( HepMC::FourVector(-2.445,28.816,6.082,29.552), 1,
116    v4->add_particle_out( p7 );
117    HepMC::GenParticle* p8 = new HepMC::GenParticle( HepMC::FourVector(3.962,-49.498,-26.687,56.373),
118    v4->add_particle_out( p8 );
119    //
120    // tell the event which vertex is the signal process vertex
121    evt->set_signal_process_vertex( v3 );
122    // the event is complete, we now print it out
123    evt->print( os );
124
125    // look at the flow we created
126    os << std::endl;
127    FlowVec result1 = p1->flow().dangling_connected_partners( p1->flow().icode(1) );
128    FlowVec result2 = p1->flow().connected_partners( p1->flow().icode(1) );
129    FlowVec::iterator it;
130    os << "dangling partners of particle " << p1->barcode() << std::endl;
```

```
131     for( it = result1.begin(); it != result1.end(); ++it ) {
132       os << (*it)->barcode() << " " ;
133       os.width(8);
134       os << (*it)->pdg_id() << " " << (*it)->flow(1)  << std::endl;
135     }
136     os << "all partners of particle " << p1->barcode() << std::endl;
137     for( it = result2.begin(); it != result2.end(); ++it ) {
138       os << (*it)->barcode() << " " ;
139       os.width(8);
140       os << (*it)->pdg_id() << " " << (*it)->flow(1)  << std::endl;
141     }
142     FlowVec result3 = p2->flow().dangling_connected_partners( p2->flow().icode(1) );
143     FlowVec result4 = p2->flow().connected_partners( p2->flow().icode(1) );
144     os << "dangling partners of particle " << p2->barcode() << std::endl;
145     for( it = result3.begin(); it != result3.end(); ++it ) {
146       os << (*it)->barcode() << " " ;
147       os.width(8);
148       os << (*it)->pdg_id() << " " << (*it)->flow(1)  << std::endl;
149     }
150     os << "all partners of particle " << p2->barcode() << std::endl;
151     for( it = result4.begin(); it != result4.end(); ++it ) {
152       os << (*it)->barcode() << " " ;
153       os.width(8);
154       os << (*it)->pdg_id() << " " << (*it)->flow(1)  << std::endl;
155     }
156     // write event
157     xout1 << evt;
158     // testing bug #73987 - flow not copied
159     // call the write method directly
160     evt->write(xout4);
161     // make a copy and write it
162     HepMC::GenEvent(*evt).write(xout5);
163
164     // try changing and erasing flow
165     p2->set_flow(2,345);
166             xout2 << evt;
167     FlowVec result5 = p2->flow().connected_partners( p2->flow().icode(1) );
168     if ( result4 != result5 ) {
169         std::cerr << "ERROR: list of partners has changed after adding flow" << std::endl;
170         ++numbad;
171     }
172     // the flow method returns a copy,
173     // so we must set the flow again to change it
174     HepMC::Flow f2 = p2->flow();
175     if( f2.erase(2) ) {
176         p2->set_flow( f2 );
177     } else {
178         std::cerr << "ERROR: first erase was NOT successful" << std::endl;
179         ++numbad;
180     }
181     f2 = p2->flow();
182     if( f2.erase(2) ) {
183         std::cerr << "ERROR: second erase was successful" << std::endl;
184     }
185             xout3 << evt;
186     FlowVec result6 = p2->flow().connected_partners( p2->flow().icode(1) );
187     if ( result4 != result6 ) {
188         std::cerr << "ERROR: list of partners has changed after removing flow" << std::endl;
189         ++numbad;
190     }
191
192     // now clean-up by deleteing all objects from memory
193     //
194     // deleting the event deletes all contained vertices, and all particles
195     // contained in those vertices
196     delete evt;
197
```

```
198     if( numbad > 0 ) std::cerr << numbad << " errors in testFlow" << std::endl;
199
200     return numbad;
201 }
```

11.12 testHepMC.cc.in

The **HepMC** (p. 25) tests can also serve as useful examples based on example_EventSelection Apply an event selection to the events in testHepMC.input Events containing a photon of pT > 25 GeV pass the selection and are written to "testHepMC.out" Add arbitrary PDF information to the good events Also write events using IO_AsciiParticles Test the new GenCrossSection class

```
1 //--------------------------------------------------------------------
2 // testHepMC.cc.in
3 //
4 // garren@fnal.gov, March 2006
5 // based on example_EventSelection
6 // Apply an event selection to the events in testHepMC.input
7 // Events containing a photon of pT > 25 GeV pass the selection
8 // and are written to "testHepMC.out"
9 // Also write events using IO_AsciiParticles
10 //--------------------------------------------------------------------
11 //
12
13 #include "HepMC/GenEvent.h"
14 #include "HepMC/GenCrossSection.h"
15 #ifndef HEPMC_IO_ASCII_REMOVED
16 #include "HepMC/IO_Ascii.h"
17 #endif
18 #ifdef HEPMC_HAS_IO_GENEVENT
19 #include "HepMC/IO_GenEvent.h"
20 #endif
21 #include "HepMC/IO_AsciiParticles.h"
22
23 // define methods and classes used by this test
24 #include "IsGoodEvent.h"
25 #include "testHepMCMethods.h"
26
27 void read_testIOGenEvent(std::ostream & os);
28 void read_testUnits(std::ostream & os);
29 void read_variousFormats(std::ostream & os);
30 void writeWithCrossSection(std::ostream & os);
31 void readWithCrossSection(std::ostream & os);
32 void writeWithWeight(std::ostream & os);
33 void readWithWeight(std::ostream & os);
34 void read_nan(std::ostream & os);
35
36 int main() {
37     std::ofstream os( "testHepMC.cout" );
38     std::ofstream osv( "testHepMCVarious.cout" );
39     read_testIOGenEvent(os);
40     read_testUnits(os);
41     read_variousFormats(osv);
42     read_nan(os);
43     writeWithCrossSection(os);
44     readWithCrossSection(os);
45     writeWithWeight(os);
46     readWithWeight(os);
47     return 0;
48 }
49
50 void read_testIOGenEvent(std::ostream & os)
51 {
52     os << std::endl;
53     os << "basic IO_GenEvent input and output" << std::endl;
54     // declare an input strategy to read the data produced with the
55     // example_MyPythia - units are GeV and mm
56     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
```

```
57     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
58     // declare another IO_GenEvent for writing out the good events
59     HepMC::IO_GenEvent ascii_out("testHepMC.out",std::ios::out);
60     // declare an output IO_GenEvent for testing precision
61     HepMC::IO_GenEvent prec_out("testHepMCprecision.out",std::ios::out);
62     prec_out.precision(10);
63     // declare an IO_AsciiParticle for output
64     HepMC::IO_AsciiParticles particle_out("testHepMCParticle.out",std::ios::out);
65     // declare an instance of the event selection predicate
66     IsGoodEvent is_good_event;
67     //.....................................EVENT LOOP
68     int icount=0;
69     int num_good_events=0;
70     HepMC::GenEvent* evt = ascii_in.read_next_event();
71     while ( evt ) {
72         ++icount;
73         if ( icount%50==1 ) os << "Processing Event Number " << icount
74                                    << " its # " << evt->event_number()
75                                    << std::endl;
76         if ( is_good_event(evt) ) {
77             particleTypes(evt,os);
78             // verify use_input_units()
79             evt->write_units(os);
80             double pim = findPiZero(evt);
81             os << " pizero mass: " << pim << std::endl;
82             //
83             ascii_out << evt;
84             particle_out << evt;
85             prec_out << evt;
86             ++num_good_events;
87         }
88
89         // clean up and get next event
90         delete evt;
91         ascii_in >> evt;
92     }
93     //.....................................PRINT RESULT
94     os << num_good_events << " out of " << icount
95                << " processed events passed the cuts. Finished." << std::endl;
96 }
97
98 void read_testUnits(std::ostream & os)
99 {
100     os << std::endl;
101     os << "IO_GenEvent input and output using define_units" << std::endl;
102     // declare an input strategy to read the data produced with the
103     // example_MyPythia - units are GeV and mm
104     // we DO NOT define input units here, instead we use define_units
105     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
106     // declare another IO_GenEvent for writing out the good events
107     HepMC::IO_GenEvent ascii_out("testDefineUnits.out",std::ios::out);
108     // declare an instance of the event selection predicate
109     IsGoodEvent is_good_event;
110     //.....................................EVENT LOOP
111     int icount=0;
112     int num_good_events=0;
113     HepMC::GenEvent* evt = ascii_in.read_next_event();
114     while ( evt ) {
115         ++icount;
116         evt->define_units( HepMC::Units::GEV, HepMC::Units::MM );
117         if ( icount%50==1 ) os << "Processing Event Number " << icount
118                                    << " its # " << evt->event_number()
119                                    << std::endl;
120         if ( is_good_event(evt) ) {
121             // verify define_units()
122             evt->write_units(os);
123             double pim = findPiZero(evt);
```

