SFH Reference Manual

Generated by Doxygen 1.3.2

Thu Oct 26 12:52:55 2006

Contents

Chapter 1

Self-Filling Histograms

By

Benno List

and

Jenny List, University of Hamburg

1.1 How Do I Get the Code?

The code is located in a cvs repository at DESY afs. To check it out, do

```
cvs -d /afs/desy.de/user/b/blist/.cvs co jbltools/sfh
```

1.2 What are Self-Filling Histograms?

What we call "Self-Filling Histograms" are histograms, derived from `root` histograms, where one can define during booking not only name, title, and binning of a histogram, but also what is plotted and under which conditions.

Consider this example:

```
SFH1F *h1 =
  new SFH1F ("JPsiElasticTT",
             "#muODS: m_{J/#Psi}: elastic production, (#mu-#mu)",
             40, 2., 4.,
             *this,
             jpsiMass,
             jpsiElastic&&jpsiTrackTrack&&jpsiNoCosmic&&jpsi2mu);
```

Without going into the details yet it is certainly plausible that this should be a histogram with name "JPsiElasticTT" and title "μODS: $m_{J/\Psi}$: elastic production, $(\mu-\mu)$", with 40 bins between 2 and 4 GeV, which shows the mass of J/ψ meson candidates, after cuts have been applied that demand:

- The J/ has been produced elastically.
- The J/ has 2 tracks.

- It's not a cosmic.
- It decays to 2 muons.

The classes contained in the package **SFH** provide the means to write this type of code.

At the core of our toolkit are, obviously, the self-filling histogram classes, which are called **SFH1F**, **SFH2F**, etc, similarly to the corresponding root classes TH1F, TH2F. etc. from which they are derived.

1.2.1 What do we mean by self-filling?

The main point is that a self-filling histogram implements a method **SFH1F::Fill()**, which in contrast to the method of `Int_t TH1F::Fill(Axis_t x)` does not take an argument.

When we loop over the entries of a root Tree, the `Fill()` method has to be called once per entry. If we have a list of all self-filling histograms, which is provided by class HList, this is easily done in a single call, which in turn calls all `Fill()` methods of the histograms.

1.2.2 How does the histogram know what to fill?

The secret lies in the objects `jpsiMass`, `jpsiElastic` etc, which are declared as follows (at the moment we assume there is only one, and exactly one, J/psi candidate per Tree entry):

```
FloatFun& jpsiMass       = *new JpsiMassFun;

BaseCut&  jpsiElastic    = *new JpsiElasticCut;
BaseCut&  jpsiTrackTrack = *new JpsiTTCut;
BaseCut&  jpsiNoCosmic   = *new JpsiNoCosmicCut;
BaseCut&  jpsi2mu        = *new Jpsi2MuCut;
```

So, `jpsiMass` is an object of type JpsiMassFun.

Assuming we have a ntuple declared in an include file NTuple.h, automatically generated by root, and this ntuple has a member jpsimass, we could have this code:

```
#include "Ntuple.h"
#include "jbltools/sfh/FloatFun.h"

external Ntuple* ntuple; // global pointer to ntuple

class JpsiMassFun : public FloatFun {
  public:
    virtual float operator() () const {
      return nt->jpsimass;
    }
};
```

So, JpsiMassFun is a class that implements `operator()`, which, when called, returns the jpsi mass stored in the ntuple row that is currently in memory.

This class is so light-weight that it can be completely implemented within a header file, without need for a separate implementation file.

Consequently, the base class **FloatFun** is an abstract base class which does not much more than declare that any **FloatFun** subclass must implement operator():

```
class FloatFun {
  public:
    virtual float operator() () const = 0;
  protected:
    virtual ~FloatFun() {};
};
```

(Why the destructor is protected is an issue that will be covered later.)

1.2.3 How does the histogram know how to cut?

By now, it should be almost obvious how the cuts are implemented: They are derived from an abstract base class **BaseCut**:

```
class BaseCut {
  public:
    virtual bool operator() () const = 0;
  protected:
    virtual ~BaseCut() {};
};
```

The only difference between **FloatFun** and **BaseCut** is that a BaseCut::operator() returns a bool, not a float.

So, to implement one cut we could write:

```
#include "Ntuple.h"
#include "jbltools/sfh/BaseCut.h"

external Ntuple* ntuple; // global pointer to ntuple

class Jpsi2MuCut : public BaseCut {
  public:
    virtual bool operator() () const {
      return nt->trackIsMuon[0] && nt->trackIsMuon[1];
    }
};
```

1.2.4 How can we form logical expressions of cuts?

Actually, this is very simple. Essentially, the code looks like this:

```
class AndOfTwoCuts: public BaseCut {
  public:
    AndOfTwoCuts (const BaseCut& lhs_, const BaseCut& rhs_)
    : lhs (&lhs_), rhs (&rhs_) {};

    virtual bool operator() () const {return (*lhs)() && (*rhs)();}
  protected:
    const BaseCut *lhs;
    const BaseCut *rhs;
};

inline AndOfTwoCuts& operator&& (const BaseCut& lhs_, const BaseCut& rhs_) {
  return *new AndOfTwoCuts (lhs_, rhs_);
}
```

(The real code, residing in **BaseCut.h**, is slightly more complicated, but not much.)

We first define a class AndOfTwoCuts that holds pointers to two other **BaseCut** objects, and returns the AND of their results. Additionally we overload **operator&&()**, so that it generates a new AndOfTwoCuts object, which we can pass (by reference!) on to a **SFH1F**. Voilà!

1.3 How can I Use these Self-Filling Histograms?

Typically, one creates one's own class that is derived from class **EventLoop**, let's name it AnalysisLoop.

The purpose of an AnalysisLoop object is to book, fill, and write out a number of histograms. It might look like this:

```
#include "jbltools/sfh/EventLoop.h"
#include "jbltools/sfh/Binning.h"
#include "jbltools/sfh/SFH1F.h"
#include "PTMissFun.h"
class AnalysisLoop: public EventLoop {
  public:
    // Constructor books histograms
    AnalysisLoop (Ntuple* ntuple)
    {
      Binning ptmissbinning (50, 0., 100.);
      FloatFun& ptmiss = *new PTMissFun (ntuple);
      // self-filling histogram for missing pt
      h = new SFH1F ("ptmiss", "Missing pt", ptmissbinning, this, ptmiss);

    }

    // Destructor does nothing
    virtual ~AnalysisLoop ()
    {}

    // output method writes histos to postscript and file
    virtual void output (const char* rootfile = "",
                         const char* psfile = "")
    {
      // create canvas
      TCanvas *canvas = new TCanvas("canvas", "ptmiss", 600, 800);
      // Write plot to psfile
      TPostScript ps (psfile, 111);
      canvas->Clear();
      h->Draw ("E0");                // Draws the histo
      canvas->Update();
      ps.Close();
      // Write histogram to file
      TFile file (rootfile, "RECREATE");
      h->Write();                    // Writes out the histo
      file.Write();
      file.Close();
    }
  protected:
    SFH1F *h;                        // The self-filling histogram
};
```

The corresponding main program opens an ntuple and loops over its entries, calling the (inherited) loop() method of the AnalysisLoop object for each row:

```
#define Ntuple_cxx
#include "Ntuple.h"                            // The ntuple
#include "AnalysisLoop.h"                      // The AnalysisLoop
int main(int argc, char** argv) {
  Ntuple nt;                                   // open ntuple
```

```
  TApplication theApp("main", &argc, argv);    // for graphics
  AnalysisLoop theAnalysisLoop (&nt);          // book histos
  // Loop over ntuple entries
  for (int i = 0; i < int(nt.fChain->GetEntriesFast()); ++i) {
    if (nt.LoadTree (i) < 0) break;
    nt.fChain->GetEntry(i);
    theAnalysisLoop.loop();                    // fill the histos
  }
  theAnalysisLoop.output("out.root","out.ps"); // store the histos
  return 0;
}
```

Here we use the headerfile Ntuple.h, generated by TTree::MakeClass, that describes the ntuple structure.

AnalysisLoop is derived from **EventLoop**, and thereby from **SFROList** (list of self-filling registered objects). **SFROList** defines a Fill() method that fills all self-filling objects by calling their respective Fill() method. **EventLoop::loop()** calls this Fill() method, and therefore fills all self-filling objects that have registered themselves with the loop, in our example the ptmiss-histogram.

Chapter 2

SFH Hierarchical Index

2.1 SFH Class Hierarchy

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

Chapter 3

SFH Compound Index

3.1 SFH Compound List

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

Chapter 4

SFH File Index

4.1 SFH File List

Here is a list of all documented files with brief descriptions:

Chapter 5

SFH Page Index

5.1 SFH Related Pages

Here is a list of all related documentation pages:

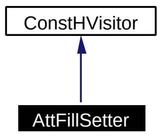
Chapter 6

SFH Class Documentation

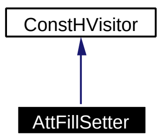
6.1 AttFillSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AttFillSetter:



Collaboration diagram for AttFillSetter:



6.1.1 Detailed Description

An **HVisitor** that sets the Fill attributes fcolor, and fstyle.

See root class TAttFill for a documentation of these attributes.

All root histogram classes are derived from TAttFill.

Definition at line 168 of file HVisitors.h.

Public Member Functions

- **AttFillSetter** (Color_t fcolor_=-1, Style_t fstyle_=-1)

    Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- Color_t **fcolor**

  The new color; -1: dont set it.

- Style_t **fstyle**

  The new style; -1: dont set it.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AttFillSetter::AttFillSetter (Color_t *fcolor_* = -1, Style_t *fstyle_* = -1) `[inline]`

Constructor.

Parameters:

fcolor_ The new color; -1: dont set it

fstyle_ The new style; -1: dont set it

Definition at line 171 of file HVisitors.h.

References fcolor, and fstyle.

6.1.3 Member Function Documentation

6.1.3.1 virtual void AttFillSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 177 of file HVisitors.h.

References fcolor, and fstyle.

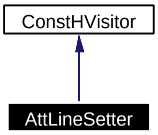
The documentation for this class was generated from the following file:

- **HVisitors.h**

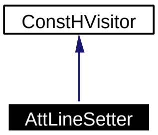
6.2 AttLineSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AttLineSetter:



Collaboration diagram for AttLineSetter:



6.2.1 Detailed Description

An **HVisitor** that sets the line attributes lcolor, lstyle, and lwidth.

See root class TAttLine for a documentation of these attributes.

All root histogram classes are derived from TAttLine.

Definition at line 130 of file HVisitors.h.

Public Member Functions

- **AttLineSetter** (Color_t lcolor_=-1, Style_t lstyle_=-1, Width_t lwidth_=-1)
  Constructor.
- virtual void **visit** (**RegO** &h) const
  The visit function.
- virtual void **visit** (TLine &h) const

Private Attributes

- Color_t **lcolor**
  The new color; -1: dont set it.
- Style_t **lstyle**
  The new style; -1: dont set it.
- Width_t **lwidth**
  The new width; -1: dont set it.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AttLineSetter::AttLineSetter (Color_t *lcolor_* = -1, Style_t *lstyle_* = -1, Width_t *lwidth_* = -1) `[inline]`

Constructor.

Parameters:

*lcolor_* The new color; -1: dont set it

*lstyle_* The new style; -1: dont set it

*lwidth_* The new width; -1: dont set it

Definition at line 133 of file HVisitors.h.

References lcolor, lstyle, and lwidth.

6.2.3 Member Function Documentation

6.2.3.1 virtual void AttLineSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 140 of file HVisitors.h.

References lcolor, lstyle, and lwidth.

6.2.3.2 virtual void AttLineSetter::visit (TLine & *h*) const `[inline, virtual]`

Parameters:

h The object to be visited

Definition at line 148 of file HVisitors.h.

References lcolor, lstyle, and lwidth.

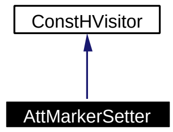
The documentation for this class was generated from the following file:

- **HVisitors.h**

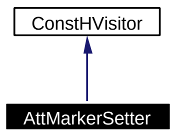
6.3 AttMarkerSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AttMarkerSetter:



Collaboration diagram for AttMarkerSetter:



6.3.1 Detailed Description

An **HVisitor** that sets the marker attributes color, size, and style.

See root class TAttMarker for a documentation of these attributes All root histogram classes are derived from TAttMarker

Definition at line 101 of file HVisitors.h.

Public Member Functions

- **AttMarkerSetter** (Color_t tcolor_=-1, Size_t msize_=-1, Style_t mstyle_=-1)

  Constructor.

- virtual void **visit** (**RegO** &h) const

Private Attributes

- Color_t **tcolor**

  The new color; -1: dont set it.

- Size_t **msize**

  The new size; -1: dont set it.

- Style_t **mstyle**

  The new style; -1: dont set it.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AttMarkerSetter::AttMarkerSetter (Color_t *tcolor_* = -1, Size_t *msize_* = -1, Style_t *mstyle_* = -1) `[inline]`

Constructor.

Parameters:

tcolor_ The new color; -1: dont set it

msize_ The new size; -1: dont set it

mstyle_ The new style; -1: dont set it

Definition at line 104 of file HVisitors.h.

References msize, mstyle, and tcolor.

6.3.3 Member Function Documentation

6.3.3.1 virtual void AttMarkerSetter::visit (RegO & *h*) const `[inline, virtual]`

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 110 of file HVisitors.h.

References msize, mstyle, and tcolor.

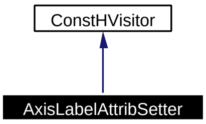
The documentation for this class was generated from the following file:

- **HVisitors.h**

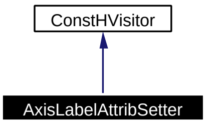
6.4 AxisLabelAttribSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AxisLabelAttribSetter:



Collaboration diagram for AxisLabelAttribSetter:



6.4.1 Detailed Description

An **HVisitor** that sets the axis attribues name, color, style, and number of divisions.

See root class TAxis for a documentation of these attributes

Definition at line 294 of file HVisitors.h.

Public Member Functions

- **AxisLabelAttribSetter** (int axis_, Double_t size_, Color_t lcolor_=-1, Style_t font_=-1, Int_t ndiv_=-1)

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Member Functions

- **AxisLabelAttribSetter** (const **AxisLabelAttribSetter** &source)

  Private copy constructor, left unimplemented: No copying!

- **AxisLabelAttribSetter** & **operator=** (const **AxisLabelAttribSetter** &source)

  Private assignment operator, left unimplemented: No assignment!

Private Attributes

- int **axis**

The axis (1=x, 2=y, 3=z).

- Double_t **size**

  The new size (<0: dont set).

- Color_t **lcolor**

  The new color (-1: dont set).

- Style_t **font**

  The new label font (-1: dont set).

- Int_t **ndiv**

  The new number of divisions (-1: dont set).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AxisLabelAttribSetter::AxisLabelAttribSetter (int *axis_*, Double_t *size_*, Color_t *lcolor_* = -1, Style_t *font_* = -1, Int_t *ndiv_* = -1) `[inline]`

Constructor.

Parameters:

axis_ The axis (1=x, 2=y, 3=z)

size_ The new size (<0: dont set)

lcolor_ The new color (-1: dont set)

font_ The new label font (-1: dont set)

ndiv_ The new number of divisions (-1: dont set)

Definition at line 297 of file HVisitors.h.

References axis, font, lcolor, ndiv, and size.

6.4.3 Member Function Documentation

6.4.3.1 virtual void AxisLabelAttribSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 310 of file HVisitors.h.

References axis, font, lcolor, ndiv, and size.

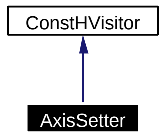
The documentation for this class was generated from the following file:

- **HVisitors.h**

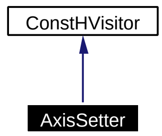
6.5 AxisSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AxisSetter:



Collaboration diagram for AxisSetter:



6.5.1 Detailed Description

An **HVisitor** that sets the x axis of a histogram to a given binning.

Definition at line 397 of file HVisitors.h.

Public Member Functions

- **AxisSetter** (Int_t n_, const Float_t *x_)

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- Int_t **n**
- const Float_t * **x**

6.5.2 Member Function Documentation

6.5.2.1 virtual void AxisSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 404 of file HVisitors.h.

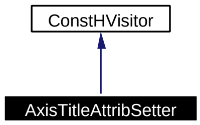
The documentation for this class was generated from the following file:

- **HVisitors.h**

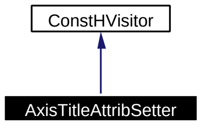
6.6 AxisTitleAttribSetter Class Reference

`#include <HVisitors.h>`

Inheritance diagram for AxisTitleAttribSetter:



Collaboration diagram for AxisTitleAttribSetter:



6.6.1 Detailed Description

An **HVisitor** that sets the axis attribues name, color, style, and number of divisions.

See root class TAxis for a documentation of these attributes

Definition at line 230 of file HVisitors.h.

Public Member Functions

- **AxisTitleAttribSetter** (int axis_, const char ∗name_=0, Double_t size_=-1, Double_t offset_=-999., Color_t lcolor_=-1, Style_t font_=-1)

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Member Functions

- **AxisTitleAttribSetter** (const **AxisTitleAttribSetter** &source)

  Private copy constructor, left unimplemented: No copying!

- **AxisTitleAttribSetter** & **operator=** (const **AxisTitleAttribSetter** &source)

  Private assignment operator, left unimplemented: No assignment!

Private Attributes

- int **axis**

  The axis (1=x, 2=y, 3=z).

- char * **name**

  The new name (0: dont set).

- Double_t **size**

  The new size (<0: dont set).

- Double_t **offset**

  The new offset (=-999: dont set).

- Color_t **lcolor**

  The new color (-1: dont set).

- Style_t **font**

  The new label font (-1: dont set).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AxisTitleAttribSetter::AxisTitleAttribSetter (int *axis_*, const char * *name_* = 0, Double_t *size_* = -1, Double_t *offset_* = -999., Color_t *lcolor_* = -1, Style_t *font_* = -1) `[inline]`

Constructor.

Parameters:

axis_ The axis (1=x, 2=y, 3=z)

name_ The new title (0: dont set)

size_ The new size (<0: dont set)

offset_ The new size (-999: dont set)

lcolor_ The new color (-1: dont set)

font_ The new number of divisions (-1: dont set)

Definition at line 233 of file HVisitors.h.

References axis, font, lcolor, name, offset, and size.

6.6.3 Member Function Documentation

6.6.3.1 virtual void AxisTitleAttribSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 253 of file HVisitors.h.

References axis, font, lcolor, name, offset, and size.

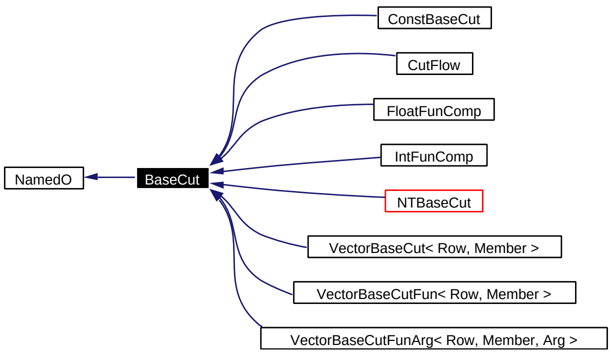
The documentation for this class was generated from the following file:

- **HVisitors.h**

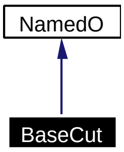
6.7 BaseCut Class Reference

`#include <BaseCutBase.h>`

Inheritance diagram for BaseCut:



Collaboration diagram for BaseCut:



6.7.1 Detailed Description

Abstract base class for function objects doing cuts.

Cut objects derived from BaseCut have to implement just the function operator(); this operator has to return a bool that indicates whether an event passes the cut or not.

Cut objects are mainly used by self-filling histograms.

The destructor of BaseCut is protected; therefore, BaseCut objects cannot be created as automatic variables, only on the heap. This ensures that all BaseCut objects live until the end of the program run, otherwise it would be very hard to ensure that a self-filling histogram, which stores only a pointer to a cut object, does not try to access a cut object that has already been destroyed. Purists can use the "destroy" method to delete a **FloatFun** object.

Cut objects can be combined with the usual logical operators !, &&, ||

Changelog:

- 15.7.04 BL: Added method **getIterator()**
- 29.5.05 BL: Derived from class **NamedO**: makes lookup according to name possible
- 2.6.05 BL: New constructors with a name

Author: Benno List, Jenny Boehme

Date
2005/06/06 15:01:42

Author
blist

Definition at line 51 of file BaseCutBase.h.

Public Member Functions

- **BaseCut** (const char *name_="?")
  Constructor from a C style string, serves as default constructor.
- **BaseCut** (const std::string &name_)
  Constructor from a C++ string.
- **BaseCut** (const std::string &name_, const std::string &definition_)
  Constructor from two C++ strings.
- virtual bool **operator()** () const=0
  Returns a bool value, typically the result of a cut on ntuple data.
- virtual **FloatFun** & **Float** () const
  Returns a **FloatFun** *object.*
- virtual **IntFun** & **Int** () const
  Returns a **IntFun** *object.*
- virtual void **destroy** ()
  Replaces delete, which cannot be called because of protected destructor.
- virtual const **FillIterator** * **getIterator** () const
  Returns pointer to iterator, if any.

Static Public Member Functions

- **BaseCut** * **getObject** (const std::string &**name**)
  Access function that returns a pointer to an **FloatFun** *object with this name.*
- **BaseCut** * **getObject** (const char ***name**)
  Access function that returns a pointer to an **NamedO** *object with this name.*

Protected Member Functions

- virtual ~**BaseCut** ()
  Protected destructor ensures creation on the heap.

Related Functions

(Note that these are not member functions.)

- **BaseCut** & **operator**|| (const **BaseCut** &lhs_, const **BaseCut** &rhs_)

  "Or" of two BaseCut objects

- **BaseCut** & **operator!** (const **BaseCut** &rhs_)

  Negation of a BaseCut.

- **FloatFun** & **Float** (const **BaseCut** &arg)

  Returns a **FloatFun** *object made out of the BaseCut object.*

- **IntFun** & **Int** (const **BaseCut** &arg)

  Returns a **IntFun** *object made out of the BaseCut object.*

- **BaseCut** & **cached** (const **ROListPoR** &rol, const **BaseCutPoR** &cut, const **FillIteratorPoR** &iter=0)

  Returns a cached version of a BaseCut.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 BaseCut::BaseCut (const char * *name_* = "?") `[inline, explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:

name_ The object's name

Definition at line 55 of file BaseCutBase.h.

6.7.2.2 BaseCut::BaseCut (const std::string & *name_*) `[inline, explicit]`

Constructor from a C++ string.

Parameters:

name_ The object's name

Definition at line 60 of file BaseCutBase.h.