```
124             os << " pizero mass: " << pim << std::endl;
125             //
126             particleTypes(evt,os);
127             ascii_out << evt;
128             ++num_good_events;
129         }
130 
131         // clean up and get next event
132         delete evt;
133         ascii_in >> evt;
134     }
135     //.....................................PRINT RESULT
136     os << num_good_events << " out of " << icount
137                << " processed events passed the cuts. Finished." << std::endl;
138 }
139 
140 void read_variousFormats(std::ostream & os)
141 {
142     os << std::endl;
143     os << "process varied input" << std::endl;
144     // declare an input strategy
145     HepMC::IO_GenEvent ascii_in("@srcdir@/testHepMCVarious.input",std::ios::in);
146     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
147     // declare another IO_GenEvent for writing out the good events
148     HepMC::IO_GenEvent ascii_out("testHepMCVarious.out",std::ios::out);
149     //.....................................EVENT LOOP
150     int icount=0;
151     HepMC::GenEvent* evt = ascii_in.read_next_event();
152     while ( evt ) {
153         icount++;
154         double pim;
155         os << "Processing Event Number " << icount
156                  << " its # " << evt->event_number()
157                  << std::endl;
158         ascii_out << evt;
159         // units should be unknown
160         evt->write_units(os);
161         pim = findPiZero(evt);
162         os << " pizero mass: " << pim << std::endl;
163         if( HepMC::Units::name( evt->momentum_unit() ) == "GEV" ) {
164             os << " GenEvent units are GeV" << std::endl;
165             if( pim > 1.0 ) {
166                 // presume units are MEV and out of sync
167                 os << " pizero units are MeV" << std::endl;
168                 repairUnits(evt,HepMC::Units::MEV,HepMC::Units::GEV);
169                 // set units to MeV and mm
170                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
171                 evt->write_units(os);
172                 pim = findPiZero(evt);
173                 os << " pizero mass: " << pim
174                          << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
175                 // convert units to MeV
176                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
177                 evt->write_units(os);
178                 pim = findPiZero(evt);
179                 os << " pizero mass: " << pim
180                          << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
181             } else if( pim > 0.1 ) {
182                 // presume units are GEV
183                 os << " pizero units are GeV" << std::endl;
184                 // set units to GeV and mm
185                 evt->use_units(HepMC::Units::GEV, HepMC::Units::MM);
186                 evt->write_units(os);
187                 pim = findPiZero(evt);
188                 os << " pizero mass: " << pim
189                          << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
190                 // convert units to MeV
```

```
191                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
192                 evt->write_units(os);
193                 pim = findPiZero(evt);
194                 os << " pizero mass: " << pim
195                           << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
196             } else {
197                 os << " pizero mass: " << pim
198                           << " is inconsistent with allowed units " << std::endl;
199             }
200         } else if( HepMC::Units::name( evt->momentum_unit() ) == "MEV" ) {
201             os << " GenEvent units are MeV" << std::endl;
202             if( pim > 1.0 ) {
203                 // presume units are MEV
204                 os << " pizero units are MeV" << std::endl;
205                 // set units to MeV and mm
206                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
207                 evt->write_units(os);
208                 pim = findPiZero(evt);
209                 os << " pizero mass: " << pim
210                           << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
211                 // convert units to MeV
212                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
213                 evt->write_units(os);
214                 pim = findPiZero(evt);
215                 os << " pizero mass: " << pim
216                           << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
217             } else if( pim > 0.1 ) {
218                 // presume units are GEV and out of sync
219                 os << " pizero units are GeV" << std::endl;
220                 repairUnits(evt,HepMC::Units::GEV,HepMC::Units::MEV);
221                 evt->write_units(os);
222                 pim = findPiZero(evt);
223                 os << " pizero mass: " << pim
224                           << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
225                 // convert units to MeV
226                 evt->use_units(HepMC::Units::MEV, HepMC::Units::MM);
227                 evt->write_units(os);
228                 pim = findPiZero(evt);
229                 os << " pizero mass: " << pim
230                           << " " << HepMC::Units::name( evt->momentum_unit() ) << std::endl;
231             } else {
232                 os << " pizero mass: " << pim
233                           << " is inconsistent with allowed units " << std::endl;
234             }
235         }
236         // clean up and get next event
237         delete evt;
238         ascii_in >> evt;
239     }
240     std::cout << "testHepMC: the HeavyIon and PdfInfo input stream errors are intentional" << std::end
241     //.....................................PRINT RESULT
242     os << icount << " events processed. Finished." << std::endl;
243 }
244
245 void writeWithCrossSection(std::ostream & os)
246 {
247     // declare an input strategy to read input data
248     // units are GeV and mm
249     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
250     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
251     // declare another IO_GenEvent for writing out some events
252     HepMC::IO_GenEvent ascii_out("testCrossSection.out",std::ios::out);
253     // declare an output stream for printing events
254     std::ofstream xout( "testCrossSection.cout" );
255     // create an empty GenCrossSection object
256     HepMC::GenCrossSection cross;
257     //.....................................EVENT LOOP
```

```
258     int icount=0;
259     const double xs0 = 0.00346;
260     const double xs1 = 0.12;
261     const double xs2 = 33.234;
262     const double xs3 = 459.345;
263     double xserr = 0.0001;
264     double wgt1, wgt2;
265     HepMC::GenEvent* evt = ascii_in.read_next_event();
266     while ( evt ) {
267         icount++;
268         // use a variety of arbitrary cross section values
269         if( icount < 10 ) {
270             const double xs = xs0 - 1.34 * xserr;
271             cross.set_cross_section( xs, xserr );
272         } else if( icount < 20 ) {
273             const double xs = xs1 - 1.34 * xserr;
274             cross.set_cross_section( xs, xserr );
275         } else if( icount < 30 ) {
276             const double xs = xs2 - 1.34 * xserr;
277             cross.set_cross_section( xs, xserr );
278         } else {
279             const double xs = xs3 - 1.34 * xserr;
280             cross.set_cross_section( xs, xserr );
281         }
282         xserr *= 0.99;
283         if ( icount == 10 ) xserr += 0.01;
284         if ( icount == 20 ) xserr += 0.4;
285         if ( icount == 30 ) xserr += 1.0;
286         // attach this cross section to the event
287         evt->set_cross_section( cross );
288         evt->write_cross_section(os);
289         // add weights
290         wgt1 = 0.9853 + (double)icount * 0.00033;
291         wgt2 = 0.9853 + (double)(icount+1) * 0.00033;
292         evt->weights().push_back(0.3456);
293         evt->weights()["weightName"] = wgt1;
294         evt->weights()["second weight name"] = wgt2;
295         if ( icount%20==1 ) {
296             os << "writeWithCrossSection: Processing Event Number " << icount
297                                     << " its # " << evt->event_number()
298                                     << std::endl;
299             ascii_out << evt;
300             evt->print(xout);
301         }
302 
303         // clean up and get next event
304         delete evt;
305         ascii_in >> evt;
306     }
307     //.....................................PRINT RESULT
308     os << "writeWithCrossSection processed " << icount << " events. Finished." << std::endl;
309 }
310 
311 void readWithCrossSection(std::ostream & os)
312 {
313     // read the file we just wrote
314     HepMC::IO_GenEvent ascii_in("testCrossSection.out",std::ios::in);
315     // declare another IO_GenEvent for writing out some events
316     HepMC::IO_GenEvent ascii_out("testCrossSection2.out",std::ios::out);
317     //.....................................EVENT LOOP
318     int icount=0;
319     HepMC::GenEvent* evt = ascii_in.read_next_event();
320     while ( evt ) {
321         ++icount;
322         os << "readWithCrossSection: Processing Event Number " << icount
323                                 << " its # " << evt->event_number()
324                                 << std::endl;
```