6.7.2.3 BaseCut::BaseCut (const std::string & *name_*, const std::string & *definition_*) `[inline]`

Constructor from two C++ strings.

Parameters:

name_ The object's name

definition_ The object's definition

Definition at line 65 of file BaseCutBase.h.

6.7.3 Member Function Documentation

6.7.3.1 BaseCut* BaseCut::getObject (const std::string & *name*) [inline, static]

Access function that returns a pointer to an **FloatFun** object with this name.

This access function must never be called within the constructor of any class that is derived from BaseCut!

Parameters:
name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 87 of file BaseCutBase.h.

6.7.3.2 BaseCut* BaseCut::getObject (const char * *name*) [inline, static]

Access function that returns a pointer to an **NamedO** object with this name.

This access function must never be called within the constructor of any class that is derived from NamedO!

Parameters:
name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 97 of file BaseCutBase.h.

6.7.4 Friends And Related Function Documentation

6.7.4.1 BaseCut & cached (const ROListPoR & *rol*, const BaseCutPoR & *cut*, const FillIteratorPoR & *iter* = 0) [related]

Returns a cached version of a BaseCut.

Parameters:
rol Cached objects list where to register
cut A BaseCut
iter An optional iterator

Definition at line 170 of file BaseCut.C.

References getIterator(), BaseCutPoR::pbc, and FillIteratorPoR::pfi.

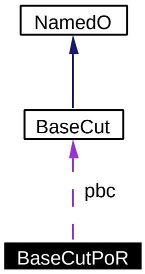
The documentation for this class was generated from the following files:

- **BaseCutBase.h**
- **BaseCut.C**
- **BaseCut.h**

6.8 BaseCutPoR Class Reference

`#include <BaseCutPoR.h>`

Collaboration diagram for BaseCutPoR:



6.8.1 Detailed Description

BaseCut Pointer or Reference.

Automagically converts between pointers and references to **BaseCut** objects

Author: Benno List, Jenny Böhme

Date

2005/06/29 14:40:18

Author

blist

Definition at line 26 of file BaseCutPoR.h.

Public Member Functions

- **BaseCutPoR** (**BaseCut** &bc_)

    Constructor from a reference.

- **BaseCutPoR** (**BaseCut** *pbc_)

    Constructor from a pointer.

- **BaseCutPoR** (int=0)

    Constructor from an int (will normally be 0!).

- **BaseCutPoR** (const std::string &name)

    Constructor from a name.

- **BaseCutPoR** (const char *name)

    Constructor from a name.

Public Attributes

- **BaseCut * pbc**

  A **BaseCut** *pointer.*

6.8.2 Constructor & Destructor Documentation

6.8.2.1 BaseCutPoR::BaseCutPoR (BaseCut & *bc_*) `[inline]`

Constructor from a reference.

Parameters:

bc_ A **BaseCut** reference

Definition at line 29 of file BaseCutPoR.h.

References pbc.

6.8.2.2 BaseCutPoR::BaseCutPoR (BaseCut * *pbc_*) `[inline]`

Constructor from a pointer.

Parameters:

pbc_ A **BaseCut** pointer

Definition at line 33 of file BaseCutPoR.h.

References pbc.

6.8.2.3 BaseCutPoR::BaseCutPoR (const std::string & *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 21 of file BaseCutPoR.C.

6.8.2.4 BaseCutPoR::BaseCutPoR (const char * *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 24 of file BaseCutPoR.C.

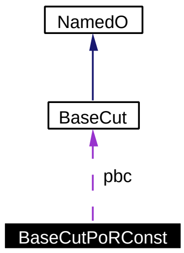
The documentation for this class was generated from the following files:

- **BaseCutPoR.h**
- **BaseCutPoR.C**

6.9 BaseCutPoRConst Class Reference

`#include <BaseCutPoR.h>`

Collaboration diagram for BaseCutPoRConst:



6.9.1 Detailed Description

const **BaseCut** Pointer or Reference

Automagically converts between pointers and references to const **BaseCut** objects

Author: Benno List, Jenny Böhme

Date
2005/06/29 14:40:18

Author
blist

Definition at line 62 of file BaseCutPoR.h.

Public Member Functions

- **BaseCutPoRConst** (const **BaseCut** &bc_)

  Constructor from a reference.

- **BaseCutPoRConst** (const **BaseCut** *pbc_)

  Constructor from a pointer.

- **BaseCutPoRConst** (int=0)

  Constructor from an int (will normally be 0!).

- **BaseCutPoRConst** (const std::string &name)

  Constructor from a name.

- **BaseCutPoRConst** (const char *name)

  Constructor from a name.

Public Attributes

- const **BaseCut** ∗ **pbc**

    A **BaseCut** *pointer.*

6.9.2 Constructor & Destructor Documentation

6.9.2.1 BaseCutPoRConst::BaseCutPoRConst (const BaseCut & *bc_*) `[inline]`

Constructor from a reference.

Parameters:

*bc_* A **BaseCut** reference

Definition at line 65 of file BaseCutPoR.h.

References pbc.

6.9.2.2 BaseCutPoRConst::BaseCutPoRConst (const BaseCut ∗ *pbc_*) `[inline]`

Constructor from a pointer.

Parameters:

*pbc_* A **BaseCut** pointer

Definition at line 69 of file BaseCutPoR.h.

References pbc.

6.9.2.3 BaseCutPoRConst::BaseCutPoRConst (const std::string & *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 27 of file BaseCutPoR.C.

6.9.2.4 BaseCutPoRConst::BaseCutPoRConst (const char ∗ *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 30 of file BaseCutPoR.C.

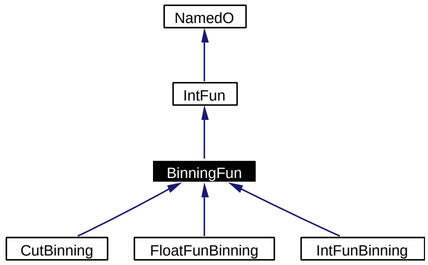
The documentation for this class was generated from the following files:

- **BaseCutPoR.h**
- **BaseCutPoR.C**

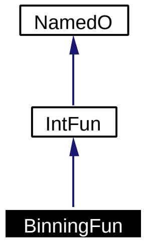
6.10 BinningFun Class Reference

`#include <BinningFun.h>`

Inheritance diagram for BinningFun:



Collaboration diagram for BinningFun:



6.10.1 Detailed Description

Abstract base class for function objects that sort events into bins.

The purpose of a BinningFun is to sort histogram entries into mutually exclusive bins (or non-exclusive bins, see blow). BinningFun objects are mainly used by SetOfHistogram and MatrixOfHistogram objects.

operator() has to return a value between -1 and n-1 (where n is the number of bins returned by getNBins), where -1 stands for "outside any bin" and the other values designate the bins.

The destructor is protected to ensure BinningFun objects can only be created on the heap (with "new"). The resulting memory leak is acknowledged.

getBinName() and **getBinTitle()** return strings that are used to automatically create histogram names (ids) and titles by SetOfHistogram and MatrixOfHistogram objects. The have to create new C-style strings that are deleted by the consumer.

A BinningFun can collaborate with a **FillIterator** object if more than one entry per event (e.g. if we have several D∗ mesons in one event).

The class **FloatFunBinning** can be used to generate a BinningFun in an easy way from a **FloatFun**, which delivers the value (e.g., t) that is used for binning, and some strings that are used to construct the name and title strings.

Exclusive and Non-exclusive Binnings

We distinguish between exclusive and non-exclusive binnings.

For an exclusive binning, an event (or particle) belongs at most to one bin. Therefore, in the Fill() method of a **SFSetOfHistograms** or **SFMatrixOfHistograms**, at most one histogram is filled. An exclusive binning is usefule e.g. to plot a quantity differentially in another quantity.

For an non-exclusive binning, an event may belong to several bins. In this case, the Fill() method of a **SFSetOfHistograms** or **SFMatrixOfHistograms** has to iterate over the possible values between 0 and getNBins()-1, and evaluate operator()(int) for each bin to see whether it should make an entry in the corrersponding histogram. Such non-exclusive binning are useful for cut-flow applications.

For non-exclusive binnings, **operator()()** makes really no sense. It must be implemented nevertheless.

Author: Jenny Böhme, Benno List

Date

2005/06/11 11:31:11

Author

blist

Changelog:

- 4.6.04 BL: Derive BiningFun from Binning
- 29.6.04 BL: Add constructor from Binning object
- 6.7.04 BL: Derive BinningFun also from **IntFun**
- 27.5.05 BL: Add **getMaxBins()**
- 7.6.05 BL: Added names
- 10.6.05 BL: Introduced non-exclusive binnings

Definition at line 75 of file BinningFun.h.

Public Member Functions

- **BinningFun** ()

  Default: Create an empty binning.
- **BinningFun** (int nbins, float xlow, float xup, const char ∗**name**=0)

  Create an equidistant binning.
- **BinningFun** (int nbins, float xlow, float xup, const std::string &**name**)

  Create an equidistant binning.
- **BinningFun** (int nbins, const float binedges[ ], const char ∗**name**=0)

  Create an nonequidistant binning, taking floats.
- **BinningFun** (int nbins, const float binedges[ ], const std::string &**name**)

  Create an nonequidistant binning, taking floats.
- **BinningFun** (int nbins, const double binedges[ ], const char ∗**name**=0)

Create an nonequidistant binning, taking doubles.

- **BinningFun** (int nbins, const double binedges[ ], const std::string &**name**)
  Create an nonequidistant binning, taking doubles.
- **BinningFun** (const Binning &binning_, const char ∗**name**=0)
  Create a binning from a Binning object.
- virtual int **operator()** () const=0
  Returns a bin number between -1 and getNBins()-1.
- virtual bool **operator()** (int i) const
  Returns whether an event belongs to bin i.
- virtual bool **isExclusive** () const
  Returns whether a BinningFun is exclusive.
- virtual const char ∗ **getBinName** (int i) const=0
  returns a name string.
- virtual const char ∗ **getBinTitle** (int i) const=0
  returns a title string.
- virtual int **getMaxBins** () const
  Gets the maximal return value of operator() plus 1, or the higest "sensible" value for a bin number.

Protected Member Functions

- virtual ∼**BinningFun** ()
  Protected destructor to ensure creation on the heap.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 BinningFun::BinningFun (int *nbins*, float *xlow*, float *xup*, const char ∗ *name* = 0) `[inline]`

Create an equidistant binning.

The binning will have nbins bins between xlow and xup.

Parameters:

nbins The number of bins

xlow The lowest bin edge

xup The highest bin edge

name The object name

Definition at line 83 of file BinningFun.h.

References NamedO::str().

6.10.2.2 BinningFun::BinningFun (int *nbins*, float *xlow*, float *xup*, const std::string & *name*) `[inline]`

Create an equidistant binning.

The binning will have nbins bins between xlow and xup.

Parameters:

nbins The number of bins

xlow The lowest bin edge

xup The highest bin edge

name The object name

Definition at line 95 of file BinningFun.h.

6.10.2.3 BinningFun::BinningFun (int *nbins*, const float *binedges*[ ], const char * *name* = 0) `[inline]`

Create an nonequidistant binning, taking floats.

The binning will have nbins bins with binedges given by binedges. binedges[i] and binedges[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges An array with binedges

name The object name

Definition at line 112 of file BinningFun.h.

References NamedO::str().

6.10.2.4 BinningFun::BinningFun (int *nbins*, const float *binedges*[ ], const std::string & *name*) `[inline]`

Create an nonequidistant binning, taking floats.

The binning will have nbins bins with binedges given by binedges. binedges[i] and binedges[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges An array with binedges

name The object name

Definition at line 128 of file BinningFun.h.

6.10.2.5 BinningFun::BinningFun (int *nbins*, const double *binedges*[ ], const char * *name* = 0) [inline]

Create an nonequidistant binning, taking doubles.

The binning will have nbins bins with binedges given by binedges. binedges[i] and binedges[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges An array with binedges

name The object name

Definition at line 145 of file BinningFun.h.

References NamedO::str().

6.10.2.6 BinningFun::BinningFun (int *nbins*, const double *binedges*[ ], const std::string & *name*) [inline]

Create an nonequidistant binning, taking doubles.

The binning will have nbins bins with binedges given by binedges. binedges[i] and binedges[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges An array with binedges

name The object name

Definition at line 162 of file BinningFun.h.

6.10.2.7 BinningFun::BinningFun (const Binning & *binning_*, const char * *name* = 0) [inline]

Create a binning from a Binning object.

Parameters:

binning_ The binning

name The object name

Definition at line 171 of file BinningFun.h.

References NamedO::str().

6.10.3 Member Function Documentation

6.10.3.1 virtual bool BinningFun::operator() (int *i*) const [inline, virtual]

Returns whether an event belongs to bin i.

The default implementation is valid only for exclusive binnings, i.e. binnings where an event belongs exactly to one bin

Parameters:
i The bin number

Reimplemented in **CutBinning** (p. 67).

Definition at line 191 of file BinningFun.h.

6.10.3.2 virtual const char∗ BinningFun::getBinName (int *i*) const `[pure virtual]`

returns a name string.

The name string MAY, and MUST, be destroyed by the user! therefore: you MUST NOT return a string constant here, like "dummy", but use new[] to allocate the string. The name string should be short, unique, and contain no spaces, like "tbin0".

Parameters:
i The bin number

Implemented in **CutBinning** (p. 67), **FloatFunBinning** (p. 98), and **IntFunBinning** (p. 128).

Referenced by SetOfH::genBinName(), and MatrixOfH::genBinName().

6.10.3.3 virtual const char∗ BinningFun::getBinTitle (int *i*) const `[pure virtual]`

returns a title string.

The name string MAY, and MUST, be destroyed by the user! therefore: you MUST NOT return a string constant here, like "dummy", but use new[] to allocate the string. The name string should describe the bin, like "0.1 < t < 0.5".

Parameters:
i The bin number

Implemented in **CutBinning** (p. 68), **FloatFunBinning** (p. 98), and **IntFunBinning** (p. 128).

Referenced by SetOfH::genBinTitle(), and MatrixOfH::genBinTitle().

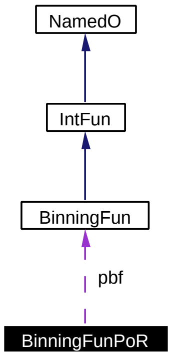
The documentation for this class was generated from the following file:

- **BinningFun.h**

6.11 BinningFunPoR Class Reference

`#include <BinningFunPoR.h>`

Collaboration diagram for BinningFunPoR:



6.11.1 Detailed Description

BinningFun Pointer or Reference.

Automagically converts between pointers and references to **BinningFun** objects

Author: Benno List, Jenny Böhme

Date

2004/06/29 13:03:38

Author

blist

Definition at line 24 of file BinningFunPoR.h.

Public Member Functions

- **BinningFunPoR** (**BinningFun** &bf_)

  Constructor from a reference.

- **BinningFunPoR** (**BinningFun** *pbf_=0)

  Constructor from a pointer.

Public Attributes

- **BinningFun** * **pbf**

  A **BinningFun** *pointer.*

6.11.2 Constructor & Destructor Documentation

6.11.2.1 BinningFunPoR::BinningFunPoR (BinningFun & *bf_*) `[inline]`

Constructor from a reference.

Parameters:
*bf_* A **BinningFun** reference

Definition at line 27 of file BinningFunPoR.h.

References pbf.

6.11.2.2 BinningFunPoR::BinningFunPoR (BinningFun * *pbf_* = 0) `[inline]`

Constructor from a pointer.

Parameters:
*pbf_* A **BinningFun** pointer

Definition at line 31 of file BinningFunPoR.h.

References pbf.

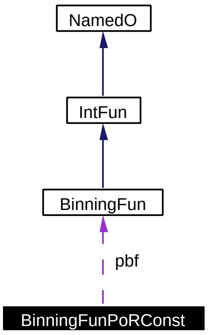
The documentation for this class was generated from the following file:

- **BinningFunPoR.h**

6.12 BinningFunPoRConst Class Reference

`#include <BinningFunPoR.h>`

Collaboration diagram for BinningFunPoRConst:



6.12.1 Detailed Description

const **BinningFun** Pointer or Reference

Automagically converts between pointers and references to const **BinningFun** objects

Author: Benno List, Jenny Böhme

Date
2004/06/29 13:03:38

Author
blist

Definition at line 49 of file BinningFunPoR.h.

Public Member Functions

- **BinningFunPoRConst** (const **BinningFun** &bf_)

    Constructor from a reference.

- **BinningFunPoRConst** (const **BinningFun** *pbf_=0)

    Constructor from a pointer.

Public Attributes

- const **BinningFun** * **pbf**

    A **BinningFun** *pointer.*

6.12.2 Constructor & Destructor Documentation

6.12.2.1 BinningFunPoRConst::BinningFunPoRConst (const BinningFun & *bf_*) [inline]

Constructor from a reference.

Parameters:
*bf_* A **BinningFun** reference

Definition at line 52 of file BinningFunPoR.h.

References pbf.

6.12.2.2 BinningFunPoRConst::BinningFunPoRConst (const BinningFun ∗ *pbf_* = 0) [inline]

Constructor from a pointer.

Parameters:
*pbf_* A **BinningFun** pointer

Definition at line 56 of file BinningFunPoR.h.

References pbf.

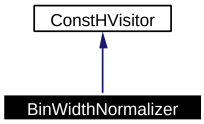
The documentation for this class was generated from the following file:

- **BinningFunPoR.h**

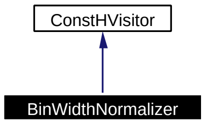
6.13 BinWidthNormalizer Class Reference

`#include <HVisitors.h>`

Inheritance diagram for BinWidthNormalizer:



Collaboration diagram for BinWidthNormalizer:



6.13.1 Detailed Description

An **HVisitor** that normalizes all TH1 objects to a constant bin width in x.

For histograms with unequidistant binning, this class can be used to normalize all bin contents such that they correspond to the number of entries per abscissa distance.

Definition at line 75 of file HVisitors.h.

Public Member Functions

- **BinWidthNormalizer** (Double_t width_=1)

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- Double_t **width**

  The nominal bin width.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 BinWidthNormalizer::BinWidthNormalizer (Double_t *width_* = 1) `[inline]`

Constructor.

Parameters:
*width_* The nominal bin width

Definition at line 78 of file HVisitors.h.

References width.

6.13.3 Member Function Documentation

6.13.3.1 virtual void BinWidthNormalizer::visit (RegO & *h*) const [inline, virtual]

The visit function.

Parameters:
h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 82 of file HVisitors.h.

References width.

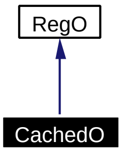
The documentation for this class was generated from the following file:

- **HVisitors.h**

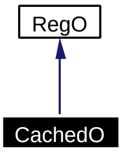
6.14 CachedO Class Reference

`#include <CachedO.h>`

Inheritance diagram for CachedO:



Collaboration diagram for CachedO:



6.14.1 Detailed Description

Mix-in class for cached, registered objects.

A CachedO object can register itself in a **ROList** object, which holds an array of pointers to **RegO** objects. It has an **invalidateCache()** method, that invalidates the cache.

Author: Benno List, Jenny Böhme

Date

2004/07/06 11:46:05

Author

blist

Changelog:

- 21.6.04 BL First version

Definition at line 29 of file CachedO.h.

Public Member Functions

- **CachedO** (const **ROListPoR** &rol)

  Constructor that registers this object in an **ROList** *object.*

- virtual ~**CachedO** ()

  Virtual destructor to ensure that dynamic casts work.

- virtual void **invalidateCache** ()=0

    The **invalidateCache()** *method tells the object that its cache is invalid.*

Private Member Functions

- **CachedO** (const **CachedO** &rhs)

    Private copy constructor, left unimplemented: No copying!

- **CachedO** & **operator=** (const **CachedO** &rhs)

    Private assignment operator, left unimplemented: No assignment!

6.14.2 Constructor & Destructor Documentation

6.14.2.1 CachedO::CachedO (const ROListPoR & *rol*)

Constructor that registers this object in an **ROList** object.

Parameters:

rol The **ROList** object where this is registered

Definition at line 22 of file CachedO.C.

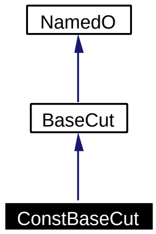
The documentation for this class was generated from the following files:

- **CachedO.h**
- **CachedO.C**

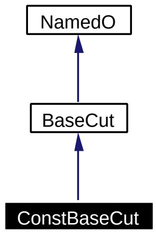
6.15 ConstBaseCut Class Reference

`#include <BaseCut.h>`

Inheritance diagram for ConstBaseCut:



Collaboration diagram for ConstBaseCut:



6.15.1 Detailed Description

A **IntFun** that returns a constant value.

Definition at line 36 of file BaseCut.h.

Public Member Functions

- **ConstBaseCut** (bool c_, const char *name_=0)

  Constructor from a C style string, serves as default constructor.

- **ConstBaseCut** (bool c_, const std::string &name_)

  Constructor from a C++ string.

- virtual bool **operator()** () const

  Returns a constant value c.

Protected Member Functions

- virtual ~**ConstBaseCut** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const bool **c**

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ConstBaseCut::ConstBaseCut (bool *c_*, const char * *name_* = 0) [inline, explicit]

Constructor from a C style string, serves as default constructor.

Parameters:

c_ The constant to be returned

name_ An optional name

Definition at line 39 of file BaseCut.h.

References c, and NamedO::str().

6.15.2.2 ConstBaseCut::ConstBaseCut (bool *c_*, const std::string & *name_*) [inline, explicit]

Constructor from a C++ string.

Parameters:

c_ The constant to be returned

name_ The object's name

Definition at line 46 of file BaseCut.h.

References c, and NamedO::str().

6.15.3 Member Data Documentation

6.15.3.1 const bool ConstBaseCut::c [protected]

The constant to be returned

Definition at line 55 of file BaseCut.h.

Referenced by ConstBaseCut(), and operator()().

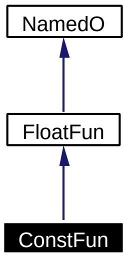
The documentation for this class was generated from the following file:

- **BaseCut.h**

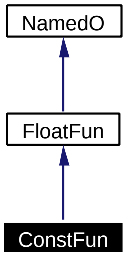
6.16 ConstFun Class Reference

`#include <FloatFun.h>`

Inheritance diagram for ConstFun:



Collaboration diagram for ConstFun:



6.16.1 Detailed Description

A **FloatFun** that returns a constant value.

Definition at line 52 of file FloatFun.h.

Public Member Functions

- **ConstFun** (Float_FF c_, const char *name_=0)
- **ConstFun** (Float_FF c_, const std::string &name_)

  Constructor from a C++ string.

- virtual Float_FF **operator()** () const

  Returns a constant value c.