```
325         if (evt->cross_section()->cross_section() <= 0) {
326           os << "testReadCrossSection: invalid cross-section!" << std::endl;
327         }
328         ascii_out << evt;
329 
330         // clean up and get next event
331         delete evt;
332         ascii_in >> evt;
333     }
334     //.....................................PRINT RESULT
335     os << "readWithCrossSection processed " << icount << " events. Finished." << std::endl;
336 }
337 
338 void read_nan(std::ostream & os)
339 {
340     // Read an input file that has corrupt information (nan's)
341     //
342     HepMC::IO_GenEvent xin("@srcdir@/testHepMCVarious.input",std::ios::in);
343     HepMC::IO_GenEvent xout("testNaN.out",std::ios::out);
344     // set input units
345     xin.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
346     //.....................................EVENT LOOP
347     int icount=0;
348     int invaliddata=0;
349     bool ok = true;
350     os << "---------------------------------------- " << std::endl;
351     os << "Begin NaN test " << std::endl;
352     HepMC::GenEvent* evt = xin.read_next_event();
353     //
354     // To recover from corrupt input, replace "while(evt) {...}"
355     // with  "while(ok) { if(evt) {... xin >> evt;} else {...} }"
356     //
357     while ( ok ) {
358         if( evt ) {
359             ++icount;
360             os << "read_nan: Processing Event Number " << icount
361                       << " its # " << evt->event_number()
362                       << std::endl;
363             xout << evt;
364             // clean up and get next event
365             delete evt;
366             xin >> evt;
367         } else if (xin.error_type() == HepMC::IO_Exception::InvalidData ) {
368             ++invaliddata;
369             os << "INPUT ERROR: " << xin.error_message() << std::endl;
370             // clean up and get next event
371             delete evt;
372             xin >> evt;
373         } else if (invaliddata > 50 ) {
374             os << "INPUT ERROR: " << xin.error_message() << std::endl;
375             ok = false;
376         } else {
377             ok = false;
378         }
379     }
380     // print status of input stream
381     if ( xin.error_type() != 0 ) {
382         os << "processing of @srcdir@/testHepMCVarious.input ended with error "
383                   << xin.error_type() << std::endl;
384         os << "  --- " << xin.error_message() << std::endl;
385     }
386     os << icount << " events processed and "
387               << invaliddata << " events ignored. Finished."
388               << std::endl;
389     os << "End NaN test " << std::endl;
390     os << "---------------------------------------- " << std::endl;
391 }
```

```
392
393 void writeWithWeight(std::ostream & os)
394 {
395     // declare an input strategy to read input data
396     // units are GeV and mm
397     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
398     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
399     // declare another IO_GenEvent for writing out some events
400     HepMC::IO_GenEvent ascii_out("testWithWeight.out",std::ios::out);
401     // declare an output stream for printing events
402     std::ofstream xout( "testWithWeight.cout" );
403     //.....................................EVENT LOOP
404     int icount=0;
405     double wgt1, wgt2;
406     HepMC::GenEvent* evt = ascii_in.read_next_event();
407     while ( evt ) {
408         icount++;
409         // add weights
410         wgt1 = 0.9853 + (double)icount * 0.00033;
411         wgt2 = 0.9853 + (double)(icount+1) * 0.00033;
412         evt->weights().push_back(0.3456);
413         evt->weights().push_back(wgt1);
414         evt->weights().push_back(wgt2);
415         if ( icount%20==1 ) {
416             os << "writeWithWeight: Processing Event Number " << icount
417                                     << " its # " << evt->event_number()
418                                     << std::endl;
419             ascii_out << evt;
420             evt->print(xout);
421         }
422
423         // clean up and get next event
424         delete evt;
425         ascii_in >> evt;
426     }
427     //.....................................PRINT RESULT
428     os << "writeWithWeight processed " << icount << " events. Finished." << std::endl;
429 }
430
431 void readWithWeight(std::ostream & os)
432 {
433     // read the file we just wrote
434     HepMC::IO_GenEvent ascii_in("testWithWeight.out",std::ios::in);
435     // declare another IO_GenEvent for writing out some events
436     HepMC::IO_GenEvent ascii_out("testWithWeight2.out",std::ios::out);
437     //.....................................EVENT LOOP
438     int icount=0;
439     HepMC::GenEvent* evt = ascii_in.read_next_event();
440     while ( evt ) {
441         ++icount;
442         os << "readWithWeight: Processing Event Number " << icount
443                                 << " its # " << evt->event_number()
444                                 << std::endl;
445         if ( !evt->cross_section() ) {
446           os << "testReadCrossSection: invalid cross-section!" << std::endl;
447         }
448         ascii_out << evt;
449
450         // clean up and get next event
451         delete evt;
452         ascii_in >> evt;
453     }
454     //.....................................PRINT RESULT
455     os << "readWithWeight processed " << icount << " events. Finished." << std::endl;
456 }
```

11.13 testHepMCIteration.cc.in

Use Matt's example_EventSelection along with example_UsingIterators to check **HepMC** (p. 25) iteration. Apply an event selection to the events in testHepMC.input Events containing a photon of pT > 25 GeV pass the selection. Use iterators on these events.

```
1
2 // testHepMCIteration.cc.in
3 //
4 // garren@fnal.gov, May 2007
5 // Use Matt's example_EventSelection along with example_UsingIterators
6 // to check HepMC iteration.
7 // Apply an event selection to the events in testHepMC.input
8 // Events containing a photon of pT > 25 GeV pass the selection.
9 // Use iterators on these events.
11
12 #include <list>
13
14 #include "HepMC/IO_GenEvent.h"
15 #include "HepMC/IO_AsciiParticles.h"
16 #include "HepMC/GenEvent.h"
17 #include "HepMC/GenRanges.h"
18
19 // define methods and classes used by this test
20 #include "IsGoodEvent.h"
21 #include "testHepMCIteration.h"
22
23 bool findW( HepMC::GenEvent* evt, std::ofstream& os);
24 bool simpleIter ( HepMC::GenEvent* evt, std::ostream& os = std::cout );
25 bool simpleIter2( HepMC::GenEvent* evt, std::ostream& os = std::cout );
26 bool simpleIter3( HepMC::GenEvent* evt, std::ostream& os = std::cout );
27 bool simpleIter4( HepMC::GenEvent* evt, std::ostream& os = std::cout );
28
33 class PrintW {
34 public:
35     PrintW( std::ostream & os, int num ) : m_out( os ),m_event_num( num ) {}
36     void operator()( HepMC::GenParticle* p ) {
37         if ( IsWBoson(p) ) {
38             m_out << std::endl;
39             m_out << "A W boson has been found in event: " << m_event_num << std::endl;
40             p->print( m_out );
41             // return all parents
42             // we do this by pointing to the production vertex of the W
43             // particle and asking for all particle parents of that vertex
44             m_out << "\t Its parents are: " << std::endl;
45             if ( p->production_vertex() ) {
46                 std::for_each( p->particles_in(HepMC::parents).begin(),
47                                p->particles_in(HepMC::parents).end(),
48                                PrintParticle(m_out));
49             }
50
51             // return immediate children
52             m_out << "\t\t" << "Its children are: " << std::endl;
53             if ( p->end_vertex() ) {
54                 std::for_each( p->particles_out(HepMC::children).begin(),
55                                p->particles_out(HepMC::children).end(),
56                                PrintChildren(m_out));
57             }
58
59             // return all descendants
60             // we do this by pointing to the end vertex of the W
61             // particle and asking for all particle descendants of that vertex
62             m_out << "\t\t Its descendants are: " << std::endl;
63             if ( p->end_vertex() ) {
64                 std::for_each( p->particles_out(HepMC::descendants).begin(),
```

```
65                                  p->particles_out(HepMC::descendants).end(),
66                                  PrintDescendants(m_out));
67              }
68          }       // if IsWBoson
69      }
70  private:
71     std::ostream & m_out;
72     int            m_event_num;
73  };
74
79  class PrintConstW {
80  public:
81      PrintConstW( std::ostream & os, int num ) : m_out( os ),m_event_num( num ) {}
82      void operator()( HepMC::GenParticle* p ) {
83          if ( IsWBoson(p) ) {
84              m_out << std::endl;
85              m_out << "A W boson has been found in event: " << m_event_num << std::endl;
86              p->print( m_out );
87              // return all parents
88              // we do this by pointing to the production vertex of the W
89              // particle and asking for all particle parents of that vertex
90              m_out << "\t Its parents are: " << std::endl;
91              if ( p->production_vertex() ) {
92                  std::for_each( p->particles_in(HepMC::parents).begin(),
93                                 p->particles_in(HepMC::parents).end(),
94                                 PrintParticle(m_out));
95              }
96
97              // return immediate children
98              m_out << "\t\t" << "Its children are: " << std::endl;
99              if ( p->end_vertex() ) {
100                 std::for_each( p->particles_out(HepMC::children).begin(),
101                                p->particles_out(HepMC::children).end(),
102                                PrintChildren(m_out));
103             }
104
105             // return all descendants
106             // we do this by pointing to the end vertex of the W
107             // particle and asking for all particle descendants of that vertex
108             m_out << "\t\t Its descendants are: " << std::endl;
109             if ( p->end_vertex() ) {
110                 std::for_each( p->particles_out(HepMC::descendants).begin(),
111                                p->particles_out(HepMC::descendants).end(),
112                                PrintDescendants(m_out));
113             }
114         }       // if IsWBoson
115     }
116 private:
117    std::ostream & m_out;
118    int            m_event_num;
119 };
120
121 int main() {
122     // declare an input strategy to read the data produced with the
123     // example_MyPythia
124     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
125     // declare an instance of the event selection predicate
126     IsGoodEvent is_good_event;
127     // define some output streams
128     std::ofstream os( "testHepMCIteration.out" );
129     std::ofstream os2( "testHepMCIteration2.out" );
130     std::ofstream os3( "testHepMCIteration3.out" );
131     //.....................................EVENT LOOP
132     int icount=0;
133     int num_good_events=0;
134     HepMC::GenEvent* evt = ascii_in.read_next_event();
135     HepMC::GenEvent* evcopy;
```

```
136     while ( evt ) {
137         icount++;
138         if ( icount%50==1 ) std::cout << "Processing Event Number " << icount
139                                       << " its # " << evt->event_number()
140                                       << std::endl;
141         // icount of 100 should be the last event
142         if ( icount==100 ) std::cout << "Processing Event Number " << icount
143                                       << " its # " << evt->event_number()
144                                       << std::endl;
145         evcopy = evt;
146         if ( is_good_event(evcopy) ) {
147             ++num_good_events;
148             // simple iteration several different ways
149             os << "Event " << evcopy->event_number() << " is good " << std::endl;
150             simpleIter( evcopy, os );
151             os2 << "Event " << evcopy->event_number() << " is good " << std::endl;
152             simpleIter2( evcopy, os2 );
153             os3 << "Event " << evcopy->event_number() << " is good " << std::endl;
154             simpleIter2( evcopy, os3 );
155             std::cout << "Event " << evcopy->event_number() << " is good " << std::endl;
156             simpleIter3( evcopy );
157             simpleIter4( evcopy );
158             // test iterators
159             findW( evcopy, os );
160             // this is the same as findW except that we use the STL for_each algorithm
161             std::for_each( evt->particles_begin(), evt->particles_end(),
162                            PrintW(os2,evcopy->event_number()));
163             // repeat, using the const iterator
164             std::for_each( evt->particles_begin(), evt->particles_end(),
165                            PrintConstW(os3,evcopy->event_number()));
166         }
167         evcopy->clear();
168 
169         // clean up and get next event
170         delete evt;
171         evt = ascii_in.read_next_event();
172     }
173     //.....................................PRINT RESULT
174     std::cout << num_good_events << " out of " << icount
175               << " processed events passed the cuts. Finished." << std::endl;
176 }
177 
178 bool simpleIter( HepMC::GenEvent* evt, std::ostream& os )
179 {
180     // use GenEvent::vertex_iterator to fill a list of all
181     // vertices in the event
182     std::list<HepMC::GenVertex*> allvertices;
183     for ( HepMC::GenEvent::vertex_iterator v = evt->vertices_begin();
184           v != evt->vertices_end(); ++v ) {
185         allvertices.push_back(*v);
186     }
187 
188     // fill a list of all final state particles in the event, by requiring
189     // that each particle satisfyies the IsFinalState predicate
190     IsFinalState isfinal;
191     std::list<HepMC::GenParticle*> finalstateparticles;
192     for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
193           p != evt->particles_end(); ++p ) {
194         if ( isfinal(*p) ) finalstateparticles.push_back(*p);
195     }
196 
197     // print all photons in the event that satisfy the IsPhoton criteria
198     os << "photons in event " << evt->event_number() << ":" << std::endl;
199     for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
200           p != evt->particles_end(); ++p ) {
201         if ( IsPhoton(*p) ) (*p)->print( os );
202     }
```

```
203
204     return true;
205 }
206
207 bool simpleIter2( HepMC::GenEvent* evt, std::ostream& os )
208 {
209     // illustrates the use various helpful algorithms
210
211     // use the STL copy algorithm to fill a list of all
212     // vertices in the event
213     std::list<HepMC::GenVertex*> allvertices2;
214     copy( evt->vertices_begin(), evt->vertices_end(),
215           back_inserter(allvertices2) );
216
217     // fill a list of all final state particles in the event, by requiring
218     // that each particle satisfyies the IsFinalState predicate
219     // an STL-like algorithm called HepMC::copy_if is provided in the
220     // GenEvent.h header to do this sort of operation more easily
221     std::list<HepMC::GenParticle*> finalstateparticles2;
222     HepMC::copy_if( evt->particles_begin(), evt->particles_end(),
223                     back_inserter(finalstateparticles2), IsFinalState() );
224
225     // use the STL for_each algorithm to
226     // print all photons in the event that satisfy the IsPhoton criteria
227     os << "photons in event " << evt->event_number() << ":" << std::endl;
228     std::for_each(evt->particles_begin(), evt->particles_end(),
229                   PrintPhoton(os));
230
231     return true;
232 }
233
234 bool simpleIter3( HepMC::GenEvent* evt, std::ostream& os )
235 {
236     // very simple illustration of using GenEventVertexRange
237     // and GenEventParticleRange
238     // NOTE that instead of creating this list,
239     // you can just use GenEventVertexRange as if it were the list
240     std::list<HepMC::GenVertex*> allvertices;
241     HepMC::GenEventVertexRange vc(*evt);
242     for ( HepMC::GenEvent::vertex_iterator v = vc.begin(); v != vc.end(); ++v ) {
243         allvertices.push_back(*v);
244     }
245
246     // fill a list of all final state particles in the event, by requiring
247     // that each particle satisfyies the IsFinalState predicate
248     IsFinalState isfinal;
249     std::list<HepMC::GenParticle*> finalstateparticles;
250     HepMC::GenEventParticleRange pc(*evt);
251     for ( HepMC::GenEvent::particle_iterator p = pc.begin(); p != pc.end(); ++p ) {
252         if ( isfinal(*p) ) finalstateparticles.push_back(*p);
253     }
254
255     // print all photons in the event that satisfy the IsPhoton criteria
256     os << "photons in event " << evt->event_number() << ":" << std::endl;
257     std::for_each(pc.begin(), pc.end(), PrintPhoton(os));
258
259     return true;
260 }
261
262 bool simpleIter4( HepMC::GenEvent* evt, std::ostream& os )
263 {
264     // very simple illustration of using
265     // GenEvent::vertex_range(), which returns GenEventVertexRange,
266     // and GenEvent::particle_range(), which returns GenEventParticleRange
267     // NOTE that instead of creating these lists,
268     // you can just use GenEvent::vertex_range() and GenEvent::particle_range()
269     // as if they were a list
```

```
270
271     std::list<HepMC::GenVertex*> allvertices;
272     for ( HepMC::GenEvent::vertex_iterator v = evt->vertex_range().begin();
273           v != evt->vertex_range().end(); ++v ) {
274         allvertices.push_back(*v);
275     }
276
277     // fill a list of all final state particles in the event, by requiring
278     // that each particle satisfyies the IsFinalState predicate
279     IsFinalState isfinal;
280     std::list<HepMC::GenParticle*> finalstateparticles;
281     for ( HepMC::GenEvent::particle_iterator p = evt->particle_range().begin();
282           p != evt->particle_range().end(); ++p ) {
283         if ( isfinal(*p) ) finalstateparticles.push_back(*p);
284     }
285
286     // print all photons in the event that satisfy the IsPhoton criteria
287     os << "photons in event " << evt->event_number() << ":" << std::endl;
288     std::for_each(evt->particle_range().begin(),
289                   evt->particle_range().end(),
290                   PrintPhoton(os));
291
292     return true;
293 }
294
295 bool findW( HepMC::GenEvent* evt, std::ofstream& os )
296 {
297     int num_W = 0;
298     // use GenEvent::particle_iterator to find all W's in the event,
299     // then
300     // (1) for each W user the GenVertex::particle_iterator with a range of
301     //     parents to return and print the immediate mothers of these W's.
302     // (2) for each W user the GenVertex::particle_iterator with a range of
303     //     descendants to return and print all descendants of these W's.
304     for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
305           p != evt->particles_end(); ++p ) {
306         if ( IsWBoson(*p) ) {
307             ++num_W;
308             os << std::endl;
309             os << "A W boson has been found in event: " << evt->event_number() << std::endl;
310             (*p)->print( os );
311             // return all parents
312             // we do this by pointing to the production vertex of the W
313             // particle and asking for all particle parents of that vertex
314             os << "\t Its parents are: " << std::endl;
315             if ( (*p)->production_vertex() ) {
316                 for ( HepMC::GenVertex::particle_iterator mother
317                           = (*p)->production_vertex()->
318                           particles_begin(HepMC::parents);
319                       mother != (*p)->production_vertex()->
320                           particles_end(HepMC::parents);
321                       ++mother ) {
322                     os << "\t";
323                     (*mother)->print( os );
324                 }
325             }
326
327             // return immediate children
328             os << "\t\t" << "Its children are: " << std::endl;
329             if ( (*p)->end_vertex() ) {
330                 for ( HepMC::GenVertex::particle_iterator child =
331                       (*p)->end_vertex()->particles_begin(HepMC::children);
332                       child != (*p)->end_vertex()->particles_end(HepMC::children);
333                       ++child ) {
334                     // make a copy
335                     HepMC::GenVertex::particle_iterator cp = child;
336                     // use the copy and the original
```