Protected Member Functions

- virtual ~**ConstFun** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- Float_FF **c**

6.16.2 Constructor & Destructor Documentation

6.16.2.1 ConstFun::ConstFun (Float_FF *c_*, const char * *name_* = 0) [inline, explicit]

Parameters:

*c_* The constant to be returned

*name_* The object's name

Definition at line 55 of file FloatFun.h.

References c, and NamedO::str().

6.16.2.2 ConstFun::ConstFun (Float_FF *c_*, const std::string & *name_*) [inline]

Constructor from a C++ string.

Parameters:

*c_* The constant to be returned

*name_* The object's name

Definition at line 62 of file FloatFun.h.

References c, and NamedO::str().

6.16.3 Member Data Documentation

6.16.3.1 Float_FF ConstFun::c [protected]

The constant to be returned

Definition at line 71 of file FloatFun.h.

Referenced by ConstFun(), and operator()().

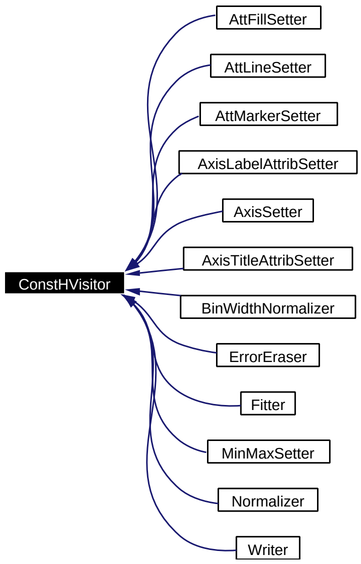
The documentation for this class was generated from the following file:

- **FloatFun.h**

6.17 ConstHVisitor Class Reference

`#include <HVisitor.h>`

Inheritance diagram for ConstHVisitor:



6.17.1 Detailed Description

Abstract base class for a constant visitor to all histograms.

A ConstHVisitor is a **HVisitor** that cannot be changed when it runs over a set of histograms.

Author: Jenny Böhme, Benno List

Date

2004/06/15 10:54:51

Author

blist

Definition at line 56 of file HVisitor.h.

Public Member Functions

- virtual void **visit** (**RegO** &h) const=0

6.17.2 Member Function Documentation

6.17.2.1 virtual void ConstHVisitor::visit (RegO & *h*) const `[pure virtual]`

Parameters:

h The objects to be visited

Implemented in **Fitter** (p. 84), **Writer** (p. 469), **Normalizer** (p. 226), **BinWidthNormalizer** (p. 53), **AttMarkerSetter** (p. 26), **AttLineSetter** (p. 24), **AttFillSetter** (p. 22), **ErrorEraser** (p. 72), **MinMaxSetter** (p. 212), **AxisTitleAttribSetter** (p. 32), **AxisLabelAttribSetter** (p. 28), and **AxisSetter** (p. 29).

Referenced by ROList::visit().

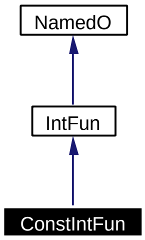
The documentation for this class was generated from the following file:

- **HVisitor.h**

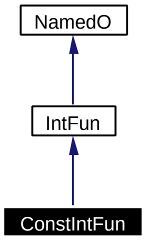
6.18 ConstIntFun Class Reference

`#include <IntFun.h>`

Inheritance diagram for ConstIntFun:



Collaboration diagram for ConstIntFun:



6.18.1 Detailed Description

A **IntFun** that returns a constant value.

Definition at line 44 of file IntFun.h.

Public Member Functions

- **ConstIntFun** (int c_, const char *name_=0)
  Constructor from a C style string, serves as default constructor.
- **ConstIntFun** (int c_, const std::string &name_)
  Constructor from a C++ string.
- virtual int **operator()** () const
  Returns a constant value c.

Protected Member Functions

- virtual ~**ConstIntFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const int **c**

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ConstIntFun::ConstIntFun (int *c_*, const char * *name_* = 0) `[inline, explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:

*c_* The constant to be returned

*name_* An optional name

Definition at line 47 of file IntFun.h.

References c, and NamedO::str().

6.18.2.2 ConstIntFun::ConstIntFun (int *c_*, const std::string & *name_*) `[inline, explicit]`

Constructor from a C++ string.

Parameters:

*c_* The constant to be returned

*name_* An optional name

Definition at line 55 of file IntFun.h.

References c, and NamedO::str().

6.18.3 Member Data Documentation

6.18.3.1 const int ConstIntFun::c `[protected]`

The constant to be returned

Definition at line 66 of file IntFun.h.

Referenced by ConstIntFun(), and operator()().

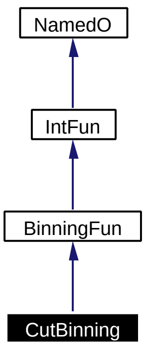
The documentation for this class was generated from the following file:

- **IntFun.h**

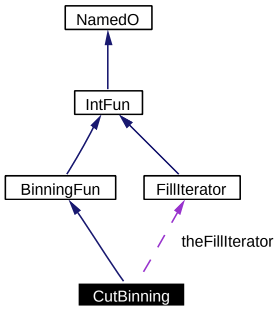
6.19 CutBinning Class Reference

`#include <CutBinning.h>`

Inheritance diagram for CutBinning:



Collaboration diagram for CutBinning:



6.19.1 Detailed Description

Class that generates a non-exclusive binning from an array of cuts.

The purpose of a CutBinning is to sort histogram entries into non-exclusive bins. An array of pointers to **BaseCut** objects is used to create the CutBinning object. The CutBinning object can then be used as **BinningFun** for a SFSetOfHistos or SFMatrixOfHistos to show how a quantity evolves when different cuts are applied.

Author: Benno List, Jenny List

Date
2005/07/20 10:51:32

Author
blist

Changelog:

- 10.6.05 BL: First version

- 20.7.05 BL: Added constructors for pointers to non-const **BaseCut**

Definition at line 46 of file CutBinning.h.

Public Member Functions

- **CutBinning** (int nbins, const **BaseCut** ∗cuts[ ], const char ∗**name**=0)

    Create a CutBinning from an array of pointers to cuts.

- **CutBinning** (int nbins, **BaseCut** ∗cuts[ ], const char ∗**name**=0)

    Create a CutBinning from an array of pointers to cuts.

- **CutBinning** (int nbins, const **BaseCut** ∗cuts[ ], const char ∗titles[ ], const char ∗**name**=0)

    Create a CutBinning from an array of pointers to cuts.

- **CutBinning** (int nbins, **BaseCut** ∗cuts[ ], const char ∗titles[ ], const char ∗**name**=0)

    Create a CutBinning from an array of pointers to cuts.

- virtual int **operator()** () const

    Returns a bin number between -1 and getNBins()-1.

- virtual bool **operator()** (int i) const

    Returns whether an event belongs to bin i.

- virtual bool **isExclusive** () const

    Returns whether a CutBinning is exclusive.

- virtual const char ∗ **getBinName** (int i) const

    returns a name string.

- virtual const char ∗ **getBinTitle** (int i) const

    returns a title string.

- virtual const **FillIterator** ∗ **getIterator** () const

    Returns pointer to iterator, if any.

- virtual int **getMaxBins** () const

    Gets the maximal return value of operator() plus 1, or the higest "sensible" value for a bin number.

- virtual bool **hasBinLabels** () const

    Returns whether the bins have alphanumeric bin labels.

- virtual const char ∗ **getBinLabel** (int i) const

    Returns the bin label (string must not be deleted by user!).

Protected Member Functions

- virtual ~**CutBinning** ()

  Protected destructor to ensure creation on the heap.

- void **genDefinition** ()

  Generates a definition string and sets the definition to it.

- void **checkIterators** ()

  Makes sure all cuts depend on the same (or no) iterator.

Private Attributes

- std::vector< const **BaseCut** * > **theCuts**
- std::vector< std::string > **binNames**
- std::vector< std::string > **binTitles**
- const **FillIterator** * **theFillIterator**

6.19.2 Constructor & Destructor Documentation

6.19.2.1 CutBinning::CutBinning (int *nbins*, const BaseCut * *cuts*[ ], const char * *name* = 0)

Create a CutBinning from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

name The object name

Definition at line 31 of file CutBinning.C.

References genDefinition(), and NamedO::str().

6.19.2.2 CutBinning::CutBinning (int *nbins*, BaseCut * *cuts*[ ], const char * *name* = 0)

Create a CutBinning from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

name The object name

Definition at line 45 of file CutBinning.C.

References genDefinition(), and NamedO::str().

6.19.2.3 CutBinning::CutBinning (int *nbins*, const BaseCut ∗ *cuts*[ ], const char ∗ *titles*[ ], const char ∗ *name* = 0)

Create a CutBinning from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

titles The title strings

name The object name

Definition at line 59 of file CutBinning.C.

References genDefinition(), and NamedO::str().

6.19.2.4 CutBinning::CutBinning (int *nbins*, BaseCut ∗ *cuts*[ ], const char ∗ *titles*[ ], const char ∗ *name* = 0)

Create a CutBinning from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

titles The title strings

name The object name

Definition at line 73 of file CutBinning.C.

References genDefinition(), and NamedO::str().

6.19.3 Member Function Documentation

6.19.3.1 bool CutBinning::operator() (int *i*) const `[virtual]`

Returns whether an event belongs to bin i.

Parameters:

i The bin number

Reimplemented from **BinningFun** (p. 46).

Definition at line 88 of file CutBinning.C.

6.19.3.2 const char ∗ CutBinning::getBinName (int *i*) const `[virtual]`

returns a name string.

The name string MAY, and MUST, be destroyed by the user! therefore: you MUST NOT return a string constant here, like "dummy", but use new[] to allocate the string. The name string should be short, unique, and contain no spaces, like "tbin0".

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 93 of file CutBinning.C.

6.19.3.3 const char * CutBinning::getBinTitle (int *i*) const [virtual]

returns a title string.

The name string MAY, and MUST, be destroyed by the user! therefore: you MUST NOT return a string constant here, like "dummy", but use new[] to allocate the string. The name string should describe the bin, like "0.1 < t < 0.5".

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 101 of file CutBinning.C.

Referenced by getBinLabel().

6.19.3.4 const char * CutBinning::getBinLabel (int *i*) const [virtual]

Returns the bin label (string must not be deleted by user!).

Parameters:

i The bin number

Definition at line 144 of file CutBinning.C.

References getBinTitle().

6.19.3.5 void CutBinning::genDefinition () [protected]

Generates a definition string and sets the definition to it.

This method is called in the constructor, therefore it makes no sense to make it virtual, because the constructor will always call **CutBinning::setDefinition()**.

Definition at line 109 of file CutBinning.C.

References NamedO::setDefinition().

Referenced by CutBinning().

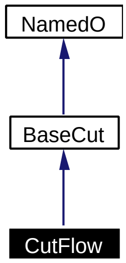
The documentation for this class was generated from the following files:

- **CutBinning.h**
- **CutBinning.C**

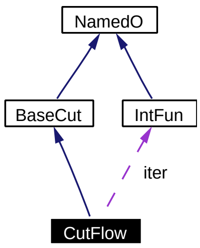
6.20 CutFlow Class Reference

`#include <CutFlow.h>`

Inheritance diagram for CutFlow:



Collaboration diagram for CutFlow:



6.20.1 Detailed Description

Class that returns the value of a cut depending on an iterator.

The purpose of a CutFlow object is to return the value of a cut whose number is given by an **IntFun** object (which will normally be a **FillIterator**). A CutFlow object will normally be used in conjuntion with a **RangeIterator** to generate a histogram that shows how the number of events evolves when different cuts are applied.

Here is an example how a CutFlow object can be used:

```
enum{NCUTS = 3};
const BaseCut *cutflow[NCUTS];
const char *cuttitles[NCUTS];
cutflow[0] = 0;
cutflow[1] = &(ncentral == 4);
cutflow[2] = &(*cutflow[1] && nnonvertex == 4);

cuttitles[0] = "No cuts";
cuttitles[1] = "4 central tracks";
cuttitles[2] = "... and only 4 nonvertex tracks";

BinningFun& cutflowbinning = *new CutBinning  (NCUTS, cutflow, cuttitles, "cut");
FillIterator& cutiter = *new RangeIterator (0, NCUTS-1, "cutnumber");
BaseCut& cutflowcut = *new CutFlow  (NCUTS, cutflow, cutiter, "cutflow");

h_cutflow        = new SFH1F ("cutflow", "Cut Flow (Events after L4)", cutflowbinning,
                              this, Float(cutiter), cutflowcut);
```

Author: Benno List, Jenny List

Date

2005/07/20 10:56:41

Author

blist

Changelog:

- 13.6.05 BL: First version
- 20.7.05 BL: Added constructors for pointers to non-const **BaseCut**

Definition at line 58 of file CutFlow.h.

Public Member Functions

- **CutFlow** (int nbins, const **BaseCut** ∗cuts[ ], const **IntFunPoR** &iter_, const char ∗**name**=0)

  Create a CutFlow from an array of pointers to cuts.

- **CutFlow** (int nbins, **BaseCut** ∗cuts[ ], const **IntFunPoR** &iter_, const char ∗**name**=0)

  Create a CutFlow from an array of pointers to cuts.

- virtual bool **operator()** () const

  Returns the value of the cut with the number given by the iterator.

- virtual const **FillIterator** ∗ **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ∼**CutFlow** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- std::vector< const **BaseCut** ∗ > **theCuts**
- const **IntFun** ∗ **iter**

6.20.2 Constructor & Destructor Documentation

6.20.2.1 CutFlow::CutFlow (int *nbins*, const BaseCut ∗ *cuts*[ ], const IntFunPoR & *iter_*, const char ∗ *name* = 0)

Create a CutFlow from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

*iter_* The Iterator

name The object name

Definition at line 28 of file CutFlow.C.

References NamedO::getName().

6.20.2.2 CutFlow::CutFlow (int *nbins*, BaseCut * *cuts*[ ], const IntFunPoR & *iter_*, const char * *name* = 0)

Create a CutFlow from an array of pointers to cuts.

Parameters:

nbins The number of bins

cuts The cuts

*iter_* The Iterator

name The object name

Definition at line 43 of file CutFlow.C.

References NamedO::getName().

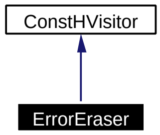
The documentation for this class was generated from the following files:

- **CutFlow.h**
- **CutFlow.C**

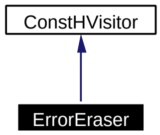
6.21 ErrorEraser Class Reference

`#include <HVisitors.h>`

Inheritance diagram for ErrorEraser:



Collaboration diagram for ErrorEraser:



6.21.1 Detailed Description

An **HVisitor** that sets all bin errors to 0 for TH1 histograms.

Definition at line 190 of file HVisitors.h.

Public Member Functions

- **ErrorEraser** ()

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

6.21.2 Member Function Documentation

6.21.2.1 virtual void ErrorEraser::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 195 of file HVisitors.h.

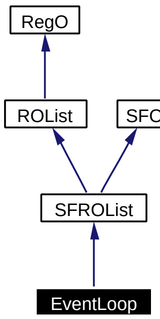
The documentation for this class was generated from the following file:

- **HVisitors.h**

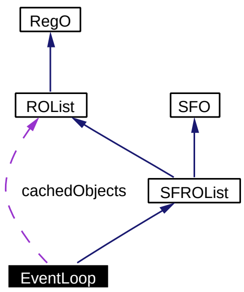
6.22 EventLoop Class Reference

`#include <EventLoop.h>`

Inheritance diagram for EventLoop:



Collaboration diagram for EventLoop:



6.22.1 Detailed Description

Base class for user event loops.

EventLoop is used as base class for user classes that perfotm analysis tasks in our framework. A class (say, class AnalysisLoop) derived from EventLoop will typically implement a constructor, where (self-filling) histograms are created (i.e., booked), and an output method that creates plots (written to a postscript file) and/or write the histograms to a RooT file.

For an ntuple, defined in the RooT-generated header file Ntuple.h (generated by TTree::MakeClass), a minimal main program will look like this:

```
*#define Ntuple_cxx
*#include "Ntuple.h"                          // The ntuple
*#include "AnalysisLoop.h"                    // The AnalysisLoop
*int main(int argc, char** argv)
*{
 Ntuple nt;                                 // open ntuple
 TApplication theApp("main", &argc, argv);  // for graphics
 AnalysisLoop theAnalysisLoop (&nt);        // book histos
 // Loop over ntuple entries
 for (int i = 0; i < int(nt.fChain->GetEntriesFast()); ++i) {
   if (nt.LoadTree (i) < 0) break;
```

```
   nt.fChain->GetEntry(i);
   theAnalysisLoop.loop();                    // fill the histos
 }
 theAnalysisLoop.output("out.root","out.ps"); // store the histos
 return 0;
*}
*
```

A minimal AnalysisLoop could look like this (assuming we have a class PTMissFun that returns the missing pt for each ntuple row):

```
*#include "jbltools/sfh/EventLoop.h"
*#include "jbltools/sfh/Binning.h"
*#include "jbltools/sfh/SFH1F.h"
*#include "PTMissFun.h"
*class AnalysisLoop: public EventLoop {
 public:
   // Constructor books histograms
   AnalysisLoop (Ntuple* ntuple)
   {
     Binning ptmissbinning (50, 0., 100.);
     FloatFun& ptmiss = *new PTMissFun (ntuple);
     // self-filling histogram for missing pt
     h = new SFH1F ("ptmiss", "Missing pt", ptmissbinning, this, ptmiss);

   }

   // Destructor does nothing
   virtual ~AnalysisLoop ()
   {}

   // output method writes histos to postscript and file
   virtual void output (const char* rootfile = "",
                        const char* psfile = "")
   {
     // create canvas
     TCanvas *canvas = new TCanvas("canvas", "ptmiss", 600, 800);
     // Write plot to psfile
     TPostScript ps (psfile, 111);
     canvas->Clear();
     h->Draw ("E0");                // Draws the histo
     canvas->Update();
     ps.Close();
     // Write histogram to file
     TFile file (rootfile, "RECREATE");
     h->Write();                    // Writes out the histo
     file.Write();
     file.Close();
   }
 protected:
   SFH1F *h;                        // The self-filling histogram
*};
*
```

Author: Jenny Böhme, Benno List

Date

2004/06/30 15:31:45

Author

blist

Definition at line 102 of file EventLoop.h.

Public Member Functions

- virtual ~**EventLoop** ()
  Virtual destructor.
- virtual void **loop** ()
  Fills the histograms and invalidates cached objects.
- virtual void **output** (const char *rootfile="", const char *psfile="")
  Generates postscript and RooT output.

Protected Attributes

- **ROList cachedObjects**
  A list of **CachedO** *objects.*

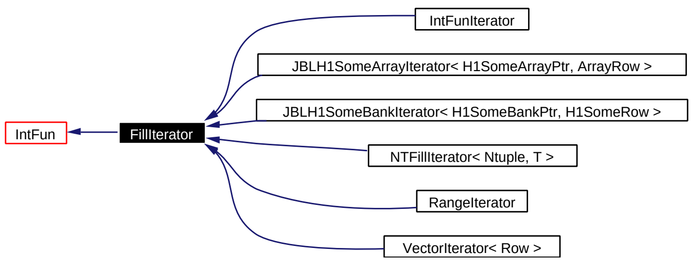
The documentation for this class was generated from the following files:

- **EventLoop.h**
- **EventLoop.C**

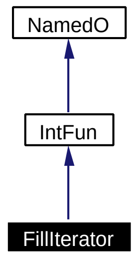
6.23 FillIterator Class Reference

`#include <FillIterator.h>`

Inheritance diagram for FillIterator:



Collaboration diagram for FillIterator:



6.23.1 Detailed Description

Abstract base class for iterators used to fill self-filling histograms.

A FillIterator object has to know how many entries for a given event should be made. If, for example, we have n D∗ meson candidates in an event, operator() should return a value between 0 (first D∗) and n-1 (last D∗). "-1" means "no D∗ available, or past the last D∗".

To iterate over all values, use code like this (iter is a pointer to a FillIterator, and may be 0, meaning "exactly 1 entry per event"):

```
  if (!iter || iter->reset()) {
  do {
    if (!cut || (*cut)()) {
      float w = wfun ? (*wfun)() : 1.;
      this->TH1F::Fill ((*xfun)(), w);
    }
  } while (iter && iter->next());
 *
```

If FillIterator objects are used, the corresponding **FloatFun** and **BaseCut** objects normally have to store a pointer to the iterator, so that a DStarMass object can ask the iterator the mass of which D∗ meson is to be plotted.

The destructor is protected to ensure FillIterator objects can only be created on the heap (with "new"). The resulting memory leak is acknowledged.

Normally, a concrete subclass of FillIterator will look something like this:

```
class MyIterator: public FillIterator {
public:
  MyIterator(): index(0) {}
  virtual int getN () const { return NUMBEROFENTRIES;};
  virtual int operator() () const {
    return (index < getN()) ? index : -1;
  }
  virtual bool next() {
    if (index++ >= getN()) {
      index = -1;
      return false;
    }
    return true;
  }
  virtual bool reset() {
    If (getN() <= 0) {
      index = -1;
      return false;
    }
    index = 0;
    return true;
  }
  virtual bool isValid() const {
    return (index >= 0) && index < getN();
  }
protected:
  ~MyIterator() {};
  int index;
};
```

Here getN() has to be implemented in a way that it returns the number of entries, e.g. D∗ mesons, for the current event. This will typically need a pointer to some Tree entry that contains the number of D∗ mesons.

Changelog

- 6.7.04 BL: FillIterator derived from **IntFun**
- 15.7.04 BL: Added method **getIterator()**
- 22.9.04 BL: Added method **isValid()**
- 4.6.05 BL: Added constructors with name as argument

Author: Jenny Böhme, Benno List

Date
: 2005/06/04 16:27:51

Author
: blist

Definition at line 90 of file FillIterator.h.

Public Member Functions

- **FillIterator** (const char ∗name_="?")
  Constructor from a C style string, serves as default constructor.

- **FillIterator** (const std::string &name_)
  Constructor from a C++ string.
- virtual int **operator()** () const=0
  Returns iterator value, starting at 0; "-1" means "invalid".
- virtual bool **next** ()=0
  Increments iterator; returns false if iterator cannot be incremented.
- virtual bool **reset** ()=0
  Resets iterator; returns false if iterator value range is empty.
- virtual bool **isValid** () const=0
  Returns whether current value of iterator is valid.
- virtual const **FillIterator** ∗ **getIterator** () const
  Returns pointer to iterator = itself.

Protected Member Functions

- virtual ∼**FillIterator** ()
  Protected destructor to ensure creation on the heap.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 FillIterator::FillIterator (const char ∗ *name_* = "?") `[inline, explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:
*name_* The object's name

Definition at line 94 of file FillIterator.h.

6.23.2.2 FillIterator::FillIterator (const std::string & *name_*) `[inline, explicit]`

Constructor from a C++ string.

Parameters:
*name_* The object's name

Definition at line 98 of file FillIterator.h.

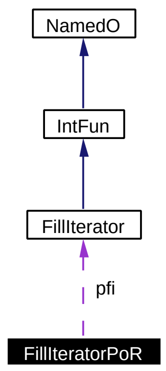
The documentation for this class was generated from the following file:

- **FillIterator.h**

6.24 FillIteratorPoR Class Reference

`#include <FillIteratorPoR.h>`

Collaboration diagram for FillIteratorPoR:



6.24.1 Detailed Description

FillIterator Pointer or Reference.

Automagically converts between pointers and references to **FillIterator** objects

Author: Benno List, Jenny Boehme

Date

2004/06/29 13:03:38

Author

blist

Definition at line 24 of file FillIteratorPoR.h.

Public Member Functions

- **FillIteratorPoR** (**FillIterator** &fi_)

  Constructor from a reference.

- **FillIteratorPoR** (**FillIterator** *pfi_=0)

  Constructor from a pointer.

Public Attributes

- **FillIterator** * **pfi**

  The pointer to a **FillIterator**.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 FillIteratorPoR::FillIteratorPoR (FillIterator & *fi_*) [inline]

Constructor from a reference.

Parameters:
fi_ A reference to a **FillIterator**

Definition at line 27 of file FillIteratorPoR.h.

References pfi.

6.24.2.2 FillIteratorPoR::FillIteratorPoR (FillIterator ∗ *pfi_* = 0) [inline]

Constructor from a pointer.

Parameters:
pfi_ A pointer to a **FillIterator**

Definition at line 31 of file FillIteratorPoR.h.

References pfi.

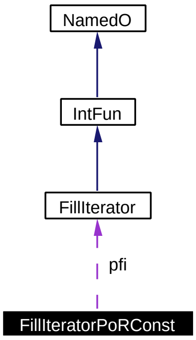
The documentation for this class was generated from the following file:

- **FillIteratorPoR.h**

6.25 FillIteratorPoRConst Class Reference

`#include <FillIteratorPoR.h>`

Collaboration diagram for FillIteratorPoRConst:



6.25.1 Detailed Description

const **FillIterator** Pointer or Reference

Automagically converts between pointers and references to const **FillIterator** objects

Author: Benno List, Jenny Boehme

Date

2004/06/29 13:03:38

Author

blist

Definition at line 49 of file FillIteratorPoR.h.

Public Member Functions

- **FillIteratorPoRConst** (const **FillIterator** &fi_)

  Constructor from a reference.

- **FillIteratorPoRConst** (const **FillIterator** *pfi_=0)

  Constructor from a pointer.

Public Attributes

- const **FillIterator** * **pfi**

  The pointer to a const **FillIterator**.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 FillIteratorPoRConst::FillIteratorPoRConst (const FillIterator & *fi_*) [inline]

Constructor from a reference.

Parameters:

*fi_* A reference to a const **FillIterator**

Definition at line 52 of file FillIteratorPoR.h.

References pfi.

6.25.2.2 FillIteratorPoRConst::FillIteratorPoRConst (const FillIterator * *pfi_* = 0) [inline]

Constructor from a pointer.

Parameters:

*pfi_* A pointer to a const **FillIterator**

Definition at line 56 of file FillIteratorPoR.h.

References pfi.

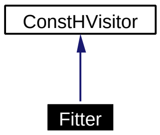
The documentation for this class was generated from the following file:

- **FillIteratorPoR.h**

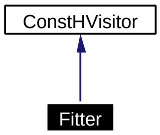
6.26 Fitter Class Reference

`#include <Fitter.h>`

Inheritance diagram for Fitter:



Collaboration diagram for Fitter:



6.26.1 Detailed Description

An **HVisitor** that fits TH1 histograms with a TF1 function.

Definition at line 16 of file Fitter.h.

Public Member Functions

- **Fitter** (TF1 *f1_, Option_t *option_, Option_t *goption_, Axis_t xmin_=0, Axis_t xmax_-=0)

  Constructor, takes the same arguments as TH1::Fit.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- TF1 * **f1**

  The fit function.

- Option_t * **option**

  The option.

- Option_t * **goption**

  The global option.

- Axis_t **xmin**

The minimal abscissa value for the fit range.

- Axis_t **xmax**

  The maximal abscissa value for the fit range.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Fitter::Fitter (TF1 ∗ *f1_*, Option_t ∗ *option_*, Option_t ∗ *goption_*, Axis_t *xmin_* = 0, Axis_t *xmax_* = 0) `[inline]`

Constructor, takes the same arguments as TH1::Fit.

Parameters:

f1_ The fit function

option_ The option

goption_ The global option

xmin_ The minimal abscissa value for the fit range

xmax_ The maximal abscissa value for the fit range

Definition at line 19 of file Fitter.h.

References f1, goption, option, xmax, and xmin.

6.26.3 Member Function Documentation

6.26.3.1 virtual void Fitter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 28 of file Fitter.h.

References f1, goption, option, xmax, and xmin.

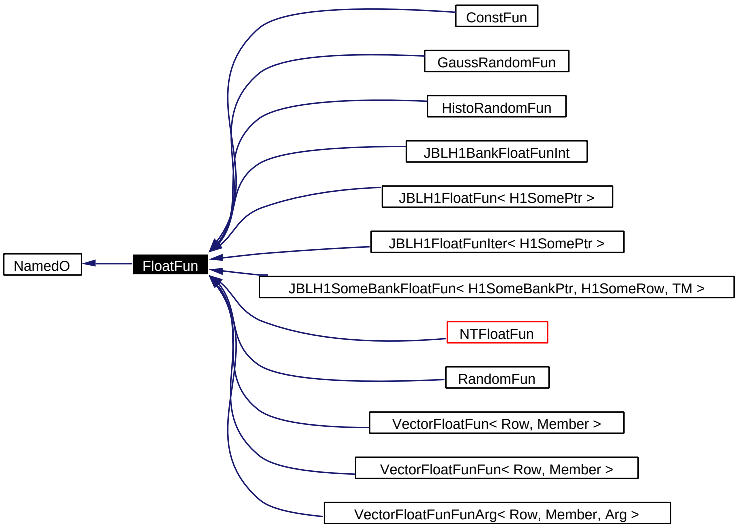
The documentation for this class was generated from the following file:

- **Fitter.h**

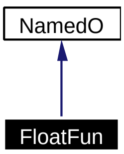
6.27 FloatFun Class Reference

`#include <FloatFunBase.h>`

Inheritance diagram for FloatFun:



Collaboration diagram for FloatFun:



6.27.1 Detailed Description

Base class for objects that return a float number for each event.

Classes derived from FloatFun have to implement just the function operator(); this operator has to return a float for an event, which can be used as abscissa or weight value for a (self-filling) histogram entry.

FloatFun objects are mainly used by self-filling histograms.

The destructor of FloatFun is protected; therefore, FloatFun objects cannot be created as automatic variables, only on the heap. This ensures that all FloatFun objects live until the end of the program run, otherwise it would be very hard to ensure that a self-filling histogram, which stores only a pointer to a cut object, does not try to access a cut object that has already been destroyed. Purists can use the "destroy" method to delete a FloatFun object.

Changelog:

- 15.7.04 BL: Added method **getIterator()**
- 22.7.04 BL: Added method **getName()**

- 29.5.05 BL: Derived from class **NamedO**: makes lookup according to name possible
- 31.5.05 BL: Added constructors that pass name to **NamedO**
- 6.6.05 BL: Added constructor with definition
- 21.7.05 BL: Use Float_FF instead of float as result type of FloatFun

Author: Benno List, Jenny Böhme

Date

2005/07/21 11:20:12

Author

blist

Definition at line 51 of file FloatFunBase.h.

Public Member Functions

- **FloatFun** (const char *name_="?")
  Constructor from a C style string, serves as default constructor.
- **FloatFun** (const std::string &name_)
  Constructor from a C++ string.
- **FloatFun** (const std::string &name_, const std::string &definition_)
  Constructor from two C++ strings.
- virtual Float_FF **operator()** () const=0
  Returns a float value, typically from an ntuple row.
- virtual void **destroy** ()
  Replaces delete, which cannot be called because of protected destructor.
- virtual const **FillIterator** * **getIterator** () const
  Returns pointer to iterator, if any.

Static Public Member Functions

- **FloatFun** * **getObject** (const std::string &**name**)
  Access function that returns a pointer to an FloatFun object with this name.
- **FloatFun** * **getObject** (const char ***name**)
  Access function that returns a pointer to an **NamedO** *object with this name.*

Protected Member Functions

- virtual ~**FloatFun** ()
  Protected destructor ensures creation on the heap.

Related Functions

(Note that these are not member functions.)

- **FloatFun** & **operator+** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Sum of two FloatFun objects.
- **FloatFun** & **operator+** (const **FloatFun** &lhs_, Float_FF rhs_)
  Sum of a FloatFun object with a scalar.
- **FloatFun** & **operator+** (Float_FF lhs_, const **FloatFun** &rhs_)
  Sum of a FloatFun object with a scalar.
- **FloatFun** & **operator-** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Difference of two FloatFun objects.
- **FloatFun** & **operator-** (const **FloatFun** &lhs_, Float_FF rhs_)
  Difference of a FloatFun object with a scalar.
- **FloatFun** & **operator-** (Float_FF lhs_, const **FloatFun** &rhs_)
  Sum of a FloatFun object with a scalar.
- **FloatFun** & **operator-** (const **FloatFun** &arg_)
  Unary minus of a FloatFun.
- **FloatFun** & **operator+** (**FloatFun** &arg_)
  Unary plus of a FloatFun.
- **FloatFun** & **operator** ∗ (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Product of two FloatFun objects.
- **FloatFun** & **operator** ∗ (Float_FF lhs_, const **FloatFun** &rhs_)
  Scalar product with a FloatFun.
- **FloatFun** & **operator** ∗ (const **FloatFun** &lhs_, Float_FF rhs_)
  Scalar product with a FloatFun.
- **FloatFun** & **operator/** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Quotient of two FloatFun objects.
- **FloatFun** & **operator/** (Float_FF lhs_, const **FloatFun** &rhs_)
  Division of a scalar by a FloatFun.
- **FloatFun** & **operator/** (const **FloatFun** &lhs_, Float_FF rhs_)
  Division of a FloatFun by a scalar.
- **FloatFun** & **abs** (const **FloatFun** &arg_)
  Absolute value of a FloatFun.
- **FloatFun** & **acos** (const **FloatFun** &arg_)

Arc cosine of a FloatFun.

- **FloatFun** & **asin** (const **FloatFun** &arg_)

  Arc sine of a FloatFun.

- **FloatFun** & **atan** (const **FloatFun** &arg_)

  Arc Tangent of a FloatFun.

- **FloatFun** & **atan2** (const **FloatFun** &arg1, const **FloatFun** &arg2)

  Arc tangent of two FloatFun objects: atan2(y, x)=atan(y/x).

- **FloatFun** & **ceil** (const **FloatFun** &arg_)

  Ceiling of a FloatFun: smallest integral value not less than argument.

- **FloatFun** & **cos** (const **FloatFun** &arg_)

  Cosine of a FloatFun.

- **FloatFun** & **cosh** (const **FloatFun** &arg_)

  Hyperbolic Cosine of a FloatFun.

- **FloatFun** & **exp** (const **FloatFun** &arg_)

  Exponential function of a FloatFun.

- **FloatFun** & **fabs** (const **FloatFun** &arg_)

  Absolute value of a FloatFun.

- **FloatFun** & **floor** (const **FloatFun** &arg_)

  Floor value of a FloatFun: largest integral value not greater than argument.

- **FloatFun** & **log** (const **FloatFun** &arg_)

  Natural logarithm of a FloatFun.

- **FloatFun** & **log10** (const **FloatFun** &arg_)

  Decadic logarithm of a FloatFun.

- **FloatFun** & **pow** (const **FloatFun** &arg1_, const **FloatFun** &arg2_)

  Power of a FloatFun.

- **FloatFun** & **sin** (const **FloatFun** &arg_)

  Sine of a FloatFun.

- **FloatFun** & **sinh** (const **FloatFun** &arg_)

  Hyperbolic sine of a FloatFun.

- **FloatFun** & **tanh** (const **FloatFun** &arg_)

  Hyperbolic tangent of a FloatFun.

- **FloatFun** & **square** (const **FloatFun** &arg_)

  Suare of a FloatFun.

- **FloatFun** & **rad** (const **FloatFun** &arg_)
  Conversion from degrees to radians.
- **FloatFun** & **deg** (const **FloatFun** &arg_)
  Conversion from radians to degrees.
- **FloatFunComp** & **operator<** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator>** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator<=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator>=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator==** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator!=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
  Comparison between two FloatFun objects.
- **FloatFunComp** & **operator<** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator<=** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator>** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator>=** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator==** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator!=** (Float_FF lhs_, const **FloatFun** &rhs_)
  Comparison between a Float_FF and a FloatFun.
- **FloatFunComp** & **operator<** (const **FloatFun** &lhs_, Float_FF rhs_)
  Comparison between a FloatFun and a Float_FF.
- **FloatFunComp** & **operator<=** (const **FloatFun** &lhs_, Float_FF rhs_)
  Comparison between a FloatFun and a Float_FF.
- **FloatFunComp** & **operator>** (const **FloatFun** &lhs_, Float_FF rhs_)
  Comparison between a FloatFun and a Float_FF.

- **FloatFunComp** & **operator>=** (const **FloatFun** &lhs_, Float_FF rhs_)

  Comparison between a FloatFun and a Float_FF.

- **FloatFunComp** & **operator==** (const **FloatFun** &lhs_, Float_FF rhs_)

  Comparison between a FloatFun and a Float_FF.

- **FloatFunComp** & **operator!=** (const **FloatFun** &lhs_, Float_FF rhs_)

  Comparison between a FloatFun and a Float_FF.

- **FloatFunComp** & **operator<** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator<=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator>** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator>=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator==** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator!=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator<** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a FloatFun object at the right side.

- **FloatFunComp** & **operator<=** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a Float_FF at the right side.

- **FloatFunComp** & **operator>** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a Float_FF at the right side.

- **FloatFunComp** & **operator>=** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a Float_FF at the right side.

- **FloatFunComp** & **operator==** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a Float_FF at the right side.

- **FloatFunComp** & **operator!=** (const **FloatFunComp** &lhs_, Float_FF rhs_)

  Chained comparison, with a Float_FF at the right side.

- **FloatFun** & **max** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the maximum of all values of fun if cut is fulfilled.

- **FloatFun** & **min** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the minimum of all values of fun if cut is fulfilled.

- **FloatFun** & **sum** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the sum of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

- **FloatFun** & **average** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the average of all values of fun if cut is fulfilled.

- **FloatFun** & **cached** (const **ROListPoR** &rol, const **FloatFunPoR** &fun, const **FillIteratorPoR** &iter=0)

  Returns a cached version of a FloatFun.

- **FloatFun** & **gauss** (const **FloatFunPoR** &rnd, double mean, double sigma)

  Returns a Gaussian distributed value from a random function.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 FloatFun::FloatFun (const char * *name_* = "?") `[inline, explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:

name_ The object's name

Definition at line 55 of file FloatFunBase.h.

6.27.2.2 FloatFun::FloatFun (const std::string & *name_*) `[inline, explicit]`

Constructor from a C++ string.

Parameters:

name_ The object's name

Definition at line 59 of file FloatFunBase.h.

6.27.2.3 FloatFun::FloatFun (const std::string & *name_*, const std::string & *definition_*) `[inline]`

Constructor from two C++ strings.

Parameters:

name_ The object's name

definition_ The object's definition

Definition at line 63 of file FloatFunBase.h.

6.27.3 Member Function Documentation

6.27.3.1 FloatFun* FloatFun::getObject (const std::string & *name*) [inline, static]

Access function that returns a pointer to an FloatFun object with this name.

This access function must never be called within the constructor of any class that is derived from FloatFun!

Parameters:

name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 80 of file FloatFunBase.h.

6.27.3.2 FloatFun* FloatFun::getObject (const char * *name*) [inline, static]

Access function that returns a pointer to an **NamedO** object with this name.

This access function must never be called within the constructor of any class that is derived from NamedO!

Parameters:

name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 90 of file FloatFunBase.h.

6.27.4 Friends And Related Function Documentation

6.27.4.1 FloatFun & max (const FloatFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) [related]

Returns the maximum of all values of fun if cut is fulfilled.

Parameters:

fun An optional FloatFun

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 648 of file FloatFun.C.

6.27.4.2 FloatFun & min (const FloatFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) [related]

Returns the minimum of all values of fun if cut is fulfilled.

Parameters:

fun An optional FloatFun

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 656 of file FloatFun.C.

6.27.4.3 FloatFun & sum (const FloatFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the sum of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

Parameters:

fun An optional FloatFun

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 664 of file FloatFun.C.

6.27.4.4 FloatFun & average (const FloatFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the average of all values of fun if cut is fulfilled.

Parameters:

fun An optional FloatFun

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 672 of file FloatFun.C.

6.27.4.5 FloatFun & cached (const ROListPoR & *rol*, const FloatFunPoR & *fun*, const FillIteratorPoR & *iter* = 0) `[related]`

Returns a cached version of a FloatFun.

Parameters:

rol Cached objects list where to register

fun A FloatFun

iter An optional iterator

Definition at line 444 of file FloatFun.C.

References getIterator(), FloatFunPoR::pff, and FillIteratorPoR::pfi.

6.27.4.6 FloatFun & gauss (const FloatFunPoR & *rnd*, double *mean*, double *sigma*) `[related]`

Returns a Gaussian distributed value from a random function.

Parameters:

rnd A random function

mean The mean value

sigma The sigma value

Definition at line 302 of file RandomFuns.C.

References FloatFunPoR::pff.

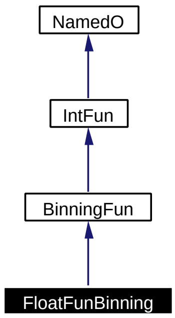
The documentation for this class was generated from the following files:

- **FloatFunBase.h**
- **FloatFun.h**
- **RandomFuns.h**

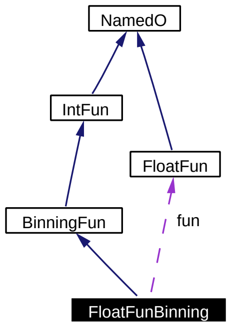
6.28 FloatFunBinning Class Reference

`#include <FloatFunBinning.h>`

Inheritance diagram for FloatFunBinning:



Collaboration diagram for FloatFunBinning:



6.28.1 Detailed Description

A class to get a concrete **BinningFun** subclass from a **FloatFun**.

The class FloatFunBinning can be used to generate a **BinningFun** in an easy way from a **FloatFun**, which delivers the value (e.g., t) that is used for binning, and some strings that are used to construct the name and title strings.

Author: Jenny Böhme, Benno List

Date

2005/08/23 12:05:43

Author

blist

Changelog:

- 4.6.04 BL: Derived BiningFun from Binning, so FloatFunBinning inherits getNBins() and getBin().

- 29.6.04 BL: Add constructor from Binning object
- 15.7.04 BL: Added method **getIterator()**
- 23.8.05 BL: Use double for xmin and xmax

Definition at line 31 of file FloatFunBinning.h.

Public Member Functions

- **FloatFunBinning** (const **FloatFunPoR** &fun_, int nbins_, double xlow_, double xhigh_, const char ∗varname_)

  Create an equidistant binning.

- **FloatFunBinning** (const **FloatFunPoR** &fun_, int nbins_, const float binedges_[ ], const char ∗varname_)

  Create an nonequidistant binning taking floats.

- **FloatFunBinning** (const **FloatFunPoR** &fun_, int nbins_, const double binedges_[ ], const char ∗varname_)

  Create an nonequidistant binning taking doubles.

- **FloatFunBinning** (const **FloatFunPoR** &fun_, const Binning &binning_, const char ∗varname_)

  Create a binning from a Binning object.

- virtual int **operator()** () const

  Returns a bin number between -1 and getNBins()-1.

- virtual const char ∗ **getBinName** (int i) const

  Returns a name string.

- virtual const char ∗ **getBinTitle** (int i) const

  Returns a title string.

- virtual const **FillIterator** ∗ **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ∼**FloatFunBinning** ()

  Protected destructor to ensure creation on the heap.

Protected Attributes

- const **FloatFun** ∗ **fun**

  The **FloatFun** *object defining the quantity used for binning.*

- char * **varname**

  The variable name.

- int **varnamelength**

  The length of the variable name.

Private Member Functions

- void **initvarname** (const char *varname_)
- **FloatFunBinning** (const **FloatFunBinning** &)

  Private copy constructor, left unimplemented: No copying!

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

  Private assignment operator, left unimplemented: No assignment!

6.28.2 Constructor & Destructor Documentation

6.28.2.1 FloatFunBinning::FloatFunBinning (const FloatFunPoR & *fun_*, int *nbins_*, double *xlow_*, double *xhigh_*, const char * *varname_*)

Create an equidistant binning.

The binning will have nbins_ bins between xlow and xhigh.

Parameters:

fun_ The **FloatFun** object defining the quantity used for binning

nbins_ The number of bins

xlow_ The lowest bin edge

xhigh_ The highest bin edge

varname_ The variable name

Definition at line 43 of file FloatFunBinning.C.

6.28.2.2 FloatFunBinning::FloatFunBinning (const FloatFunPoR & *fun_*, int *nbins_*, const float *binedges_*[ ], const char * *varname_*)

Create an nonequidistant binning taking floats.

The binning will have nbins_ bins with binedges given by binedges_. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

fun_ The **FloatFun** object defining the quantity used for binning

nbins_ The number of bins

binedges_ An array with binedges

varname_ The variable name

Definition at line 57 of file FloatFunBinning.C.

6.28.2.3 FloatFunBinning::FloatFunBinning (const FloatFunPoR & *fun_*, int *nbins_*, const double *binedges_*[ ], const char * *varname_*)

Create an nonequidistant binning taking doubles.

The binning will have nbins_ bins with binedges given by binedges_. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

fun_ The **FloatFun** object defining the quantity used for binning

nbins_ The number of bins

binedges_ An array with binedges

varname_ The variable name

Definition at line 71 of file FloatFunBinning.C.

6.28.2.4 FloatFunBinning::FloatFunBinning (const FloatFunPoR & *fun_*, const Binning & *binning_*, const char * *varname_*)

Create a binning from a Binning object.

Parameters:

fun_ The **FloatFun** object defining the quantity used for binning

binning_ The binning

varname_ The variable name

Definition at line 85 of file FloatFunBinning.C.

6.28.3 Member Function Documentation

6.28.3.1 const char * FloatFunBinning::getBinName (int *i*) const `[virtual]`

Returns a name string.