```
337                     os << "\t\t\t (id,barcode,status) "
338                        << (*cp)->pdg_id() << " "
339                        << (*child)->barcode() << " "
340                        << (*cp)->status() << std::endl;
341                 }
342             }
343 
344             // return all descendants
345             // we do this by pointing to the end vertex of the W
346             // particle and asking for all particle descendants of that vertex
347             os << "\t\t Its descendants are: " << std::endl;
348             if ( (*p)->end_vertex() ) {
349                 for ( HepMC::GenVertex::particle_iterator des
350                           =(*p)->end_vertex()->
351                           particles_begin(HepMC::descendants);
352                       des != (*p)->end_vertex()->
353                           particles_end(HepMC::descendants);
354                       ++des ) {
355                     os << "\t\t";
356                     (*des)->print( os );
357                 }
358             }
359         }       // if IsWBoson
360     }   // end particle loop
361     return true;
362 }
```

11.14 testMass.cc.in

Read events from testIOGenEvent.input Select events containing a photon of pT > 25 GeV Add arbitrary PDF information to one of the good events Write the selected events and read them back in using an istream

```
1 //----------------------------------------------------------------------
2 // testMass.cc.in
3 //
4 // garren@fnal.gov, March 2006
5 // Read events written by example_MyPythia.cc
6 // Select events containing a photon of pT > 25 GeV
7 // Add arbitrary PDF information to one of the good events
8 // Add arbitrary HeavyIon information to one of the good events
9 // Write the selected events and read them back in using an istream
10 //----------------------------------------------------------------------
11
12 #include <cmath>        // for min()
13 #include <ostream>
14
15 #include "HepMC/IO_GenEvent.h"
16 #include "HepMC/GenEvent.h"
17 #include "HepMC/Version.h"
18
19 // define methods and classes used by this test
20 #include "IsGoodEvent.h"
21
22 void   massInfo( const HepMC::GenEvent*, std::ostream& os );
23
24 int main() {
25     // output file
26     std::ofstream os( "testMass.cout" );
27     // read and process the input file
28     {
29         // declare an input strategy to read the data produced with the
30         // example_MyPythia
31         HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
32         ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
33         // declare another IO_GenEvent for output
34         HepMC::IO_GenEvent ascii_out("testMass1.out",std::ios::out);
35         // declare an instance of the event selection predicate
36         IsGoodEvent is_good_event;
37         // send version to output
38         HepMC::version(os);
39         //.....................................EVENT LOOP
40         int icount=0;
41         int num_good_events=0;
42         double x1=0., x2=0., q=0., xf1=0., xf2=0.;
43         HepMC::GenEvent* evt = ascii_in.read_next_event();
44         while ( evt ) {
45             icount++;
46             if ( icount%50==1 ) os << "Processing Event Number " << icount
47                                    << " its # " << evt->event_number()
48                                    << std::endl;
49             if ( is_good_event(evt) ) {
50                 if (num_good_events == 0 ) {
51                     // add some arbitrary PDF information
52                     x1 = std::min(0.8, 0.07 * icount);
53                     x2 = 1-x1;
54                     q = 1.69 * icount;
55                     // use beam momentum
56                     if( evt->valid_beam_particles() ) {
57                         HepMC::GenParticle* bp1 = evt->beam_particles().first;
58                         xf1 = x1*bp1->momentum().rho();
59                         xf2 = x2*bp1->momentum().rho();
```

```
60                      } else {
61                          xf1 = x1*0.34;
62                          xf2 = x2*0.34;
63                      }
64                      // provide optional pdf set id numbers
65                      // (two ints at the end of the constructor)
66                      HepMC::PdfInfo pdf( 2, 3, x1, x2, q, xf1, xf2, 230, 230);
67                      evt->set_pdf_info(pdf);
68                      // add some arbitrary HeavyIon information
69                      HepMC::HeavyIon ion(23,11,12,15,3,5,0,0,0,0.0145);
70                      evt->set_heavy_ion( ion );
71                  }
72                  os << "saving Event " << evt->event_number() << std::endl;
73                  if( evt->weights().size() > 0 ) {
74                      os << "Weights: ";
75                      evt->weights().print(os);
76                  }
77                  ascii_out << evt;
78                  ++num_good_events;
79              }
80
81              // clean up and get next event
82              delete evt;
83              ascii_in >> evt;
84          }
85          //.....................................PRINT RESULT
86          os << num_good_events << " out of " << icount
87             << " processed events passed the cuts. Finished." << std::endl;
88      }
89      // now read the file we just created
90      {
91          // declare an input strategy
92          const char infile[] = "testMass1.out";
93          std::ifstream istr( infile );
94          if( !istr ) {
95            std::cerr << "testMass: cannot open " << infile << std::endl;
96            exit(-1);
97          }
98          HepMC::IO_GenEvent xin(istr);
99          // declare another IO_GenEvent for output
100          HepMC::IO_GenEvent xout("testMass2.out",std::ios::out);
101          //.....................................EVENT LOOP
102          int ixin=0;
103          HepMC::GenEvent* evt = xin.read_next_event();
104          while ( evt ) {
105              ixin++;
106              os << "reading Event " << evt->event_number() << std::endl;
107              if( evt->weights().size() > 0 ) {
108                  os << "Weights: ";
109                  evt->weights().print(os);
110              }
111              xout << evt;
112              // look at mass info
113              massInfo(evt,os);
114
115              // clean up and get next event
116              delete evt;
117              xin >> evt;
118          }
119          //.....................................PRINT RESULT
120          os << ixin << " events in the second pass. Finished." << std::endl;
121      }
122  }
123
124  void massInfo( const HepMC::GenEvent* e, std::ostream& os )
125  {
126     double gm, m, d;
```

```
127    for ( HepMC::GenEvent::particle_const_iterator p = e->particles_begin(); p != e->particles_end();
128          ++p ) {
129
130        gm = (*p)->generated_mass();
131        m = (*p)->momentum().m();
132        d = fabs(m-gm);
133        if( d > 1.0e-5 ) {
134             os << "Event " << e->event_number()
135                << " Particle " << (*p)->barcode()
136                << " " << (*p)->pdg_id()
137                << " generated mass " << gm
138                << " mass from momentum " << m
139                << " difference " << d << std::endl;
140        }
141    }
142 }
```

11.15 testMultipleCopies.cc.in

Multiple events in memory at the same time run with valgrind or some other leak checker

```
1
2 // testMultipleCopies.cc.in
3 //
4 // garren@fnal.gov, January 2008
5 // Multiple events in memory at the same time
6 // run with valgrind or some other leak checker
8 //
9
10 #include <fstream>
11
12 #include "HepMC/IO_GenEvent.h"
13 #include "HepMC/GenEvent.h"
14 #include "HepMC/CompareGenEvent.h"
15
16 // define methods and classes used by this test
17 #include "IsGoodEvent.h"
18
19 int main() {
20     // use output file
21     std::ofstream os( "testMultipleCopies.out" );
22     {
23         // declare an input strategy
24         HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
25         // declare another input strategy
26         HepMC::IO_GenEvent ascii_in2("@srcdir@/testHepMCVarious.input",std::ios::in);
27         std::ofstream out1( "testMultipleOriginals.out" );
28         std::ofstream out2( "testMultipleCopies1.out" );
29         std::ofstream out3( "testMultipleCopies2.out" );
30         // declare an instance of the event selection predicate
31         IsGoodEvent is_good_event;
32
33         //.....................................EVENT LOOP
34         int icount=0;
35         int num_good_events=0;
36         int icnt;
37         HepMC::GenEvent* evt1 = ascii_in.read_next_event();
38         HepMC::GenEvent* evt2 = ascii_in2.read_next_event();
39         HepMC::GenEvent* evt3 = ascii_in.read_next_event();
40
41         while ( evt1 && evt2 ) {
42             icount++;
43             if ( icount%50==1 ) os << "Processing Event Number " << icount
44                                    << " stream 1 # " << evt1->event_number()
45                                    << " stream 2 # " << evt2->event_number()
46                                    << std::endl;
47
48             if ( is_good_event(evt1) ) {
49
50                 os << "good event in stream 1 # "
51                    << evt1->event_number() << std::endl;
52                 evt1->print(out1);
53                 ++num_good_events;
54                 HepMC::GenEvent ec = (*evt1);
55                 ec.print(out3);
56                 icnt=0;
57                 for ( HepMC::GenEvent::particle_const_iterator p1 =  ec.particles_begin();
58                       p1 !=  ec.particles_end(); ++p1 ) {
59                       ++icnt;
60                     os << "particle " << icnt << " barcode " <<(*p1)->barcode() << std::endl;
61                 }
62                 HepMC::GenEvent* evt4 = new HepMC::GenEvent(*evt1);
```