The name string MAY, and MUST, be destroyed by the user! The name will be (in perl notation) "$varname$bin"

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 133 of file FloatFunBinning.C.

6.28.3.2 const char * FloatFunBinning::getBinTitle (int *i*) const `[virtual]`

Returns a title string.

The name string MAY, and MUST, be destroyed by the user! The title will be (in perl notation) "$lower_edge < $varname < $higher_edge"

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 116 of file FloatFunBinning.C.

References varname, and varnamelength.

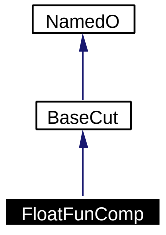
The documentation for this class was generated from the following files:

- **FloatFunBinning.h**
- **FloatFunBinning.C**

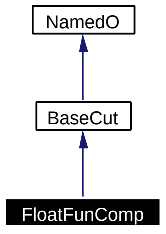
6.29 FloatFunComp Class Reference

`#include <FloatFun.h>`

Inheritance diagram for FloatFunComp:



Collaboration diagram for FloatFunComp:



6.29.1 Detailed Description

Abstract base class for comparisons between **FloatFun** objects.

This class is the base class for the result of all comparisons like **FloatFun** < **FloatFun**. It allows to chain comparisons, i.e. write 1<x<2 instead of (1<x)&&(x<2).

Definition at line 98 of file FloatFun.h.

Public Member Functions

- **FloatFunComp** (const std::string &name_)
- virtual Float_FF **lhsValue** () const=0

  Returns value of left-hand side argument.

- virtual Float_FF **rhsValue** () const=0

  Returns value of left-hand side argument.

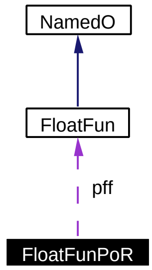
The documentation for this class was generated from the following file:

- **FloatFun.h**

6.30 FloatFunPoR Class Reference

`#include <FloatFunPoR.h>`

Collaboration diagram for FloatFunPoR:



6.30.1 Detailed Description

FloatFun Pointer or Reference.

Automagically converts between pointers and references to **FloatFun** objects

Author: Benno List, Jenny Boehme

Date
2005/06/29 14:40:18

by:

Author
blist

Changelog:

- 29.5.05 BL: Make it possible to access a **FloatFun** also by its name

Definition at line 29 of file FloatFunPoR.h.

Public Member Functions

- **FloatFunPoR** (**FloatFun** &ff_)
  Constructor from a reference.
- **FloatFunPoR** (**FloatFun** *pff_)
  Constructor from a pointer.
- **FloatFunPoR** (int=0)
  Constructor from a int (will normally be 0!).
- **FloatFunPoR** (const std::string &name)
  Constructor from a name.

- **FloatFunPoR** (const char *name)
  Constructor from a name.

Public Attributes

- **FloatFun** * **pff**
  The pointer to the **FloatFun**.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 FloatFunPoR::FloatFunPoR (FloatFun & *ff_*) `[inline]`

Constructor from a reference.

Parameters:
*ff_* A reference to a **FloatFun**

Definition at line 32 of file FloatFunPoR.h.

References pff.

6.30.2.2 FloatFunPoR::FloatFunPoR (FloatFun * *pff_*) `[inline]`

Constructor from a pointer.

Parameters:
*pff_* A pointer to a **FloatFun**

Definition at line 36 of file FloatFunPoR.h.

References pff.

6.30.2.3 FloatFunPoR::FloatFunPoR (const std::string & *name*)

Constructor from a name.

Parameters:
name A name of a **FloatFun**

Definition at line 21 of file FloatFunPoR.C.

6.30.2.4 FloatFunPoR::FloatFunPoR (const char * *name*)

Constructor from a name.

Parameters:
name A name of a **FloatFun**

Definition at line 24 of file FloatFunPoR.C.

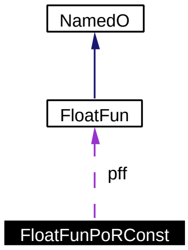
The documentation for this class was generated from the following files:

- **FloatFunPoR.h**
- **FloatFunPoR.C**

6.31 FloatFunPoRConst Class Reference

`#include <FloatFunPoR.h>`

Collaboration diagram for FloatFunPoRConst:



6.31.1 Detailed Description

Constant **FloatFun** Pointer or Reference.

Automagically converts between pointers and references to const **FloatFun** objects

Author: Benno List, Jenny Boehme

Date

2005/06/29 14:40:18

by:

Author

blist

Changelog:

- 29.5.05 BL: Make it possible to access a **FloatFun** also by its name

Definition at line 67 of file FloatFunPoR.h.

Public Member Functions

- **FloatFunPoRConst** (const **FloatFun** &ff_)
  Constructor from a reference.
- **FloatFunPoRConst** (const **FloatFun** *pff_=0)
  Constructor from a pointer.
- **FloatFunPoRConst** (int=0)
  Constructor from a int (will normally be 0!).
- **FloatFunPoRConst** (const std::string &name)
  Constructor from a name.

- **FloatFunPoRConst** (const char *name)

  Constructor from a name.

Public Attributes

- const **FloatFun** * **pff**

  The pointer to the const **FloatFun***.*

6.31.2 Constructor & Destructor Documentation

6.31.2.1 FloatFunPoRConst::FloatFunPoRConst (const FloatFun & *ff_*) `[inline]`

Constructor from a reference.

Parameters:

ff_ A reference to a const **FloatFun**

Definition at line 70 of file FloatFunPoR.h.

References pff.

6.31.2.2 FloatFunPoRConst::FloatFunPoRConst (const FloatFun * *pff_* = 0) `[inline]`

Constructor from a pointer.

Parameters:

pff_ A pointer to a const **FloatFun**

Definition at line 74 of file FloatFunPoR.h.

References pff.

6.31.2.3 FloatFunPoRConst::FloatFunPoRConst (const std::string & *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 27 of file FloatFunPoR.C.

6.31.2.4 FloatFunPoRConst::FloatFunPoRConst (const char * *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 30 of file FloatFunPoR.C.

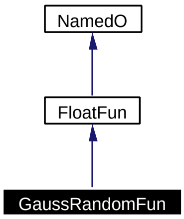
The documentation for this class was generated from the following files:

- **FloatFunPoR.h**
- **FloatFunPoR.C**

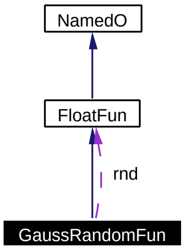
6.32 GaussRandomFun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for GaussRandomFun:



Collaboration diagram for GaussRandomFun:



6.32.1 Detailed Description

A **FloatFun** that returns a gaussian distributed value from a random number.

Definition at line 98 of file RandomFuns.h.

Public Member Functions

- **GaussRandomFun** (const **FloatFun** &rnd_, double mean_, double sigma_, const char *name_=0)
- **GaussRandomFun** (const **FloatFun** &rnd_, double mean_, double sigma_, const std::string &name_)

  Constructor from a C++ string.

- virtual Float_FF **operator()** () const

  Returns a GaussRandom number that depends on seed() and index.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ~**GaussRandomFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const **FloatFun** & **rnd**
  The random function.
- double **mean**
  The mean value.
- double **sigma**

6.32.2 Constructor & Destructor Documentation

6.32.2.1 GaussRandomFun::GaussRandomFun (const FloatFun & *rnd_*, double *mean_*, double *sigma_*, const char * *name_* = 0) [explicit]

Parameters:
*rnd_* The random function
*mean_* The mean value
*sigma_* The mean value
*name_* The object's name

Definition at line 106 of file RandomFuns.C.

6.32.2.2 GaussRandomFun::GaussRandomFun (const FloatFun & *rnd_*, double *mean_*, double *sigma_*, const std::string & *name_*)

Constructor from a C++ string.

Parameters:
*rnd_* The random function
*mean_* The mean value
*sigma_* The mean value
*name_* The object's name

Definition at line 115 of file RandomFuns.C.

6.32.3 Member Data Documentation

6.32.3.1 double GaussRandomFun::sigma [protected]

The sigma value

Definition at line 121 of file RandomFuns.h.

Referenced by operator()().

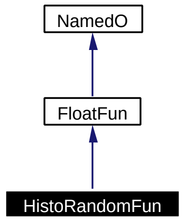
The documentation for this class was generated from the following files:

- **RandomFuns.h**
- **RandomFuns.C**

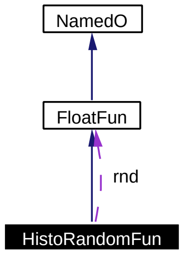
6.33 HistoRandomFun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for HistoRandomFun:



Collaboration diagram for HistoRandomFun:



6.33.1 Detailed Description

A **FloatFun** that returns a random number distributed according to a histogram.

Definition at line 190 of file RandomFuns.h.

Public Member Functions

- **HistoRandomFun** (const **FloatFun** &rnd_, const TH1 &histo, const char *name_=0)

  Constructor from a C style string, serves as default constructor.

- **HistoRandomFun** (const **FloatFun** &rnd_, const TH1 &histo, const std::string &name_-)

  Constructor from a C++ string.

- virtual Float_FF **operator()** () const

  Returns a random value, distributed according to the histogram.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- void **init** (const TH1 &histo)
  Initialization.

- virtual ~**HistoRandomFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const **FloatFun** & **rnd**
  The gaussian random function.

- int **nbins**
  The number of bins.

- double ∗ **edges**
  The bin edges.

- double ∗ **values**

6.33.2 Constructor & Destructor Documentation

6.33.2.1 HistoRandomFun::HistoRandomFun (const FloatFun & *rnd_*, const TH1 & *histo*, const char ∗ *name_* = 0) `[explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:
*rnd_* The random function
histo The histogram
*name_* An optional name

Definition at line 180 of file RandomFuns.C.

References init().

6.33.2.2 HistoRandomFun::HistoRandomFun (const FloatFun & *rnd_*, const TH1 & *histo*, const std::string & *name_*)

Constructor from a C++ string.

Parameters:
*rnd_* The random function
histo The histogram
*name_* An optional name

Definition at line 189 of file RandomFuns.C.

References init().

6.33.3 Member Function Documentation

6.33.3.1 void HistoRandomFun::init (const TH1 & *histo*) [protected]

Initialization.

Parameters:

histo The histogram

Definition at line 216 of file RandomFuns.C.

References edges, nbins, and values.

Referenced by HistoRandomFun().

6.33.4 Member Data Documentation

6.33.4.1 double∗ HistoRandomFun::values [protected]

The bin values

Definition at line 214 of file RandomFuns.h.

Referenced by init(), operator()(), and ∼HistoRandomFun().

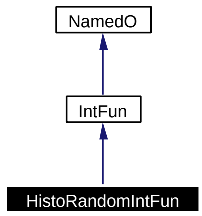
The documentation for this class was generated from the following files:

- **RandomFuns.h**
- **RandomFuns.C**

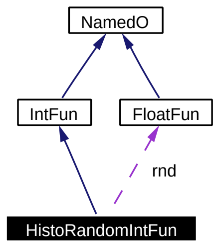
6.34 HistoRandomIntFun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for HistoRandomIntFun:



Collaboration diagram for HistoRandomIntFun:



6.34.1 Detailed Description

A **FloatFun** that returns a random number distributed according to a histogram.

Definition at line 225 of file RandomFuns.h.

Public Member Functions

- **HistoRandomIntFun** (const **FloatFun** &rnd_, const TH1 &histo, const char *name_=0)

  Constructor from a C style string, serves as default constructor.

- **HistoRandomIntFun** (const **FloatFun** &rnd_, const TH1 &histo, const std::string &name_)

  Constructor from a C++ string.

- virtual int **operator()** () const

  Returns a random value, distributed according to the histogram.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- void **init** (const TH1 &histo)
  Initialization.
- virtual ~**HistoRandomIntFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const **FloatFun** & **rnd**
  The gaussian random function.
- int **nbins**
  The number of bins.
- double ∗ **edges**
  The bin edges.
- double ∗ **values**

6.34.2 Constructor & Destructor Documentation

6.34.2.1 HistoRandomIntFun::HistoRandomIntFun (const FloatFun & *rnd_*, const TH1 & *histo*, const char ∗ *name_* = 0) [explicit]

Constructor from a C style string, serves as default constructor.

Parameters:
 rnd_ The random function
 histo The histogram
 name_ An optional name

Definition at line 237 of file RandomFuns.C.

References init().

6.34.2.2 HistoRandomIntFun::HistoRandomIntFun (const FloatFun & *rnd_*, const TH1 & *histo*, const std::string & *name_*)

Constructor from a C++ string.

Parameters:
 rnd_ The random function
 histo The histogram
 name_ An optional name

Definition at line 246 of file RandomFuns.C.

References init().

6.34.3 Member Function Documentation

6.34.3.1 void HistoRandomIntFun::init (const TH1 & *histo*) [protected]

Initialization.

Parameters:

histo The histogram

Definition at line 273 of file RandomFuns.C.

References edges, nbins, and values.

Referenced by HistoRandomIntFun().

6.34.4 Member Data Documentation

6.34.4.1 double∗ HistoRandomIntFun::values [protected]

The bin values

Definition at line 249 of file RandomFuns.h.

Referenced by init(), operator()(), and ∼HistoRandomIntFun().

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

- **RandomFuns.h**
- **RandomFuns.C**

6.35 HVisitor Class Reference

`#include <HVisitor.h>`

6.35.1 Detailed Description

Abstract base class for a visitor to all histograms.

One possibility to operate on all objects is to use an HVisitor object, which will be called with all stored objects in turn (see Gamma, Helm, Johnson, Vlissides: Design Patterns for a discussion of the Visitor pattern).

Typically, an HVisitor object will use a dynamic_cast to check whether the **RegO** object at hand is a root histogram, e.g. a TH1, TH1F, TH2F, or whatever, and then apply some method(s) to the histogram.

The visitor can store information, like a number of entries a histogram should be normalized to, a line color or whatever. It can also be used to collect statistics, like the total number of entries in all histograms in a HList (useful if the HList is really a **SetOfHistograms** or a **MatrixOfHistograms**).

The visitor can perform rather complicated operations, like plotting a histogram with several fit functions overlaid, or performing a fit.

See file **HVisitors.h** for some examples of concrete HVisitor subclasses.

Author: Jenny Böhme, Benno List

Date
 2004/06/15 10:54:51

Author
 blist

Definition at line 36 of file HVisitor.h.

Public Member Functions

- virtual void **visit** (**RegO** &h)=0
  The visitor.

6.35.2 Member Function Documentation

6.35.2.1 virtual void HVisitor::visit (RegO & *h*) `[pure virtual]`

The visitor.

Parameters:
 h The histogram to be visited

Referenced by ROList::visit().

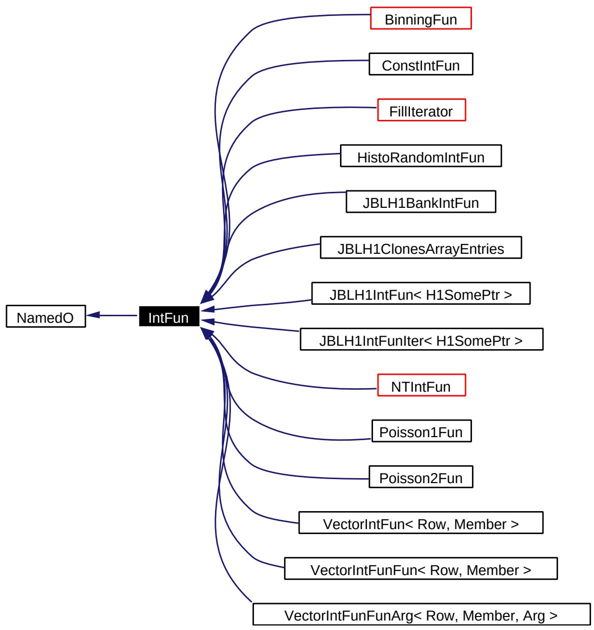
The documentation for this class was generated from the following file:

- **HVisitor.h**

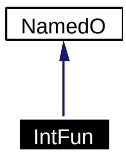
6.36 IntFun Class Reference

`#include <IntFunBase.h>`

Inheritance diagram for IntFun:



Collaboration diagram for IntFun:



6.36.1 Detailed Description

Base class for objects that return a int number for each event.

Classes derived from IntFun have to implement just the function operator(); this operator has to return a int for an event, which can be used as abscissa or weight value for a (self-filling) histogram entry.

IntFun objects are mainly used by self-filling histograms.

The destructor of IntFun is protected; therefore, IntFun objects cannot be created as automatic variables, only on the heap. This ensures that all IntFun objects live until the end of the program run, otherwise it would be very hard to ensure that a self-filling histogram, which stores only a pointer to a cut object, does not try to access a cut object that has already been destroyed. Purists can use the "destroy" method to delete a IntFun object.

Changelog:

- 15.7.04 BL: Added method **getIterator()**
- 22.7.04 BL: Added method **getName()**
- 29.5.05 BL: Derived from class **NamedO**: makes lookup according to name possible
- 2.6.05 BL: Added constructors that pass name to **NamedO**
- 6.6.05 BL: Added constructor with definition

Author: Benno List, Jenny Böhme

Date
2005/06/06 15:01:42

Author
blist

Definition at line 51 of file IntFunBase.h.

Public Member Functions

- **IntFun** (const char *name_="?")
  Constructor from a C style string, serves as default constructor.
- **IntFun** (const std::string &name_)
  Constructor from a C++ string.
- **IntFun** (const std::string &name_, const std::string &definition_)
  Constructor from two C++ strings.
- virtual int **operator()** () const=0
  Returns a int value, typically from an ntuple row.
- virtual **FloatFun** & **Float** () const
  Returns a **FloatFun** *object.*
- virtual void **destroy** ()
  Replaces delete, which cannot be called because of protected destructor.
- virtual const **FillIterator** * **getIterator** () const
  Returns pointer to iterator, if any.

Static Public Member Functions

- **IntFun** * **getObject** (const std::string &**name**)
  Access function that returns a pointer to an **FloatFun** *object with this name.*
- **IntFun** * **getObject** (const char ***name**)
  Access function that returns a pointer to an **NamedO** *object with this name.*

Protected Member Functions

- virtual ~**IntFun** ()

  Protected destructor ensures creation on the heap.

Related Functions

(Note that these are not member functions.)

- **IntFun** & **operator/** (int lhs_, const **IntFun** &rhs_)

  Scalar ratio of an IntFun.

- **IntFun** & **operator+** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Sum of two IntFun objects.

- **IntFun** & **operator+** (int lhs_, const **IntFun** &rhs_)

  Sum of a scalar and an IntFun object.

- **IntFun** & **operator+** (const **IntFun** &lhs_, int rhs_)

  Sum of an IntFun object and a scalar.

- **IntFun** & **operator-** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Difference of two IntFun objects.

- **IntFun** & **operator-** (int lhs_, const **IntFun** &rhs_)

  Difference of a scalar and an IntFun object.

- **IntFun** & **operator-** (const **IntFun** &lhs_, int rhs_)

  Difference of an IntFun object and a scalar.

- **IntFun** & **operator-** (const **IntFun** &arg_)

  Unary minus of a IntFun.

- **IntFun** & **operator+** (**IntFun** &arg_)

  Unary plus of a IntFun.

- **IntFun** & **operator** * (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Product of two IntFun objects.

- **IntFun** & **operator/** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Ratio of two IntFun objects.

- **IntFun** & **operator** * (int lhs_, const **IntFun** &rhs_)

  Scalar product of an IntFun.

- **IntFun** & **operator** * (const **IntFun** &lhs_, int rhs_)

  Scalar product of an IntFun.

- **IntFun** & **operator/** (const **IntFun** &lhs_, int rhs_)

Scalar ratio of an IntFun.

- **IntFun** & **operator &** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  And of two IntFun objects.

- **IntFun** & **operator|** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Or of two IntFun objects.

- **IntFun** & **operator^** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Xor of two IntFun objects.

- **IntFun** & **operator &** (const **IntFun** &lhs_, int rhs_)

  Bitwise And of an IntFun with a scalar.

- **IntFun** & **operator &** (int lhs_, const **IntFun** &rhs_)

  Bitwise And of an IntFun with a scalar.

- **IntFun** & **operator|** (const **IntFun** &lhs_, int rhs_)

  Bitwise Or of an IntFun with a scalar.

- **IntFun** & **operator|** (int lhs_, const **IntFun** &rhs_)

  Bitwise Or of an IntFun with a scalar.

- **IntFun** & **operator^** (const **IntFun** &lhs_, int rhs_)

  Bitwise Xor of an IntFun with a scalar.

- **IntFun** & **operator^** (int lhs_, const **IntFun** &rhs_)

  Bitwise Xor of an IntFun with a scalar.

- **IntFun** & **operator~** (const **IntFun** &arg_)

  Bitwise Complement of an IntFun.

- **IntFun** & **square** (const **IntFun** &arg_)

  Square of a IntFun.

- **IntFun** & **abs** (const **IntFun** &arg_)

  Absolute value of a IntFun.

- **IntFunComp** & **operator<** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator>** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator<=** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator>=** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator==** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator!=** (const **IntFun** &lhs_, const **IntFun** &rhs_)

  Comparison between two IntFun objects.

- **IntFunComp** & **operator<** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator<=** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator>** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator>=** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator==** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator!=** (int lhs_, const **IntFun** &rhs_)

  Comparison between a int and a IntFun.

- **IntFunComp** & **operator<** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator<=** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator>** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator>=** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator==** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator!=** (const **IntFun** &lhs_, int rhs_)

  Comparison between a IntFun and a int.

- **IntFunComp** & **operator<** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator<=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator>** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator>=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator==** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator!=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator<** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a IntFun object at the right side.

- **IntFunComp** & **operator<=** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a int at the right side.

- **IntFunComp** & **operator>** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a int at the right side.

- **IntFunComp** & **operator>=** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a int at the right side.

- **IntFunComp** & **operator==** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a int at the right side.

- **IntFunComp** & **operator!=** (const **IntFunComp** &lhs_, int rhs_)

  Chained comparison, with a int at the right side.

- **IntFun** & **max** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the maximum of all values of fun if cut is fulfilled.

- **IntFun** & **min** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the minimum of all values of fun if cut is fulfilled.

- **IntFun** & **sum** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the sum of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

- **IntFun** & **count** (const **BaseCutPoRConst** &cut=0, const **FillIteratorPoR** &intiter=0, const **FillIteratorPoRConst** &extiter=0)

  Returns the number of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

- **FloatFun** & **Float** (const **IntFun** &arg)

  Returns a **FloatFun** *object made out of the IntFun object.*

- **IntFun** & **cached** (const **ROListPoR** &rol, const **IntFunPoR** &fun, const **FillIterator-PoR** &iter=0)

  Returns a cached version of a **FloatFun**.

- **IntFun** & **poisson** (const **FloatFunPoR** &rnd, double mean)
  Returns a Poisson distributed value from a random function.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 IntFun::IntFun (const char * *name_* = "?") `[inline, explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:
*name_* The object's name

Definition at line 55 of file IntFunBase.h.

6.36.2.2 IntFun::IntFun (const std::string & *name_*) `[inline, explicit]`

Constructor from a C++ string.

Parameters:
*name_* The object's name

Definition at line 60 of file IntFunBase.h.

6.36.2.3 IntFun::IntFun (const std::string & *name_*, const std::string & *definition_*) `[inline]`

Constructor from two C++ strings.

Parameters:
*name_* The object's name
*definition_* The object's definition

Definition at line 65 of file IntFunBase.h.

6.36.3 Member Function Documentation

6.36.3.1 IntFun* IntFun::getObject (const std::string & *name*) `[inline, static]`

Access function that returns a pointer to an **FloatFun** object with this name.

This access function must never be called within the constructor of any class that is derived from IntFun!

Parameters:
name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 85 of file IntFunBase.h.

6.36.3.2 IntFun∗ IntFun::getObject (const char ∗ *name*) `[inline, static]`

Access function that returns a pointer to an **NamedO** object with this name.

This access function must never be called within the constructor of any class that is derived from NamedO!

Parameters:
name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 95 of file IntFunBase.h.

6.36.4 Friends And Related Function Documentation

6.36.4.1 IntFun & max (const IntFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the maximum of all values of fun if cut is fulfilled.

Parameters:
fun An optional IntFun
cut An optional cut
intiter An optional iterator over which to loop
extiter An optional external iterator

Definition at line 580 of file IntFun.C.

6.36.4.2 IntFun & min (const IntFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the minimum of all values of fun if cut is fulfilled.

Parameters:
fun An optional IntFun
cut An optional cut
intiter An optional iterator over which to loop
extiter An optional external iterator

Definition at line 588 of file IntFun.C.

6.36.4.3 IntFun & sum (const IntFunPoRConst & *fun*, const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the sum of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

Parameters:

fun An optional IntFun

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 596 of file IntFun.C.

6.36.4.4 IntFun & count (const BaseCutPoRConst & *cut* = 0, const FillIteratorPoR & *intiter* = 0, const FillIteratorPoRConst & *extiter* = 0) `[related]`

Returns the number of all values of fun if cut is fulfilled (if no fun is given, 1 is assumed).

Parameters:

cut An optional cut

intiter An optional iterator over which to loop

extiter An optional external iterator

Definition at line 604 of file IntFun.C.

6.36.4.5 IntFun & cached (const ROListPoR & *rol*, const IntFunPoR & *fun*, const FillIteratorPoR & *iter* = 0) `[related]`

Returns a cached version of a **FloatFun**.

Parameters:

rol Cached objects list where to register

fun An IntFun

iter An optional iterator

Definition at line 323 of file IntFun.C.

References getIterator(), FillIteratorPoR::pfi, and IntFunPoR::pif.

6.36.4.6 IntFun & poisson (const FloatFunPoR & *rnd*, double *mean*) `[related]`

Returns a Poisson distributed value from a random function.

Parameters:

rnd A random function

mean The mean value

Definition at line 308 of file RandomFuns.C.

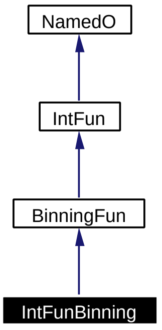
The documentation for this class was generated from the following files:

- **IntFunBase.h**
- **IntFun.C**
- **IntFun.h**
- **RandomFuns.h**

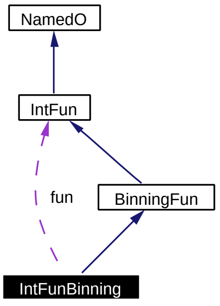
6.37 IntFunBinning Class Reference

`#include <IntFunBinning.h>`

Inheritance diagram for IntFunBinning:



Collaboration diagram for IntFunBinning:



6.37.1 Detailed Description

A class to get a concrete **BinningFun** subclass from a **IntFun**.

The class IntFunBinning can be used to generate a **BinningFun** in an easy way from a **IntFun**, which delivers the value that is used for binning, and some strings that are used to construct the name and title strings.

Author: Benno List, Jenny Böhme

Date
2004/08/04 11:33:52

Author
blist

Changelog:

- 4.8.04 BL: First version, based on **FloatFunBinning**

Definition at line 27 of file IntFunBinning.h.

Public Member Functions

- **IntFunBinning** (const **IntFunPoR** &fun_, int xlow_, int xhigh_, const char *varname_)

  Create an equidistant binning.

- virtual int **operator()** () const

  Returns a bin number between -1 and getNBins()-1.

- virtual const char * **getBinName** (int i) const

  Returns a name string.

- virtual const char * **getBinTitle** (int i) const

  Returns a title string.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ~**IntFunBinning** ()

  Protected destructor to ensure creation on the heap.

Protected Attributes

- const **IntFun** * **fun**

  The **IntFun** *object defining the quantity used for binning.*

- char * **varname**

  The variable name.

- unsigned int **varnamelength**

  The length of the variable name.

Private Member Functions

- void **initvarname** (const char *varname_)
- **IntFunBinning** (const **IntFunBinning** &)

  Private copy constructor, left unimplemented: No copying!

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

  Private assignment operator, left unimplemented: No assignment!

6.37.2 Constructor & Destructor Documentation

6.37.2.1 IntFunBinning::IntFunBinning (const IntFunPoR & *fun_*, int *xlow_*, int *xhigh_*, const char * *varname_*)

Create an equidistant binning.

The binning will have xhigh-xlow+1 bins between xlow and xhigh.

Parameters:

fun_ The **IntFun** object defining the quantity used for binning

xlow_ The lowest bin edge

xhigh_ The highest bin edge

varname_ The variable name

Definition at line 39 of file IntFunBinning.C.

6.37.3 Member Function Documentation

6.37.3.1 const char * IntFunBinning::getBinName (int *i*) const `[virtual]`

Returns a name string.

The name string MAY, and MUST, be destroyed by the user! The name will be (in perl notation) "$varname$bin"

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 88 of file IntFunBinning.C.

6.37.3.2 const char * IntFunBinning::getBinTitle (int *i*) const `[virtual]`

Returns a title string.

The name string MAY, and MUST, be destroyed by the user! The title will be (in perl notation) "$lower_edge < $varname < $higher_edge"

Parameters:

i The bin number

Implements **BinningFun** (p. 47).

Definition at line 71 of file IntFunBinning.C.

References varname, and varnamelength.

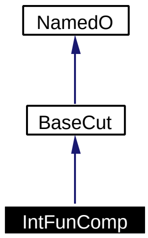
The documentation for this class was generated from the following files:

- **IntFunBinning.h**
- **IntFunBinning.C**

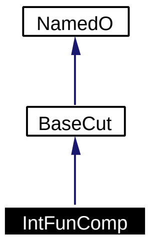
6.38 IntFunComp Class Reference

`#include <IntFun.h>`

Inheritance diagram for IntFunComp:



Collaboration diagram for IntFunComp:



6.38.1 Detailed Description

Abstract base class for comparisons between **IntFun** objects.

This class is the base class for the result of all comparisons like **IntFun** < **IntFun**. It allows to chain comparisons, i.e. write 1<x<2 instead of (1<x)&&(x<2).

Definition at line 79 of file IntFun.h.

Public Member Functions

- **IntFunComp** (const std::string &name_)
- virtual int **lhsValue** () const=0

  Returns value of left-hand side argument.

- virtual int **rhsValue** () const=0

  Returns value of left-hand side argument.

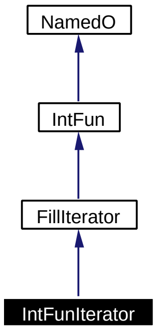
The documentation for this class was generated from the following file:

- **IntFun.h**

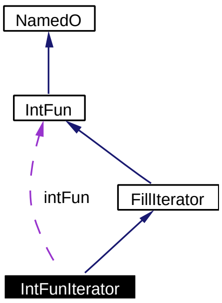
6.39 IntFunIterator Class Reference

`#include <IntFunIterator.h>`

Inheritance diagram for IntFunIterator:



Collaboration diagram for IntFunIterator:



6.39.1 Detailed Description

A **FillIterator** from an **IntFun** object.

An IntFunIterator is a **FillIterator** that uses an **IntFun** object to know how many objects (tracks, jets, D∗ candidates) there are to iterate over.

Author: Benno List, Jenny Böhme

Date

2005/06/09 16:42:59

Author

rweber

Changelog:

- 6.5.04 BL: Added name in constructor

Definition at line 33 of file IntFunIterator.h.

Public Member Functions

- **IntFunIterator** (const **IntFun** &intFun_)

  Constructor.

- **IntFunIterator** (const **IntFun** &intFun_, const std::string &name_)
- virtual int **operator()** () const

  Returns iterator value, starting at 0; "-1" means "invalid".

- virtual bool **next** ()

  Increments iterator; returns false if iterator cannot be incremented.

- virtual bool **reset** ()

  Resets iterator; returns false if iterator value range is empty.

- virtual bool **isValid** () const

  Returns whether current value of iterator is valid.

Protected Member Functions

- virtual ~**IntFunIterator** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const **IntFun** & **intFun**

  The intFun.

- int **index**

  The current index value.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 IntFunIterator::IntFunIterator (const IntFun & *intFun_*) `[inline]`

Constructor.

Parameters:

intFun_ The **IntFun** object

Definition at line 36 of file IntFunIterator.h.

References IntFun::getIterator(), NamedO::getName(), index, and intFun.

6.39.2.2 IntFunIterator::IntFunIterator (const IntFun & *intFun_*, const std::string & *name_*) [inline]

Parameters:

intFun_ The **IntFun** object

name_ The object's name

Definition at line 45 of file IntFunIterator.h.

References IntFun::getIterator(), index, and intFun.

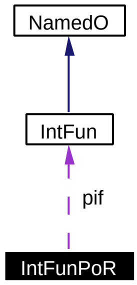
The documentation for this class was generated from the following file:

- **IntFunIterator.h**

6.40 IntFunPoR Class Reference

`#include <IntFunPoR.h>`

Collaboration diagram for IntFunPoR:



6.40.1 Detailed Description

IntFun Pointer or Reference.

Automagically converts between pointers and references to **IntFun** objects

Author: Jenny Böhme, Benno List

Date

2005/06/29 14:40:18

by:

Author

blist

Changelog:

- 15.7.04 JB: First version
- 20.7.04 BL: added constructor from **FillIteratorPoR**
- 29.5.05 BL: Make it possible to access a n **IntFun** also by its name

Definition at line 37 of file IntFunPoR.h.

Public Member Functions

- **IntFunPoR** (**IntFun** &if_)

  Constructor from a reference.

- **IntFunPoR** (**IntFun** *pif_)

  Constructor from a pointer.

- **IntFunPoR** (int=0)

Constructor from a int (will normally be 0!).

- **IntFunPoR** (const std::string &name)
  Constructor from a name.
- **IntFunPoR** (const char *name)
  Constructor from a name.
- **IntFunPoR** (const **IntFunPoR** &ifpor)
  Copy constructor.
- **IntFunPoR** (const **FillIteratorPoR** &fipor)
  Constructor from a **FillIteratorPoR**.

Public Attributes

- **IntFun * pif**
  The pointer to the **IntFun**.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 IntFunPoR::IntFunPoR (IntFun & *if_*) `[inline]`

Constructor from a reference.

Parameters:
if_ A reference to a **IntFun**

Definition at line 40 of file IntFunPoR.h.

References pif.

6.40.2.2 IntFunPoR::IntFunPoR (IntFun * *pif_*) `[inline]`

Constructor from a pointer.

Parameters:
pif_ A pointer to a **IntFun**

Definition at line 44 of file IntFunPoR.h.

References pif.

6.40.2.3 IntFunPoR::IntFunPoR (const std::string & *name*)

Constructor from a name.

Parameters:
name A name of a **FloatFun**

Definition at line 21 of file IntFunPoR.C.

6.40.2.4 IntFunPoR::IntFunPoR (const char * *name*)

Constructor from a name.

Parameters:

name A name of a **FloatFun**

Definition at line 24 of file IntFunPoR.C.

6.40.2.5 IntFunPoR::IntFunPoR (const IntFunPoR & *ifpor*) `[inline]`

Copy constructor.

Parameters:

ifpor A pointer-or-reference to an **IntFun**

Definition at line 60 of file IntFunPoR.h.

References pif.

6.40.2.6 IntFunPoR::IntFunPoR (const FillIteratorPoR & *fipor*) `[inline]`

Constructor from a **FillIteratorPoR**.

Parameters:

fipor A pointer-or-reference to a **FillIterator**

Definition at line 64 of file IntFunPoR.h.

References pif.

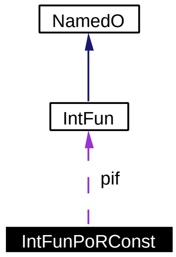
The documentation for this class was generated from the following files:

- **IntFunPoR.h**
- **IntFunPoR.C**

6.41 IntFunPoRConst Class Reference

`#include <IntFunPoR.h>`

Collaboration diagram for IntFunPoRConst:



6.41.1 Detailed Description

Constant **IntFun** Pointer or Reference.

Automagically converts between pointers and references to const **IntFun** objects

Author: Benno List, Jenny Boehme

Date
2005/06/29 14:40:18

by:

Author
blist

Changelog:

- 15.7.04 JB: First version
- 29.5.05 BL: Make it possible to access a n **IntFun** also by its name

Definition at line 86 of file IntFunPoR.h.

Public Member Functions

- **IntFunPoRConst** (const **IntFun** &if_)
  Constructor from a reference.
- **IntFunPoRConst** (const **IntFun** *pif_=0)
  Constructor from a pointer.
- **IntFunPoRConst** (const **IntFunPoRConst** &ifpor)
  Copy constructor.
- **IntFunPoRConst** (const **FillIteratorPoRConst** &fipor)

Constructor from a **FillIteratorPoR***.*

- **IntFunPoRConst** (int=0)
  Constructor from a int (will normally be 0!).
- **IntFunPoRConst** (const std::string &name)
  Constructor from a name.
- **IntFunPoRConst** (const char *name)
  Constructor from a name.

Public Attributes

- const **IntFun** * **pif**
  The pointer to the const **IntFun***.*

6.41.2 Constructor & Destructor Documentation

6.41.2.1 IntFunPoRConst::IntFunPoRConst (const IntFun & *if_*) `[inline]`

Constructor from a reference.

Parameters:
*if_* A reference to a const **IntFun**

Definition at line 89 of file IntFunPoR.h.

References pif.

6.41.2.2 IntFunPoRConst::IntFunPoRConst (const IntFun * *pif_* = 0) `[inline]`

Constructor from a pointer.

Parameters:
*pif_* A pointer to a const **IntFun**

Definition at line 93 of file IntFunPoR.h.

References pif.

6.41.2.3 IntFunPoRConst::IntFunPoRConst (const IntFunPoRConst & *ifpor*) `[inline]`

Copy constructor.

Parameters:
ifpor A pointer-or-reference to an **IntFun**

Definition at line 97 of file IntFunPoR.h.

References pif.

6.41.2.4 IntFunPoRConst::IntFunPoRConst (const FillIteratorPoRConst & *fipor*) [inline]

Constructor from a **FillIteratorPoR**.

Parameters:
fipor A pointer-or-reference to a **FillIterator**

Definition at line 101 of file IntFunPoR.h.

References pif.

6.41.2.5 IntFunPoRConst::IntFunPoRConst (const std::string & *name*)

Constructor from a name.

Parameters:
name A name of a **FloatFun**

Definition at line 27 of file IntFunPoR.C.

6.41.2.6 IntFunPoRConst::IntFunPoRConst (const char ∗ *name*)

Constructor from a name.

Parameters:
name A name of a **FloatFun**

Definition at line 30 of file IntFunPoR.C.

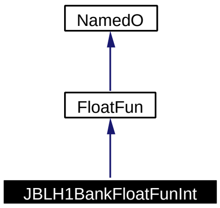
The documentation for this class was generated from the following files:

- **IntFunPoR.h**
- **IntFunPoR.C**

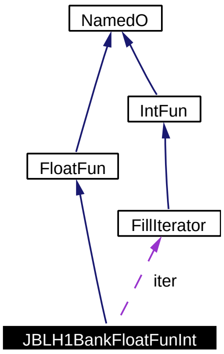
6.42 JBLH1BankFloatFunInt Class Reference

`#include <H1BankFuns.h>`

Inheritance diagram for JBLH1BankFloatFunInt:



Collaboration diagram for JBLH1BankFloatFunInt:



6.42.1 Detailed Description

Returns a Float_FF value from an int-valued column of an H1 bank row.

Definition at line 121 of file H1BankFuns.h.

Public Member Functions

- **JBLH1BankFloatFunInt** (const char ∗name_, const **FillIterator** &iter_, int column_=0, const char ∗attribname=0)

    Constructor by name.

- **JBLH1BankFloatFunInt** (const H1BankPtr &ptr_, const **FillIterator** &iter_, int column_=0, const char ∗attribname=0)

    Constructor from H1BankPtr.

- virtual Float_FF **operator()** () const

    Returns Float_FF value of selected column.

- virtual **JBLH1BankFloatFunInt** & **operator**[ ] (int newcolumn) const

Returns an JBLH1BankFloatFunInt that corresponds to a different column.

- virtual const **FillIterator** ∗ **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ∼**JBLH1BankFloatFunInt** ()

  Protected virtual destructor to ensure creation on the heap.

Private Attributes

- const H1BankPtr & **ptr**

  The bank pointer.

- const **FillIterator** & **iter**

  The iterator.

- int **column**

  The column (starting at 0!).

6.42.2 Constructor & Destructor Documentation

6.42.2.1 JBLH1BankFloatFunInt::JBLH1BankFloatFunInt (const char ∗ *name_*, const FillIterator & *iter_*, int *column_* = 0, const char ∗ *attribname* = 0) `[inline]`

Constructor by name.

Parameters:

name_ The bank name

iter_ The iterator

column_ The column (starting at 0!)

attribname Optional attribute name

Definition at line 124 of file H1BankFuns.h.

References column, NamedO::getName(), iter, ptr, and NamedO::str().

Referenced by operator[]().

6.42.2.2 JBLH1BankFloatFunInt::JBLH1BankFloatFunInt (const H1BankPtr & *ptr_*, const FillIterator & *iter_*, int *column_* = 0, const char ∗ *attribname* = 0) `[inline]`

Constructor from H1BankPtr.

Parameters:

ptr_ The bank pointer

iter_ The iterator

column_ The column (starting at 0!)

attribname Optional attribute name

Definition at line 136 of file H1BankFuns.h.

References column, NamedO::getName(), iter, ptr, and NamedO::str().

6.42.3 Member Function Documentation

6.42.3.1 virtual JBLH1BankFloatFunInt& JBLH1BankFloatFunInt::operator[] (int *newcolumn*) const `[inline, virtual]`

Returns an JBLH1BankFloatFunInt that corresponds to a different column.

Parameters:

newcolumn The new column

Definition at line 152 of file H1BankFuns.h.

References iter, JBLH1BankFloatFunInt(), and ptr.

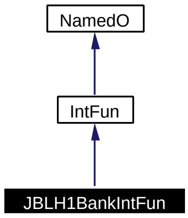
The documentation for this class was generated from the following file:

- **H1BankFuns.h**

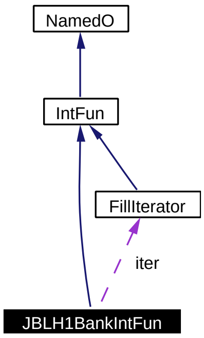
6.43 JBLH1BankIntFun Class Reference

`#include <H1BankFuns.h>`

Inheritance diagram for JBLH1BankIntFun:



Collaboration diagram for JBLH1BankIntFun:



6.43.1 Detailed Description

Returns a int value from an int-valued column of an H1 bank row.

Definition at line 170 of file H1BankFuns.h.

Public Member Functions

- **JBLH1BankIntFun** (const char *name_, const **FillIterator** &iter_, int column_=0, const char *attribname=0)

  Constructor by name.

- **JBLH1BankIntFun** (const H1BankPtr &ptr_, const **FillIterator** &iter_, int column_=0, const char *attribname=0)

  Constructor from H1BankPtr.

- virtual int **operator()** () const

  Returns int value of selected column.

- virtual **JBLH1BankIntFun** & **operator**[ ] (int newcolumn) const

  Returns an JBLH1BankIntFun that corresponds to a different column.

- virtual **JBLH1BankIntFun** & **Int** (int newcolumn) const
  Returns an JBLH1BankIntFun that corresponds to a different column.
- virtual JBLH1BankFloatFun & **Float** (int newcolumn) const
  Returns an JBLH1BankFloatFun that corresponds to a different column.
- virtual **JBLH1BankFloatFunInt** & **FloatFromInt** (int newcolumn) const
  Returns an **JBLH1BankFloatFunInt** *that corresponds to a different column.*
- virtual const **FillIterator** ∗ **getIterator** () const
  Returns pointer to iterator.

Protected Member Functions

- virtual ∼**JBLH1BankIntFun** ()
  Protected virtual destructor to ensure creation on the heap.

Private Attributes

- const H1BankPtr & **ptr**
  The bank pointer.
- const **FillIterator** & **iter**
  The iterator.
- int **column**
  The column (starting at 0!).

6.43.2 Constructor & Destructor Documentation

6.43.2.1 JBLH1BankIntFun::JBLH1BankIntFun (const char ∗ *name_*, const FillIterator & *iter_*, int *column_* = 0, const char ∗ *attribname* = 0) `[inline]`

Constructor by name.

Parameters:
- ***name_*** The bank name
- ***iter_*** The iterator
- ***column_*** The column (starting at 0!)
- ***attribname*** Optional attribute name

Definition at line 173 of file H1BankFuns.h.

References column, NamedO::getName(), Int(), iter, ptr, and NamedO::str().

Referenced by Int(), and operator[]().

6.43.2.2 JBLH1BankIntFun::JBLH1BankIntFun (const H1BankPtr & *ptr_*, const FillIterator & *iter_*, int *column_* = 0, const char * *attribname* = 0) [inline]

Constructor from H1BankPtr.

Parameters:

ptr_ The bank pointer

iter_ The iterator

column_ The column (starting at 0!)

attribname Optional attribute name

Definition at line 183 of file H1BankFuns.h.

References column, NamedO::getName(), Int(), iter, ptr, and NamedO::str().

6.43.3 Member Function Documentation

6.43.3.1 virtual JBLH1BankIntFun& JBLH1BankIntFun::operator[] (int *newcolumn*) const [inline, virtual]

Returns an JBLH1BankIntFun that corresponds to a different column.