```
63                 evt4->print(out2);
64                 if( !compareGenEvent(evt1,evt4) ) { return -1; }
65                 delete evt4;
66             }
67
68             // clean up and get next events
69             delete evt1;
70             delete evt2;
71             ascii_in >> evt1;
72             ascii_in2 >> evt2;
73         }
74         // might have either evt1 or evt2 still in memory, cleanup here
75         delete evt1;
76         delete evt2;
77         delete evt3;
78
79         //.....................................PRINT RESULT
80         os << std::endl;
81         os << num_good_events << " out of " << icount
82            << " processed events passed the cuts." << std::endl;
83         os << std::endl;
84         os << " GenEvent copy constructor passes the test" << std::endl;
85         os << std::endl;
86     }
87
88     // test operator= and swap
89     {
90         // declare an input strategy
91         HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
92         //
93         HepMC::GenEvent* evt5 = ascii_in.read_next_event();
94         HepMC::GenEvent* evt6 = new HepMC::GenEvent();
95         os << "event number for evt5: " << evt5->event_number() << std::endl;
96         os << "event number for evt6: " << evt6->event_number() << std::endl;
97         // copy  GenEvent object
98         (*evt6) = (*evt5);
99         if( !compareGenEvent(evt5,evt6) ) { return -4; }
100         delete evt5;
101         os << "event number for evt6 after copy: " << evt6->event_number() << std::endl;
102         os << std::endl;
103         delete evt6;
104         os << " GenEvent operator= passes the test" << std::endl;
105         os << std::endl;
106
107         evt5 = ascii_in.read_next_event();
108         evt6 = ascii_in.read_next_event();
109         HepMC::GenEvent* evt7 = new HepMC::GenEvent(*evt5);
110         HepMC::GenEvent* evt8 = new HepMC::GenEvent(*evt6);
111         os << "event number for evt5: " << evt5->event_number() << std::endl;
112         os << "event number for evt6: " << evt6->event_number() << std::endl;
113         os << "before swap, evt5 has: " << evt5->vertices_size() << " vertices and "
114            << evt5->particles_size() << " particles" << std::endl;
115         os << "before swap, evt6 has: " << evt6->vertices_size() << " vertices and "
116            << evt6->particles_size() << " particles" << std::endl;
117         os << "before swap, evt7 has: " << evt7->vertices_size() << " vertices and "
118            << evt7->particles_size() << " particles" << std::endl;
119         os << "before swap, evt8 has: " << evt8->vertices_size() << " vertices and "
120            << evt8->particles_size() << " particles" << std::endl;
121         (*evt6).swap((*evt5));
122         os << "event number for evt5 after swap: " << evt5->event_number() << std::endl;
123         os << "event number for evt6 after swap: " << evt6->event_number() << std::endl;
124         // evt6 should now match evt7
125         os << "after swap, evt6 has: " << evt6->vertices_size() << " vertices and "
126            << evt6->particles_size() << " particles" << std::endl;
127         os << "after swap, evt7 has: " << evt7->vertices_size() << " vertices and "
128            << evt7->particles_size() << " particles" << std::endl;
129         if( !compareGenEvent(evt6,evt7) ) { return -6; }
```

```
130         // evt5 should now match evt8
131         os << "after swap, evt5 has: " << evt5->vertices_size() << " vertices and "
132            << evt5->particles_size() << " particles" << std::endl;
133         os << "after swap, evt8 has: " << evt8->vertices_size() << " vertices and "
134            << evt8->particles_size() << " particles" << std::endl;
135         if( !compareGenEvent(evt5,evt8) ) { return -5; }
136         // cleanup
137         delete evt5;
138         delete evt6;
139         delete evt7;
140         delete evt8;
141         os << std::endl;
142         os << " GenEvent swap passes the test" << std::endl;
143         os << std::endl;
144     }
145     return 0;
146 }
```

11.16 testPrintBug.cc

Thanks to Bob McElrath and Frank Siegert for this test

```
1 //
2 // Thanks to Bob McElrath and Frank Siegert for this test
3 //
4
5 #include <fstream>
6
7 #include "HepMC/GenEvent.h"
8 #include "HepMC/SimpleVector.h"
9
10 int main()
11 {
12   HepMC::GenEvent* p_event;
13
14   p_event = new HepMC::GenEvent();
15   p_event->use_units(HepMC::Units::GEV, HepMC::Units::MM);
16
17   // define an output stream
18   std::ofstream os( "testPrintBug.out" );
19
20   for(int i=0; i<10; i++) {
21     HepMC::FourVector vector(1.0,1.0,1.0,1.0);
22     HepMC::GenVertex* vertex = new HepMC::GenVertex(vector,i);
23     for(int j=0; j<3; j++) {
24       HepMC::GenParticle* particle = new HepMC::GenParticle(vector,1,2);
25       vertex->add_particle_in(particle);
26     }
27     for(int j=0; j<3; j++) {
28       HepMC::GenParticle* particle = new HepMC::GenParticle(vector,1,2);
29       vertex->add_particle_out(particle);
30     }
31     p_event->add_vertex(vertex);
32   }
33   p_event->print(os);
34   // cleanup
35   delete p_event;
36   return 0;
37 }
```

11.17 testSimpleVector.cc

Exercise all the vector methods

```
1 //
2 // First pass - simply exercise all the vector methods
3 //
4 #include <iostream>
5
6 #include "HepMC/SimpleVector.h"
7
8 int main()
9 {
10   // ThreeVector
11   HepMC::ThreeVector vector3;
12   HepMC::ThreeVector v3(1.1,2.2,3.3);
13   HepMC::ThreeVector vx(1.34);
14
15   HepMC::ThreeVector v3copy( v3 );
16
17   double eps = 1.e-15; // allowed differnce between doubles
18   int numbad = 0;
19
20   double x = v3.x();
21   double y = v3.y();
22   double z = v3.z();
23   double p2 = v3.perp2();
24   double pt = v3.perp();
25   double r = v3.r();
26   double th = v3.theta();
27   double ph = v3.phi();
28   double mag = std::sqrt(x*x + y*y + z*z);
29   double pperp = std::sqrt(x*x + y*y);
30
31   vx.set(1., 2., 3.);
32   vx.setX(1.1);
33   vx.setY(2.3);
34   vx.setZ(4.4);
35   vx.setPhi(0.12);
36   vx.setTheta(0.54);
37
38   vector3 = v3;
39
40   if( fabs( mag - r ) > eps ) {
41      std::cout << "different ThreeVector magnitude: " << mag << " " << r << std::endl;
42      std::cout << "difference is : " << ( mag - r ) << std::endl;
43      ++numbad;
44   }
45   if( fabs( pperp - pt ) > eps ) {
46      std::cout << "different ThreeVector Pt: " << pperp << " " << pt << std::endl;
47      std::cout << "difference is : " << ( pperp - pt ) << std::endl;
48      ++numbad;
49   }
50
51   if( v3 == vector3 ) {
52   } else {
53      ++numbad;
54      std::cout << "vectors v3 and vector3 are different" << std::endl;
55   }
56   if( v3 != v3copy ) {
57      ++numbad;
58      std::cout << "vectors v3 and v3copy are different" << std::endl;
59   }
60
61   // FourVector
62   HepMC::FourVector vector;
```

```
63   HepMC::FourVector v4(1.1,2.2,3.3,4.4);
64   HepMC::FourVector vt(1.34);
65
66   HepMC::FourVector vectorcopy( v4 );
67   vector = v4;
68
69   double px = v4.px();
70   double py = v4.py();
71   double pz = v4.pz();
72   double e  = v4.e();
73    x = vectorcopy.x();
74    y = vectorcopy.y();
75    z = vectorcopy.z();
76   double t = vectorcopy.t();
77
78    p2 = v4.perp2();
79    pt = v4.perp();
80    th = v4.theta();
81    ph = v4.phi();
82    r = v4.rho();
83   double masssq1 = v4.m2();
84   double mass1 = v4.m();
85   double pr1 = v4.pseudoRapidity();
86   double eta1 = v4.eta();
87   double masssq2 = vector.m2();
88   double mass2 = vector.m();
89   double pr2 = vector.pseudoRapidity();
90   double eta2 = vector.eta();
91
92   vt.set(1., 2., 3., 5.5);
93   vt.setX(1.1);
94   vt.setY(2.3);
95   vt.setZ(4.4);
96   vt.setT(6.5);
97   vt.setPx(3.1);
98   vt.setPy(2.2);
99   vt.setPz(-1.1);
100   vt.setE(5.4);
101
102   mag = std::sqrt(x*x + y*y + z*z);
103   pperp = std::sqrt(x*x + y*y);
104   if( fabs( mag - r ) > eps ) {
105      std::cout << "different FourVector magnitude: " << mag << " " << r << std::endl;
106      std::cout << "difference is : " << ( mag - r ) << std::endl;
107      ++numbad;
108   }
109   if( fabs( pperp - pt ) > eps ) {
110      std::cout << "different FourVector Pt: " << pperp << " " << pt << std::endl;
111      std::cout << "difference is : " << ( pperp - pt ) << std::endl;
112      ++numbad;
113   }
114
115   if( px != x ) {
116      std::cout << "different X values: " << px << " " << x << std::endl;
117      ++numbad;
118   }
119   if( py != y ) {
120      std::cout << "different Y values: " << py << " " << y << std::endl;
121      ++numbad;
122   }
123   if( pz != z ) {
124      std::cout << "different Z values: " << pz << " " << z << std::endl;
125      ++numbad;
126   }
127   if( e != t ) {
128      std::cout << "different E values: " << e << " " << t << std::endl;
129      ++numbad;
```