Parameters:

newcolumn The new column

Definition at line 200 of file H1BankFuns.h.

References iter, JBLH1BankIntFun(), and ptr.

6.43.3.2 virtual JBLH1BankIntFun& JBLH1BankIntFun::Int (int *newcolumn*) const [inline, virtual]

Returns an JBLH1BankIntFun that corresponds to a different column.

Parameters:

newcolumn The new column

Definition at line 205 of file H1BankFuns.h.

References iter, JBLH1BankIntFun(), and ptr.

Referenced by JBLH1BankIntFun().

6.43.3.3 virtual JBLH1BankFloatFun& JBLH1BankIntFun::Float (int *newcolumn*) const [inline, virtual]

Returns an JBLH1BankFloatFun that corresponds to a different column.

Parameters:

newcolumn The new column

Definition at line 210 of file H1BankFuns.h.

References iter, and ptr.

6.43.3.4 virtual JBLH1BankFloatFunInt& JBLH1BankIntFun::FloatFromInt (int *newcolumn*) const [inline, virtual]

Returns an **JBLH1BankFloatFunInt** that corresponds to a different column.

Parameters:

newcolumn The new column

Definition at line 215 of file H1BankFuns.h.

References iter, and ptr.

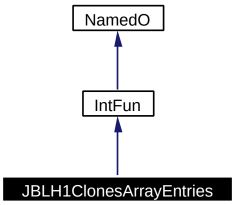
The documentation for this class was generated from the following file:

- **H1BankFuns.h**

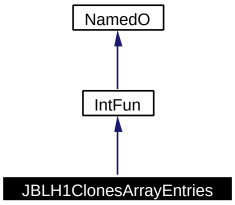
6.44 JBLH1ClonesArrayEntries Class Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1ClonesArrayEntries:



Collaboration diagram for JBLH1ClonesArrayEntries:



6.44.1 Detailed Description

An **IntFun** that returns the number of entries in an H1ClonesArray.

Definition at line 429 of file H1Funs.h.

Public Member Functions

- **JBLH1ClonesArrayEntries** (const char ∗branchname_)

  Constructor from a branch name.

- **JBLH1ClonesArrayEntries** (const H1ClonesArrayPtr &ptr_)

  Constructor from an existing pointer.

- virtual int **operator()** () const

  Returns number of entries in the H1ClonesArray.

Protected Member Functions

- virtual ∼**JBLH1ClonesArrayEntries** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- const H1ClonesArrayPtr & **ptr**

    The pointer to the H1ClonesArray.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 JBLH1ClonesArrayEntries::JBLH1ClonesArrayEntries (const char * *branchname_*) [inline, explicit]

Constructor from a branch name.

Parameters:

branchname_ The branch name

Definition at line 432 of file H1Funs.h.

References ptr.

6.44.2.2 JBLH1ClonesArrayEntries::JBLH1ClonesArrayEntries (const H1ClonesArrayPtr & *ptr_*) [inline, explicit]

Constructor from an existing pointer.

Parameters:

ptr_ The branch pointer

Definition at line 440 of file H1Funs.h.

References ptr.

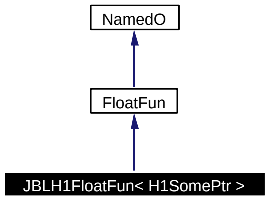
The documentation for this class was generated from the following file:

- **H1Funs.h**

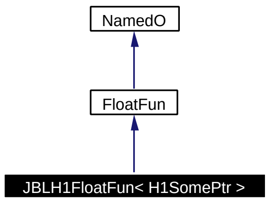
6.45 JBLH1FloatFun< H1SomePtr > Class Template Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1FloatFun< H1SomePtr >:



Collaboration diagram for JBLH1FloatFun< H1SomePtr >:



6.45.1 Detailed Description

template<class H1SomePtr> class JBLH1FloatFun< H1SomePtr >

A **FloatFun** that returns a variable from an H1Tree.

The template argument H1SomePtr is the H1 Pointer type that has to be used to retrieve the value, such as H1FloatPtr or H1IntPtr.

An optional index can be used if one wants to access a single element of a fixed size array, such as for subtriggers.

Definition at line 112 of file H1Funs.h.

Public Member Functions

- **JBLH1FloatFun** (const char *branchname_, int index_=-1)

  Constructor from a branch name.

- **JBLH1FloatFun** (const H1SomePtr &ptr_, int index_=-1, const char *pointername_=0)

  Constructor from an existing pointer.

- Float_FF **operator()** () const

  Returns the value of the variable.

- **JBLH1FloatFun**< H1SomePtr > & **operator**[] (int index_)

  Returns a version with a fixed index.

- **JBLH1FloatFunIter**< H1SomePtr > & **operator**[] (const **FillIterator** &iter_)

  Returns a version with an iterator.

Protected Member Functions

- virtual ~**JBLH1FloatFun** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- const H1SomePtr & **ptr**

  The H1 pointer.

- int **index**

  The optional index.

- std::string **branchname**

  The branch name.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 template<class H1SomePtr> JBLH1FloatFun< H1SomePtr >::JBLH1FloatFun (const char * *branchname_*, int *index_* = -1) `[inline, explicit]`

Constructor from a branch name.

Parameters:

branchname_ The branch name

index_ An optional index

Definition at line 115 of file H1Funs.h.

References JBLH1FloatFun< H1SomePtr >::branchname, JBLH1FloatFun< H1SomePtr >::index, JBLH1FloatFun< H1SomePtr >::ptr, and NamedO::str().

6.45.2.2 template<class H1SomePtr> JBLH1FloatFun< H1SomePtr >::JBLH1FloatFun (const H1SomePtr & *ptr_*, int *index_* = -1, const char * *pointername_* = 0) `[inline, explicit]`

Constructor from an existing pointer.

Parameters:

ptr_ The branch pointer

index_ An optional index

pointername_ An optional pointer name

Definition at line 126 of file H1Funs.h.

References JBLH1FloatFun< H1SomePtr >::branchname, JBLH1FloatFun< H1SomePtr >::index, JBLH1FloatFun< H1SomePtr >::ptr, and NamedO::str().

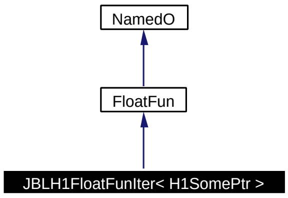
The documentation for this class was generated from the following file:

- **H1Funs.h**

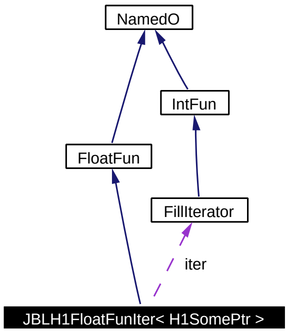
6.46 JBLH1FloatFunIter< H1SomePtr > Class Template Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1FloatFunIter< H1SomePtr >:



Collaboration diagram for JBLH1FloatFunIter< H1SomePtr >:



6.46.1 Detailed Description

template<class H1SomePtr> class JBLH1FloatFunIter< H1SomePtr >

A **FloatFun** using an iterator that returns a variable from an H1Tree.

The template argument H1SomePtr is the H1 Pointer type that has to be used to retrieve the value, such as H1FloatPtr or H1IntPtr.

Definition at line 55 of file H1Funs.h.

Public Member Functions

- **JBLH1FloatFunIter** (const char *branchname_, const **FillIterator** &iter_)

  Constructor from a branch name.

- **JBLH1FloatFunIter** (const H1SomePtr &ptr_, const **FillIterator** &iter_, const char *pointername_=0)

  Constructor from an existing pointer.

- Float_FF **operator()** () const

Returns the value of the variable, using the iterator.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**JBLH1FloatFunIter** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- const H1SomePtr & **ptr**

  The H1 pointer.

- const **FillIterator** & **iter**

  The iterator.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 template<class H1SomePtr> JBLH1FloatFunIter< H1SomePtr >::JBLH1FloatFunIter (const char * *branchname_*, const FillIterator & *iter_*) `[inline]`

Constructor from a branch name.

Parameters:

branchname_ The branch name

iter_ The iterator

Definition at line 58 of file H1Funs.h.

References NamedO::getName(), JBLH1FloatFunIter< H1SomePtr >::iter, and JBLH1FloatFunIter< H1SomePtr >::ptr.

6.46.2.2 template<class H1SomePtr> JBLH1FloatFunIter< H1SomePtr >::JBLH1FloatFunIter (const H1SomePtr & *ptr_*, const FillIterator & *iter_*, const char * *pointername_* = 0) `[inline]`

Constructor from an existing pointer.

Parameters:

ptr_ The branch pointer

iter_ The iterator

pointername_ An optional pointer name

Definition at line 69 of file H1Funs.h.

References NamedO::getName(), JBLH1FloatFunIter< H1SomePtr >::iter, and JBLH1FloatFunIter< H1SomePtr >::ptr.

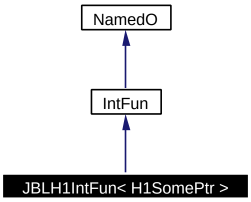
The documentation for this class was generated from the following file:

- **H1Funs.h**

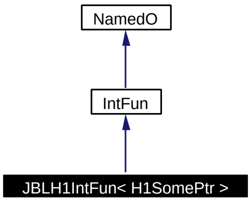
6.47 JBLH1IntFun< H1SomePtr > Class Template Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1IntFun< H1SomePtr >:



Collaboration diagram for JBLH1IntFun< H1SomePtr >:



6.47.1 Detailed Description

template<class H1SomePtr> class JBLH1IntFun< H1SomePtr >

A **IntFun** that returns a variable from an H1Tree.

The template argument H1SomePtr is the H1 Pointer type that has to be used to retrieve the value, such as H1IntPtr or H1ShortPtr.

An optional index can be used if one wants to access a single element of a fixed size array, such as for subtriggers.

Definition at line 233 of file H1Funs.h.

Public Member Functions

- **JBLH1IntFun** (const char *branchname_, int index_=-1)

  Constructor from a branch name.

- **JBLH1IntFun** (const H1SomePtr &ptr_, int index_=-1)

  Constructor from an existing pointer.

- int **operator()** () const

  Returns the value of the variable.

- **JBLH1IntFun**< H1SomePtr > & **operator**[ ] (int index_)

  Returns a version with a fixed index.

- **JBLH1IntFunIter**< H1SomePtr > & **operator**[ ] (const **FillIterator** &iter_)

  Returns a version with an iterator.

- **FloatFun** & **Float** () const

  Returns a **FloatFun** *object.*

Protected Member Functions

- virtual ∼**JBLH1IntFun** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- H1SomePtr **ptr**

  The H1 pointer.

- int **index**

  The optional index.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 template<class H1SomePtr> JBLH1IntFun< H1SomePtr >::JBLH1IntFun (const char * *branchname_*, int *index_* = -1) `[inline, explicit]`

Constructor from a branch name.

Parameters:

branchname_ The branch name

index_ An optional index

Definition at line 236 of file H1Funs.h.

References JBLH1IntFun< H1SomePtr >::index, JBLH1IntFun< H1SomePtr >::ptr, and Named-O::str().

6.47.2.2 template<class H1SomePtr> JBLH1IntFun< H1SomePtr >::JBLH1IntFun (const H1SomePtr & *ptr_*, int *index_* = -1) `[inline, explicit]`

Constructor from an existing pointer.

Parameters:

ptr_ The branch pointer

index_ An optional index

Definition at line 244 of file H1Funs.h.

References JBLH1IntFun< H1SomePtr >::index, JBLH1IntFun< H1SomePtr >::ptr, and Named-O::str().

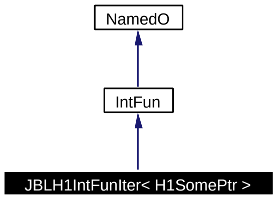
The documentation for this class was generated from the following file:

- **H1Funs.h**

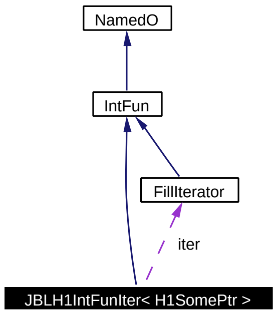
6.48 JBLH1IntFunIter< H1SomePtr > Class Template Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1IntFunIter< H1SomePtr >:



Collaboration diagram for JBLH1IntFunIter< H1SomePtr >:



6.48.1 Detailed Description

template<class H1SomePtr> class JBLH1IntFunIter< H1SomePtr >

A **IntFun** using an iterator that returns a variable from an H1Tree.

The template argument H1SomePtr is the H1 Pointer type that has to be used to retrieve the value, such as H1IntPtr or H1ShortPtr.

Definition at line 176 of file H1Funs.h.

Public Member Functions

- **JBLH1IntFunIter** (const char *branchname_, const **FillIterator** &iter_)

  Constructor from a branch name.

- **JBLH1IntFunIter** (const H1SomePtr &ptr_, const **FillIterator** &iter_)

  Constructor from an existing pointer.

- int **operator()** () const

  Returns the value of the variable, using the iterator.

- virtual const **FillIterator** ∗ **getIterator** () const
  Returns pointer to iterator.

Protected Member Functions

- virtual ∼**JBLH1IntFunIter** ()
  Protected destructor to ensure creation on the heap.

Private Attributes

- const H1SomePtr & **ptr**
  The H1 pointer.
- const **FillIterator** & **iter**
  The iterator.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 template<class H1SomePtr> JBLH1IntFunIter< H1SomePtr >::JBLH1IntFunIter (const char ∗ *branchname_*, const FillIterator & *iter_*) `[inline]`

Constructor from a branch name.

Parameters:
branchname_ The branch name
iter_ The iterator

Definition at line 179 of file H1Funs.h.

References NamedO::getName(), JBLH1IntFunIter< H1SomePtr >::iter, and JBLH1IntFunIter< H1SomePtr >::ptr.

6.48.2.2 template<class H1SomePtr> JBLH1IntFunIter< H1SomePtr >::JBLH1IntFunIter (const H1SomePtr & *ptr_*, const FillIterator & *iter_*) `[inline]`

Constructor from an existing pointer.

Parameters:
ptr_ The branch pointer
iter_ The iterator

Definition at line 190 of file H1Funs.h.

References NamedO::getName(), JBLH1IntFunIter< H1SomePtr >::iter, and JBLH1IntFunIter< H1SomePtr >::ptr.

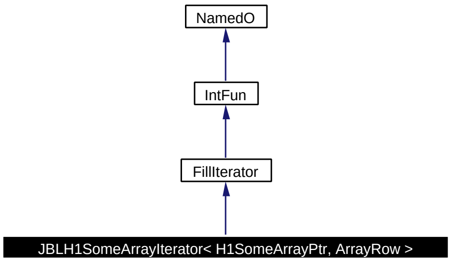
The documentation for this class was generated from the following file:

- **H1Funs.h**

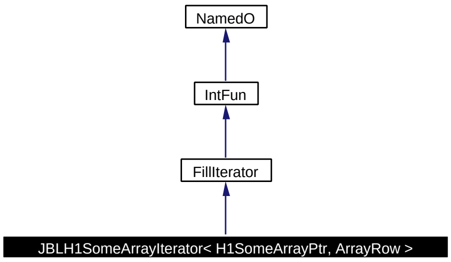
6.49 JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow > Class Template Reference

`#include <H1Funs.h>`

Inheritance diagram for JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >:



Collaboration diagram for JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >:



6.49.1 Detailed Description

template<class H1SomeArrayPtr = H1ClonesArrayPtr, class ArrayRow = H1Part> class JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >

A **FillIterator** for an H1ClonesArray.

H1 uses many H1ClonesArrayPtr objects to store data on mODS, such as H1PartSelTrackPtr, H1PartCandArrayPtr etc. An JBLH1SomeArrayIterator is a convenient way to define an iterator for such an array.

Changelog:

- 22.12.04 BL: Added "isValid"

Definition at line 297 of file H1Funs.h.

Public Member Functions

- **JBLH1SomeArrayIterator** ()

  Default constructor.

- **JBLH1SomeArrayIterator** (const char *branchname_)

  Constructor from a branch name.

- **JBLH1SomeArrayIterator** (const H1SomeArrayPtr &ptr_, const char *pointername_-=0)

  Constructor from an existing pointer.

- virtual int **operator()** () const

  Returns iterator value, starting at 0.

- virtual bool **next** ()

  Increments iterator; returns false if iterator cannot be incremented.

- virtual bool **reset** ()

  Resets iterator; returns false if iterator value range is empty.

- virtual bool **isValid** () const

  Returns whether current value of iterator is valid.

- virtual **FloatFun** & **GetP** () const

  Returns a **FloatFun** *object that gives the momentum P.*

- virtual **FloatFun** & **GetPt** () const

  Returns a **FloatFun** *object that gives the transverse momentum Pt.*

- virtual **FloatFun** & **GetPz** () const

  Returns a **FloatFun** *object that gives the longitudinal momentum Pz.*

- virtual **FloatFun** & **GetE** () const

  Returns a **FloatFun** *object that gives the energy E.*

- virtual **FloatFun** & **GetPhi** () const

  Returns a **FloatFun** *object that gives the azimuthal angle Phi.*

- virtual **FloatFun** & **GetTheta** () const

  Returns a **FloatFun** *object that gives the polar angle Theta.*

- virtual **FloatFun** & **GetMass** () const

  Returns a **FloatFun** *object that gives the Mass.*

- virtual **FloatFun** & **GetCharge** () const

  Returns a **FloatFun** *object that gives the Charge.*

- const char * **getBranchName** () const
- template<class H1PartSome, class TM> **FloatFun** & **Float** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns a **FloatFun** *object.*

- template<class H1PartSome, class TM> **FloatFun** & **GetFloatFun** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns a **FloatFun** *object.*

- template<class H1PartSome, class TM> **IntFun** & **Int** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns an **IntFun** *object.*

- template<class H1PartSome, class TM> **IntFun** & **GetIntFun** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns an **IntFun** *object.*

- template<class H1PartSome, class TM> **BaseCut** & **Bool** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns a **BaseCut** *object.*

- template<class H1PartSome, class TM> **BaseCut** & **GetBaseCut** (TM(H1PartSome::*p_mem)() const, const char *attribname="?") const

  Returns a **BaseCut** *object.*

- const H1SomeArrayPtr & **getBranchPointer** () const

  Returns the branch pointer.

Protected Member Functions

- virtual ~**JBLH1SomeArrayIterator** ()

  Protected destructor to ensure creation on the heap.

Private Attributes

- const H1SomeArrayPtr & **ptr**

  The pointer to the H1ClonesArray.

- int **index**

  The index.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 template<class H1SomeArrayPtr = H1ClonesArrayPtr, class ArrayRow = H1Part> JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::JBLH1SomeArrayIterator (const char * *branchname_*) `[inline, explicit]`

Constructor from a branch name.

Parameters:

branchname_ The branch name

Definition at line 311 of file H1Funs.h.

References NamedO::getName(), JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::index, and JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::ptr.

6.49.2.2 template<class H1SomeArrayPtr = H1ClonesArrayPtr, class ArrayRow = H1Part> JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::JBLH1SomeArrayIterator (const H1SomeArrayPtr & *ptr_*, const char * *pointername_* = 0) [inline, explicit]

Constructor from an existing pointer.

Parameters:

ptr_ The branch pointer

pointername_ An optional pointer branch name

Definition at line 322 of file H1Funs.h.

References NamedO::getName(), JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::index, and JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::ptr.

6.49.3 Member Function Documentation

6.49.3.1 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> FloatFun & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::Float (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns a **FloatFun** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 698 of file H1Funs.h.

6.49.3.2 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> FloatFun & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::GetFloatFun (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns a **FloatFun** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 703 of file H1Funs.h.

6.49.3.3 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> IntFun & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::Int (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns an **IntFun** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 709 of file H1Funs.h.

6.49.3.4 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> IntFun & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::GetIntFun (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns an **IntFun** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 714 of file H1Funs.h.

6.49.3.5 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> BaseCut & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::Bool (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns a **BaseCut** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 720 of file H1Funs.h.

6.49.3.6 template<class H1SomeArrayPtr, class ArrayRow> template<class H1PartSome, class TM> BaseCut & JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::GetBaseCut (TM(H1PartSome::* *p_mem*)() const, const char * *attribname* = "?") const [inline]

Returns a **BaseCut** object.

Parameters:

p_mem The access method

attribname An optional attribute name

Definition at line 725 of file H1Funs.h.

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

- **H1Funs.h**

6.50 JBLH1SomeBank< H1SomeBankPtr, H1SomeRow > Class Template Reference

`#include <H1BankFuns.h>`

6.50.1 Detailed Description

template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> class JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >

A class that allows generic access to an H1 bank.

An JBLH1Bank represents an H1 bank as object. An JBLH1Bank object can provide an iterator (of type JBLH1BankIterator) that can be used to iterate over the rows of the bank.

Definition at line 333 of file H1BankFuns.h.

Public Member Functions

- **JBLH1SomeBank** ()
  Default constructor.
- **JBLH1SomeBank** (const char *name_)
  Constructor by name.
- **JBLH1SomeBank** (const H1SomeBankPtr &ptr_)
  Constructor from H1BankPtr.
- virtual **JBLH1SomeBankIterator**< H1SomeBankPtr, H1SomeRow > & **iter** () const
  Returns an JBLH1BankIterator to iterate over the rows of the bank.
- virtual const H1SomeBankPtr & **getH1BankPtr** () const
  Returns the bank pointer.
- virtual const char * **getName** () const
  Returns pointer to name.

Protected Member Functions

- virtual ~**JBLH1SomeBank** ()
  Protected virtual destructor to ensure creation on the heap.

Private Attributes

- const H1SomeBankPtr & **ptr**
  The bank pointer.

- std::string **name**

  A name (good for debugging).

6.50.2 Constructor & Destructor Documentation

6.50.2.1 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::JBLH1SomeBank (const char * *name_*) `[inline, explicit]`

Constructor by name.

Parameters:

name_ The bank name

Definition at line 343 of file H1BankFuns.h.

References JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::name, and JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::ptr.

6.50.2.2 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::JBLH1SomeBank (const H1SomeBankPtr & *ptr_*) `[inline, explicit]`

Constructor from H1BankPtr.

Parameters:

ptr_ The bank pointer

Definition at line 350 of file H1BankFuns.h.

References JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::name, and JBLH1SomeBank< H1SomeBankPtr, H1SomeRow >::ptr.