```
130   }
131   if( fabs( masssq1 - masssq2 ) > eps ) {
132      std::cout << "different mass sq values: " << masssq1 << " " << masssq2 << std::endl;
133      std::cout << "difference is : " << ( masssq1 - masssq2 ) << std::endl;
134      ++numbad;
135   }
136   if( fabs( mass1 - mass2 ) > eps ) {
137      std::cout << "different mass values: " << mass1 << " " << mass2 << std::endl;
138      std::cout << "difference is : " << ( mass1 - mass2 ) << std::endl;
139      ++numbad;
140   }
141   if( fabs( pr1 - pr2 ) > eps ) {
142      std::cout << "different pseudorapidity values: " << pr1 << " " << pr2 << std::endl;
143      std::cout << "difference is : " << ( pr1 - pr2 ) << std::endl;
144      ++numbad;
145   }
146   if( fabs( eta1 - eta2 ) > eps ) {
147      std::cout << "different eta values: " << eta1 << " " << eta2 << std::endl;
148      std::cout << "difference is : " << ( eta1 - eta2 ) << std::endl;
149      ++numbad;
150   }
151   if( v4 == vector ) {
152   } else {
153      std::cout << "vectors v and vector are different" << std::endl;
154      ++numbad;
155   }
156   if( v4 != vectorcopy ) {
157      std::cout << "vectors v and vectorcopy are different" << std::endl;
158      ++numbad;
159   }
160
161   return numbad;
162 }
```

11.18 testStreamIO.cc.in

Use streaming IO to read and write a file

```
1
2 // testStreamIO.cc.in
3 //
4 // garren@fnal.gov, March 2006
5 //
6 // The same as testHepMC, but using the IO stream directly
8 //
9
10 #include <fstream>
11
12 #include "HepMC/GenEvent.h"
13 #include "HepMC/IO_AsciiParticles.h"
14 #ifdef HEPMC_HAS_IO_GENEVENT
15 #include "HepMC/IO_GenEvent.h"
16 #endif
17 #include "HepMC/Version.h"
18 #include "HepMC/IO_Exception.h"
19
20 // define methods and classes used by this test
21 #include "IsGoodEvent.h"
22 #include "testHepMCMethods.h"
23
24 void read_testIOGenEvent(std::ostream & os);
25 void read_variousFormats(std::ostream & os);
26 void write_to_stream(std::ostream & os);
27 void write_to_stream3(std::ostream & os);
28 void read_from_stream4(std::ostream & os);
29
30 int main() {
31     std::ofstream os( "testStreamIO.cout" );
32     std::ofstream osv( "testStreamIOVarious.cout" );
33     write_to_stream(os);
34     read_testIOGenEvent(os);
35     read_variousFormats(osv);
36     write_to_stream3(os);
37     read_from_stream4(os);
38     return 0;
39 }
40
41 void write_to_stream(std::ostream & os)
42 {
43     os << std::endl;
44     os << "basic IO_GenEvent input with streaming output" << std::endl;
45     // declare an input strategy to read the data produced with the
46     // example_MyPythia - units are GeV and mm
47     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
48     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
49     // declare an output stream
50     const char outfile[] = "testStreamIO.out";
51     std::ofstream ascii_out( outfile );
52     if( !ascii_out ) {
53       std::cerr << "cannot open " << outfile << std::endl;
54       exit(-1);
55     }
56     ascii_out.precision(16);
57     HepMC::write_HepMC_IO_block_begin( ascii_out );
58     // declare an instance of the event selection predicate
59     IsGoodEvent is_good_event;
60     //.....................................EVENT LOOP
61     int icount=0;
62     int num_good_events=0;
63     HepMC::GenEvent* evt = ascii_in.read_next_event();
```

```
64     while ( evt ) {
65         icount++;
66         if ( icount%50==1 ) os << "Processing Event Number " << icount
67                                << " its # " << evt->event_number()
68                                << std::endl;
69         if ( is_good_event(evt) ) {
70             ++num_good_events;
71             particleTypes( evt, os );
72             ascii_out << (*evt);
73         }
74 
75         // clean up and get next event
76         delete evt;
77         ascii_in >> evt;
78     }
79     HepMC::write_HepMC_IO_block_end( ascii_out );
80     //....................................PRINT RESULT
81     os << num_good_events << " out of " << icount
82               << " processed events passed the cuts. Finished." << std::endl;
83 }
84 
85 void read_testIOGenEvent(std::ostream & os)
86 {
87     os << std::endl;
88     os << "streaming input and output" << std::endl;
89     // input units are GeV and mm
90     const char infile[] = "@srcdir@/testIOGenEvent.input";
91     std::ifstream is( infile );
92     if( !is ) {
93       std::cerr << "cannot open " << infile << std::endl;
94       exit(-1);
95     }
96     // declare an output stream
97     const char outfile[] = "testStreamIO2.out";
98     std::ofstream ascii_out( outfile );
99     if( !ascii_out ) {
100       std::cerr << "cannot open " << outfile << std::endl;
101       exit(-1);
102     }
103     ascii_out.precision(16);
104     HepMC::write_HepMC_IO_block_begin( ascii_out );
105     // declare another output stream to test precision
106     const char poutfile[] = "testStreamIOprecision.out";
107     std::ofstream pout( poutfile );
108     if( !pout ) {
109       std::cerr << "cannot open " << poutfile << std::endl;
110       exit(-1);
111     }
112     pout.precision(10);
113     // declare an IO_AsciiParticle for output
114     HepMC::IO_AsciiParticles particle_out("testStreamIOParticle.out",std::ios::out);
115     // declare an instance of the event selection predicate
116     IsGoodEvent is_good_event;
117     //....................................EVENT LOOP
118     int icount=0;
119     int num_good_events=0;
120     HepMC::GenEvent evt;
121     while ( is ) {
122         // WARNING - we are not using pointers, so this could be an empty event
123         is >> evt;
124         // make sure this is a valid event
125         if( evt.is_valid() ) {
126             ++icount;
127             if ( icount%50==1 ) os << "Processing Event Number " << icount
128                                    << " its # " << evt.event_number()
129                                    << std::endl;
130             if ( is_good_event( &evt ) ) {
```

```
131                 ++num_good_events;
132                 particleTypes(&evt,os);
133                 ascii_out << evt;
134                 pout << evt;
135                 // We must explicitly create the pointer if we want to use this event
136                 // with any IO strategy (e.g., IO_AsciiParticles)
137                 HepMC::GenEvent* pevt= &evt;
138                 particle_out << pevt;
139             }
140         }
141     }
142     HepMC::write_HepMC_IO_block_end( ascii_out );
143     //.....................................PRINT RESULT
144     os << num_good_events << " out of " << icount
145               << " processed events passed the cuts. Finished." << std::endl;
146 }
147
148 void read_variousFormats(std::ostream & os)
149 {
150     os << std::endl;
151     os << "process varied input" << std::endl;
152     // declare an input stream
153     const char infile[] = "@srcdir@/testHepMCVarious.input";
154     std::ifstream is( infile );
155     if( !is ) {
156       std::cerr << "cannot open " << infile << std::endl;
157       exit(-1);
158     }
159     // set input units
160     HepMC::set_input_units( is, HepMC::Units::GEV, HepMC::Units::MM );
161     // declare an output stream
162     const char outfile[] = "testStreamIOVarious.out";
163     std::ofstream ascii_out( outfile );
164     if( !ascii_out ) {
165       std::cerr << "cannot open " << outfile << std::endl;
166       exit(-1);
167     }
168     ascii_out.precision(16);
169     HepMC::write_HepMC_IO_block_begin( ascii_out );
170     //.....................................EVENT LOOP
171     int icount=0, ibad=0;
172     HepMC::GenEvent evt;
173     while ( is ) {
174         // we have to do our own try/catch blocks
175         try {
176            is >> evt;
177         }
178         catch (HepMC::IO_Exception& e) {
179             evt.clear();
180             ++ibad;
181         }
182         // WARNING - we are not using pointers, so this could be an empty event
183         // make sure this is a valid event
184         if( evt.is_valid() ) {
185             icount++;
186             double pim;
187             os << "Processing Event Number " << icount
188                       << " its # " << evt.event_number()
189                       << std::endl;
190             ascii_out << evt;
191             // units should be unknown
192             evt.write_units(os);
193             pim = findPiZero(&evt);
194             os << " pizero mass: " << pim << std::endl;
195             // set units to GeV and mm
196             evt.use_units(HepMC::Units::GEV, HepMC::Units::MM);
197             evt.write_units(os);
```

```
198             pim = findPiZero(&evt);
199             os << " pizero mass: " << pim
200                      << " " << HepMC::Units::name( evt.momentum_unit() ) << std::endl;
201             // convert units to MeV
202             evt.use_units(HepMC::Units::MEV, HepMC::Units::MM);
203             evt.write_units(os);
204             pim = findPiZero(&evt);
205             os << " pizero mass: " << pim
206                      << " " << HepMC::Units::name( evt.momentum_unit() ) << std::endl;
207         }
208     }
209     std::cout << "testSteamIO: the HeavyIon and PdfInfo input stream errors are intentional" << std::e
210     HepMC::write_HepMC_IO_block_end( ascii_out );
211     //.....................................PRINT RESULT
212     os << icount << " valid events processed.  " ;
213     os << ibad << " invalid events processed. Finished." << std::endl;
214 }
215
216 void write_to_stream3(std::ostream & os)
217 {
218     os << std::endl;
219     os << "basic IO_GenEvent input with streaming output using member function" << std::endl;
220     // declare an input strategy to read the data produced with the
221     // example_MyPythia - units are GeV and mm
222     HepMC::IO_GenEvent ascii_in("@srcdir@/testIOGenEvent.input",std::ios::in);
223     ascii_in.use_input_units( HepMC::Units::GEV, HepMC::Units::MM );
224     // declare an output stream
225     const char outfile[] = "testStreamIO3.out";
226     std::ofstream ascii_out( outfile );
227     if( !ascii_out ) {
228       std::cerr << "cannot open " << outfile << std::endl;
229       exit(-1);
230     }
231     ascii_out.precision(16);
232     HepMC::write_HepMC_IO_block_begin( ascii_out );
233     // declare an instance of the event selection predicate
234     IsGoodEvent is_good_event;
235     //.....................................EVENT LOOP
236     int icount=0;
237     int num_good_events=0;
238     HepMC::GenEvent* evt = ascii_in.read_next_event();
239     while ( evt ) {
240         icount++;
241         if ( icount%50==1 ) os << "Processing Event Number " << icount
242                                      << " its # " << evt->event_number()
243                                      << std::endl;
244         if ( is_good_event(evt) ) {
245             ++num_good_events;
246             particleTypes( evt, os );
247             evt->write(ascii_out);
248         }
249
250         // clean up and get next event
251         delete evt;
252         ascii_in >> evt;
253     }
254     HepMC::write_HepMC_IO_block_end( ascii_out );
255     //.....................................PRINT RESULT
256     os << num_good_events << " out of " << icount
257               << " processed events passed the cuts. Finished." << std::endl;
258 }
259
260 void read_from_stream4(std::ostream & os)
261 {
262     os << std::endl;
263     os << "streaming input and output using member functions" << std::endl;
264     // input units are GeV and mm
```

```
265     const char infile[] = "@srcdir@/testIOGenEvent.input";
266     std::ifstream is( infile );
267     if( !is ) {
268       std::cerr << "cannot open " << infile << std::endl;
269       exit(-1);
270     }
271     // declare an output stream
272     const char outfile[] = "testStreamIO4.out";
273     std::ofstream ascii_out( outfile );
274     if( !ascii_out ) {
275       std::cerr << "cannot open " << outfile << std::endl;
276       exit(-1);
277     }
278     ascii_out.precision(16);
279     HepMC::write_HepMC_IO_block_begin( ascii_out );
280     // declare an instance of the event selection predicate
281     IsGoodEvent is_good_event;
282     //.....................................EVENT LOOP
283     int icount=0;
284     int num_good_events=0;
285     HepMC::GenEvent evt;
286     while ( is ) {
287         // WARNING - we are not using pointers, so this could be an empty event
288         evt.read(is);
289         // make sure this is a valid event
290         if( evt.is_valid() ) {
291             ++icount;
292             if ( icount%50==1 ) os << "Processing Event Number " << icount
293                                    << " its # " << evt.event_number()
294                                    << std::endl;
295             if ( is_good_event( &evt ) ) {
296                 ++num_good_events;
297                 particleTypes(&evt,os);
298                 evt.write(ascii_out);
299             }
300         }
301     }
302     HepMC::write_HepMC_IO_block_end( ascii_out );
303     //....................................PRINT RESULT
304     os << num_good_events << " out of " << icount
305                << " processed events passed the cuts. Finished." << std::endl;
306 }
```

11.19 testUnits.cc

Test MomentumUnits and PositionUnits Make sure set and change methods work as expected.

```
1 //
2 // Test Units
3 //
4 #include <iostream>
5
6 #include "HepMC/Units.h"
7
8 int main()
9 {
10
11   int err = 0;
12   double cf;
13
14   std::cout << "Default units: " << HepMC::Units::name(HepMC::Units::default_momentum_unit())
15      << " " << HepMC::Units::name(HepMC::Units::default_length_unit()) << std::endl;
16
17   // check momentum conversion factors
18   cf = conversion_factor( HepMC::Units::GEV, HepMC::Units::GEV );
19   if( cf != 1 ) {
20       ++err;
21       std::cerr << "wrong conversion factor " << cf
22                 << " for GEV to GEV - should be 1 \n";
23   }
24   cf = conversion_factor( HepMC::Units::MEV, HepMC::Units::MEV );
25   if( cf != 1 ) {
26       ++err;
27       std::cerr << "wrong conversion factor " << cf
28                 << " for MEV to MEV - should be 1 \n";
29   }
30   cf = conversion_factor( HepMC::Units::MEV, HepMC::Units::GEV );
31   if( cf != 0.001 ) {
32       ++err;
33       std::cerr << "wrong conversion factor " << cf
34                 << " for MEV to GEV - should be 0.001 \n";
35   }
36   cf = conversion_factor( HepMC::Units::GEV, HepMC::Units::MEV );
37   if( cf != 1000.0 ) {
38       ++err;
39       std::cerr << "wrong conversion factor " << cf
40                 << " for GEV to MEV - should be 1000 \n";
41   }
42
43   // check length conversion factors
44   cf = conversion_factor( HepMC::Units::MM, HepMC::Units::MM );
45   if( cf != 1 ) {
46       ++err;
47       std::cerr << "wrong conversion factor " << cf
48                 << " for MM to MM - should be 1 \n";
49   }
50   cf = conversion_factor( HepMC::Units::CM, HepMC::Units::CM );
51   if( cf != 1 ) {
52       ++err;
53       std::cerr << "wrong conversion factor " << cf
54                 << " for CM to CM - should be 1 \n";
55   }
56   cf = conversion_factor( HepMC::Units::CM, HepMC::Units::MM );
57   if( cf != 10.0 ) {
58       ++err;
59       std::cerr << "wrong conversion factor " << cf
60                 << " for CM to MM - should be 10 \n";
61   }
```

```
62   cf = conversion_factor( HepMC::Units::MM, HepMC::Units::CM );
63   if( cf != 0.1 ) {
64       ++err;
65       std::cerr << "wrong conversion factor " << cf
66                 << " for MM to CM - should be 0.1 \n";
67   }
68
69   return err;
70 }
```

11.20 VectorConversion.h

This example converts from ThreeVector and FourVector to CLHEP::Hep3Vector and CLHEP::HepLorentzVector Similar (or perhaps templated) conversion methods could be added to any vector class.

```
1 #ifndef VECTOR_CONVERSION_H
2 #define VECTOR_CONVERSION_H
4 // garren@fnal.gov, January 2007
5 //
6 // This example converts from ThreeVector and FourVector to
7 // CLHEP::Hep3Vector and CLHEP::HepLorentzVector
8 // Similar (or perhaps templated) conversion methods could be added to
9 // any vector class.
10 //
12
13 #include "HepMC/SimpleVector.h"
14 #include "CLHEP/Vector/LorentzVector.h"
19
21 inline CLHEP::Hep3Vector convertTo( const HepMC::ThreeVector& v )
22      { return CLHEP::Hep3Vector( v.x(), v.y(), v.z() ); }
23
25 inline CLHEP::HepLorentzVector convertTo( const HepMC::FourVector& v )
26      { return CLHEP::HepLorentzVector( v.x(), v.y(), v.z(), v.t() ); }
27
28 #endif  // VECTOR_CONVERSION_H
```

Chapter 12

HepMC Page Documentation

12.1 Todo List

Member filterEvent (p. **283**) `Have to build a list, since the GV::add_particle_out method modifies the end vertex!`

Member filterEvent (p. **283**) `Why does this cause an error?`

Index