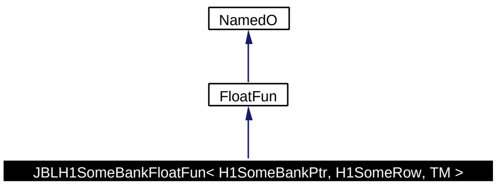
The documentation for this class was generated from the following file:

- **H1BankFuns.h**

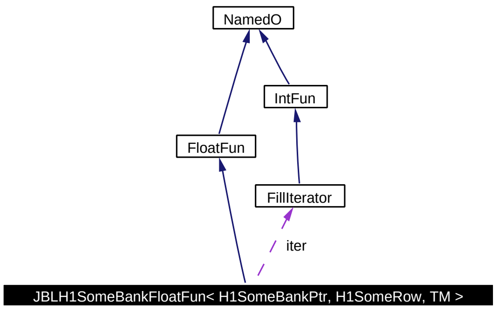
6.51 JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM > Class Template Reference

`#include <H1BankFuns.h>`

Inheritance diagram for JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >:



Collaboration diagram for JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >:



6.51.1 Detailed Description

template<class H1SomeBankPtr, class H1SomeRow, class TM> class JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >

Returns a Float_FF value from an float-valued column of an H1 bank row.

Definition at line 32 of file H1BankFuns.h.

Public Member Functions

- **JBLH1SomeBankFloatFun** (const H1SomeBankPtr &ptr_, const **FillIterator** &iter_, TM(H1SomeRow::*ptr_mem_)() const, const char *attribname=0)

  Constructor from H1BankPtr.

- virtual Float_FF **operator()** () const

  Returns Float_FF value of selected column.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**JBLH1SomeBankFloatFun** ()

  Protected virtual destructor to ensure creation on the heap.

Private Attributes

- const H1SomeBankPtr & **ptr**

  The bank pointer.

- const **FillIterator** & **iter**

  The iterator.

- TM(H1SomeRow::* **ptr_mem**)() const

6.51.2 Constructor & Destructor Documentation

6.51.2.1 template<class H1SomeBankPtr, class H1SomeRow, class TM> JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >::JBLH1SomeBankFloatFun (const H1SomeBankPtr & *ptr_*, const FillIterator & *iter_*, TM(H1SomeRow::* *ptr_mem_*)() const, const char * *attribname* = 0) `[inline]`

Constructor from H1BankPtr.

Parameters:

ptr_ The bank pointer

iter_ The iterator

ptr_mem_ The access method

attribname Optional attribute name

Definition at line 35 of file H1BankFuns.h.

References NamedO::getName(), JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >::iter, and JBLH1SomeBankFloatFun< H1SomeBankPtr, H1SomeRow, TM >::ptr.

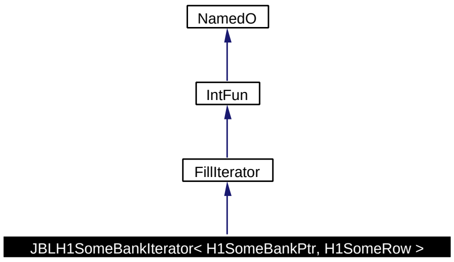
The documentation for this class was generated from the following file:

- **H1BankFuns.h**

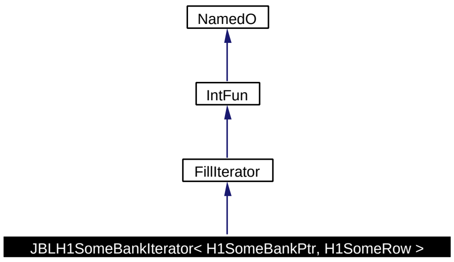
6.52 JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow > Class Template Reference

`#include <H1BankFuns.h>`

Inheritance diagram for JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >:



Collaboration diagram for JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >:



6.52.1 Detailed Description

template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> class JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >

A **FillIterator** for an H1 bank.

An JBLH1BankIterator iteratoes over the number of entries of an H1 bank.

Special methods are provided to generate **FloatFun** and **IntFun** objects that can be used to access the values of individual columns: GetFloat, GetFloatFromInt and GetInt return such function objects.

Definition at line 241 of file H1BankFuns.h.

Public Member Functions

- **JBLH1SomeBankIterator** ()

  Default constructor.

- **JBLH1SomeBankIterator** (const char *name_)

  Constructor by name.

- **JBLH1SomeBankIterator** (const H1SomeBankPtr &ptr_)

  Constructor from H1BankPtr.

- virtual int **operator()** () const

  Returns iterator value, starting at 0.

- virtual bool **next** ()

  Increments iterator; returns false if iterator cannot be incremented.

- virtual bool **reset** ()

  Resets iterator; returns false if iterator value range is empty.

- virtual bool **isValid** () const

  Returns whether current value of iterator is valid.

- virtual JBLH1BankFloatFun & **GetFloat** (int column, const char *attribname=0)

  Returns an JBLH1BankFloatFun that represents the value of a Float_FF-valued column.

- template<class TM> **JBLH1SomeBankFloatFun**< H1SomeBankPtr, H1SomeRow, TM > & **GetFloat** (TM(H1SomeRow::*ptr_mem)() const, const char *attribname=0)

  Returns a **FloatFun** *that represents the value of an access method.*

- virtual **JBLH1BankFloatFunInt** & **GetFloatFromInt** (int column)

  Returns an **JBLH1BankFloatFunInt** *that represents the value of a int-valued column as Float_FF.*

- virtual **JBLH1BankIntFun** & **GetInt** (int column)

  Returns an JBLH1BankInttFun that represents the value of a int-valued column.

Protected Member Functions

- virtual ~**JBLH1SomeBankIterator** ()

  Protected virtual destructor to ensure creation on the heap.

Private Attributes

- const H1SomeBankPtr & **ptr**

  The pointer to the H1 bank.

- int **index**

  The index, i.e. the row number.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::JBLH1SomeBankIterator (const char * *name_*) `[inline, explicit]`

Constructor by name.

Parameters:

name_ The bank name

Definition at line 253 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::index, and JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

6.52.2.2 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::JBLH1SomeBankIterator (const H1SomeBankPtr & *ptr_*) `[inline, explicit]`

Constructor from H1BankPtr.

Parameters:

ptr_ The bank pointer

Definition at line 262 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::index, and JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

6.52.3 Member Function Documentation

6.52.3.1 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> virtual JBLH1BankFloatFun& JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::GetFloat (int *column*, const char * *attribname* = 0) `[inline, virtual]`

Returns an JBLH1BankFloatFun that represents the value of a Float_FF-valued column.

Parameters:

column The column

attribname Optional attribute name

Definition at line 287 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

6.52.3.2 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> template<class TM> JBLH1SomeBankFloatFun<H1SomeBankPtr, H1SomeRow, TM>& JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::GetFloat (TM(H1SomeRow::∗ *ptr_mem*)() const, const char ∗ *attribname* = 0) `[inline]`

Returns a **FloatFun** that represents the value of an access method.

Parameters:

ptr_mem The access method

attribname Optional attribute name

Definition at line 296 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

6.52.3.3 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> virtual JBLH1BankFloatFunInt& JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::GetFloatFromInt (int *column*) `[inline, virtual]`

Returns an **JBLH1BankFloatFunInt** that represents the value of a int-valued column as Float_FF.

Parameters:

column The column

Definition at line 302 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

6.52.3.4 template<class H1SomeBankPtr = H1BankPtr, class H1SomeRow = H1Row> virtual JBLH1BankIntFun& JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::GetInt (int *column*) `[inline, virtual]`

Returns an JBLH1BankInttFun that represents the value of a int-valued column.

Parameters:

column The column

Definition at line 307 of file H1BankFuns.h.

References JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::ptr.

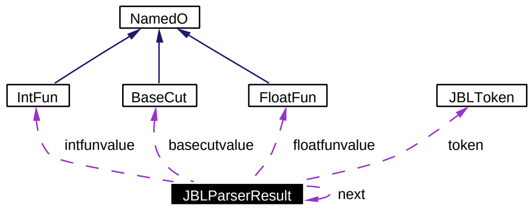
The documentation for this class was generated from the following file:

- **H1BankFuns.h**

6.53 JBLParserResult Class Reference

`#include <JBLParserResult.h>`

Collaboration diagram for JBLParserResult:



6.53.1 Detailed Description

Returns the result of parsing a string that defines a function object.

This class is used to parse a string that defines a function object, be it an **IntFun**, a **FloatFun**, or a **BaseCut**.

The nomenclature of the various subroutines follows the syntax given in appendix A13 of B.W.KERNIGHAN & D.M.RITCHIE: The C programming language (2nd ed). - London (Prentice Hall) 1988.

Author: Benno List, Jenny List

Date
2005/07/21 11:20:12

Author
blist

Changelog:

- 31.5.05 BL: First version
- 1.6.05 BL: Added NOT operator, clean up, better comparison operators
- 6.6.05 BL: Added methods parseExpression (std::istream& is), print, and operators<< and >>, fixed bug affecting argument lists longer than 2
- 8.6.05 BL: Bug fixes, simplifications
- 14.6.05 BL: Added automatic initialization, moved maps completely into **JBLParserResult.C**

Definition at line 46 of file JBLParserResult.h.

Public Types

- typedef Float_FF(∗ **Fun_f_f**)(Float_FF)
- typedef Float_FF(∗ **Fun_f_f_f**)(Float_FF, Float_FF)

- typedef Float_FF(∗ **Fun_f_f_i**)(Float_FF, Float_FF)
- typedef int(∗ **Fun_i_i**)(int)
- typedef bool(∗ **Fun_b_b**)(bool)
- typedef **FloatFun** &(∗ **Fun_FF_FF**)(const **FloatFun** &)
- typedef **FloatFun** &(∗ **Fun_FF_FF_FF**)(const **FloatFun** &, const **FloatFun** &)
- typedef **FloatFun** &(∗ **Fun_FF_FF_IF**)(const **FloatFun** &, const **IntFun** &)
- typedef **IntFun** &(∗ **Fun_IF_IF**)(const **IntFun** &)
- typedef **BaseCut** &(∗ **Fun_BC_BC**)(const **BaseCut** &)

Public Member Functions

- **JBLParserResult** ()

  Default constructor (type EMPTY).

- **JBLParserResult** (const **JBLParserResult** &rhs)

  Copy Constructor.

- ∼**JBLParserResult** ()

  Destructor.

- **JBLParserResult** & **operator=** (const **JBLParserResult** &rhs)

  Assignment operator.

- bool **isEmpty** () const

  Signals whether object is empty.

- bool **isNumber** () const

  Signals whether object is a number (int or float).

- bool **isInt** () const

  Signals whether object is an integer.

- bool **isFloat** () const

  Signals whether object is a float.

- bool **isBool** () const

  Signals whether object is a bool.

- bool **isIntFun** () const

  Signals whether object is an **IntFun**.

- bool **isFloatFun** () const

  Signals whether object is a **FloatFun**.

- bool **isFun** () const

  *Signals whether object is any function object (***IntFun** *or* **FloatFun***).*

- bool **isBaseCut** () const

  Signals whether object is a **BaseCut**.

- bool **isIdentifier** () const

  Signals whether object is an identifier.

- int **listlength** () const

  Gives back the list of the argument list.

- const std::string & **getName** () const

  Returns the name.

- int **getIntValue** () const

  Returns the integer value (0 if not of type INT).

- Float_FF **getFloatValue** () const

  Returns the Float_FF value (0 if not of type INT or FLOAT).

- bool **getBoolValue** () const

  Returns the bool value (false if not of type BOOL).

- **IntFun** ∗ **getIntFunValue** () const

  Returns the **IntFun** *value (0 if not of type INT or INTFUN).*

- **FloatFun** ∗ **getFloatFunValue** () const

  Returns the **FloatFun** *value (0 if not of type INT or FLOAT or INTFUN or FLOATFUN).*

- **BaseCut** ∗ **getBaseCutValue** () const

  Returns the **BaseCut** *value (0 if not of type BOOL or BASECUT).*

- std::ostream & **print** (std::ostream &os) const

  Print the object's definition.

- **JBLParserResult** & **addCopyToList** (const **JBLParserResult** &rhs)

  Add a copy of rhs to the end of the argument list.

Static Public Member Functions

- **JBLParserResult parseExpression** (const std::string &input)

  Parses a string expression.

- **JBLParserResult parseExpression** (std::istream &is)

  Parses a string expression.

- void **initMaps** ()

Private Types

- enum **Type** {
  EMPTY = 0, **INT** = 1, **FLOAT** = 2, **NUMBER** = 3,
  BOOL = 4, **INTFUN** = 8, **FLOATFUN** = 16, **FUN** = 24,
  BASECUT = 32, **IDENTIFIER** = 64 }

Private Member Functions

- **JBLParserResult** (int intvalue_)
  Constructor from an int (type INT).
- **JBLParserResult** (Float_FF floatvalue_)
  Constructor from a Float_FF (type FLOAT).
- **JBLParserResult** (bool boolvalue_)
  Constructor from a bool (type BOOL).
- **JBLParserResult** (**IntFun** &intfunvalue_)
  Constructor from an **IntFun** *(type INTFUN).*
- **JBLParserResult** (**FloatFun** &floatfunvalue_)
  Constructor from a **FloatFun** *(type FLOATFUN).*
- **JBLParserResult** (**BaseCut** &basecutvalue_)
  Constructor from a **BaseCut** *(type BASECUT).*
- **JBLParserResult** (const std::string &name_)
  Constructor from a string (type IDENTIFIER).

Static Private Member Functions

- **JBLParserResult expression** (std::istream &is)
- **JBLParserResult assignment_expression** (std::istream &is)
- **JBLParserResult conditional_expression** (std::istream &is)
- **JBLParserResult logical_OR_expression** (std::istream &is)
- **JBLParserResult logical_AND_expression** (std::istream &is)
- **JBLParserResult inclusive_OR_expression** (std::istream &is)
- **JBLParserResult exclusive_OR_expression** (std::istream &is)
- **JBLParserResult AND_expression** (std::istream &is)
- **JBLParserResult equality_expression** (std::istream &is)
- **JBLParserResult relational_expression** (std::istream &is)
- **JBLParserResult shift_expression** (std::istream &is)
- **JBLParserResult additive_expression** (std::istream &is)
- **JBLParserResult multiplicative_expression** (std::istream &is)
- **JBLParserResult cast_expression** (std::istream &is)
- **JBLParserResult unary_expression** (std::istream &is)
- **JBLParserResult postfix_expression** (std::istream &is)
- **JBLParserResult primary_expression** (std::istream &is)
- **JBLParserResult argument_expression_list** (std::istream &is)
- **JBLParserResult syntaxError** (const std::string &text)
- template<class T1, class T2> **JBLParserResult comparison** (int op, const T1 &lhs, const T2 &rhs)
- **JBLParserResult makeoperation** (int op, const std::string &opname, **JBLParserResult** &lhs, **JBLParserResult** &rhs)
- **JBLParserResult makefloatfunction** (const std::string &fname, **JBLParserResult** &arglist)

Private Attributes

- Type **type**
- int **intvalue**
- Float_FF **floatvalue**
- bool **boolvalue**
- **IntFun** ∗ **intfunvalue**
- **FloatFun** ∗ **floatfunvalue**
- **BaseCut** ∗ **basecutvalue**
- std::string **name**
- **JBLParserResult** ∗ **next**

Static Private Attributes

- **JBLToken token**

Related Functions

(Note that these are not member functions.)

- std::istream & **operator**>> (std::istream &is, **JBLParserResult** &result)
  Input a token.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 JBLParserResult::JBLParserResult (int *intvalue_*) [private]

Constructor from an int (type INT).

Parameters:
intvalue_ The integer value

Definition at line 59 of file JBLParserResult.C.

6.53.2.2 JBLParserResult::JBLParserResult (Float_FF *floatvalue_*) [private]

Constructor from a Float_FF (type FLOAT).

Parameters:
floatvalue_ The Float_FF value

Definition at line 65 of file JBLParserResult.C.

6.53.2.3 JBLParserResult::JBLParserResult (bool *boolvalue_*) [private]

Constructor from a bool (type BOOL).

Parameters:
boolvalue_ The bool value

Definition at line 71 of file JBLParserResult.C.

6.53.2.4 JBLParserResult::JBLParserResult (IntFun & *intfunvalue_*) [private]

Constructor from an **IntFun** (type INTFUN).

Parameters:

intfunvalue_ The **IntFun**

Definition at line 77 of file JBLParserResult.C.

6.53.2.5 JBLParserResult::JBLParserResult (FloatFun & *floatfunvalue_*) [private]

Constructor from a **FloatFun** (type FLOATFUN).

Parameters:

floatfunvalue_ The **FloatFun**

Definition at line 83 of file JBLParserResult.C.

6.53.2.6 JBLParserResult::JBLParserResult (BaseCut & *basecutvalue_*) [private]

Constructor from a **BaseCut** (type BASECUT).

Parameters:

basecutvalue_ The **BaseCut**

Definition at line 89 of file JBLParserResult.C.

6.53.2.7 JBLParserResult::JBLParserResult (const std::string & *name_*) [private]

Constructor from a string (type IDENTIFIER).

Parameters:

name_ The identifier name

Definition at line 95 of file JBLParserResult.C.

6.53.3 Member Function Documentation

6.53.3.1 std::ostream & JBLParserResult::print (std::ostream & *os*) const

Print the object's definition.

Parameters:

os The output stream

Definition at line 851 of file JBLParserResult.C.

References getBaseCutValue(), getBoolValue(), getFloatFunValue(), getFloatValue(), getIntFunValue(), getIntValue(), getName(), NamedO::getName(), isBaseCut(), isBool(), isEmpty(), isFloat(), isFloatFun(), isIdentifier(), isInt(), and isIntFun().

Referenced by operator<<().

6.53.3.2 JBLParserResult & JBLParserResult::addCopyToList (const JBLParserResult & *rhs*)

Add a copy of rhs to the end of the argument list.

Parameters:
rhs The additional element

Definition at line 103 of file JBLParserResult.C.

References JBLParserResult(), name, and next.

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

- **JBLParserResult.h**
- **JBLParserResult.C**

6.54 JBLToken Class Reference

`#include <JBLToken.h>`

6.54.1 Detailed Description

Tokeniser for string parsing.

This class is a helper class for class **JBLParserResult**

Author: Benno List, Jenny List

Date
2005/11/22 09:00:18

Author
blist

Changelog:

- 31.5.05 BL: First version
- 6.6.05 BL: Added methods parseExpression (std::istream& is), print, and operators<< and >>
- 6.6.05 BL: Added function issimpleoperator for ','

Definition at line 38 of file JBLToken.h.

Public Member Functions

- bool **input** (std::istream &is)
- bool **isEmpty** () const
- bool **isUnknown** () const
- bool **isIdentifier** () const
- bool **isNumber** () const
- bool **isInt** () const
- bool **isFloat** () const
- bool **isBool** () const
- bool **isOperator** () const
- bool **isParanthesis** () const
- int **getIntValue** () const
- float **getFloatValue** () const
- bool **getBoolValue** () const
- const std::string & **getStringValue** () const
- std::ostream & **print** (std::ostream &os) const

Private Types

- enum **Type** {
  EMPTY, **UNKNOWN**, **IDENTIFIER**, **INT**,
  FLOAT, **BOOL**, **OPERATOR**, **PARANTHESIS** }

Static Private Member Functions

- bool **isoperator** (char c)
- bool **issimpleoperator** (char c)
- bool **isparanthesis** (char c)

Private Attributes

- std::string **stringvalue**
- int **intvalue**
- float **floatvalue**
- bool **boolvalue**
- Type **type**

Related Functions

(Note that these are not member functions.)

- std::istream & **operator**>> (std::istream &is, **JBLToken** &token)

  Input a token.

- std::ostream & **operator**<< (std::ostream &os, const **JBLToken** &token)

  Output a token.

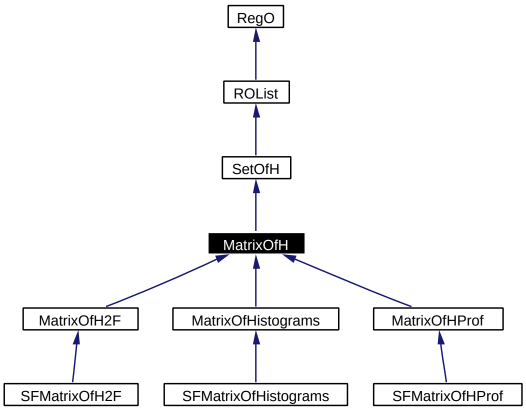
The documentation for this class was generated from the following files:

- **JBLToken.h**
- **JBLToken.C**

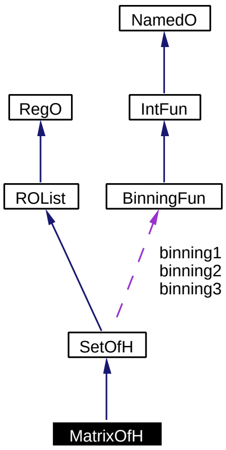
6.55 MatrixOfH Class Reference

`#include <MatrixOfH.h>`

Inheritance diagram for MatrixOfH:



Collaboration diagram for MatrixOfH:



6.55.1 Detailed Description

A 2-dimensional collection of 1-dimensional histograms that have the same axis.

A MatrixOfH contains a set of TH1 histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A MatrixOfH is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date
2005/11/26 13:51:01

Author
blist

Changelog:

- 27.5.05 BL: First version (from **MatrixOfHistograms**), derived from **SetOfH**
- 27.5.05 BL: Implemented dummy version of initHisto(i)
- 26.11.05 BL: Make initHisto public

Definition at line 51 of file MatrixOfH.h.

Public Member Functions

- **MatrixOfH** (const char *name_, const char *title_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, Option_t *option_=0)

  Constructor for an empty set of histograms.

- virtual ~**MatrixOfH** ()

  Virtual destructor.

- TH1 * **getHisto** (**IndexType** i, **IndexType** j)
- const char * **genBinTitle** (**IndexType** i, **IndexType** j) const

  Generate a bin title. Delete the returned string!

- const char * **genBinName** (**IndexType** i, **IndexType** j) const

  Generate a bin name. Delete the returned string!

- virtual void **initHisto** (**IndexType** i)

  Books and registers one histogram with number i.

- virtual void **initHisto** (**IndexType** i, **IndexType** j)=0

  Books and registers one histogram with number i.

Protected Member Functions

- virtual void **initHistos** ()

  Book all the histos , generate their names and titles.

- **IndexType getBinNumber** (**IndexType** i, **IndexType** j) const

  Get bin number for given bins in dimensions 1 and 2.

Private Member Functions

- **MatrixOfH** (const **MatrixOfH** &source)
  Private copy constructor, left unimplemented: No copying!
- **MatrixOfH** & **operator=** (const **MatrixOfH** &source)
  Private assignment, left unimplemented: No assignment!

Private Attributes

- **IndexType nbins2**
  The maximal number of bins of the 2nd Dimension.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 MatrixOfH::MatrixOfH (const char ∗ *name_*, const char ∗ *title_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, Option_t ∗ *option_* = 0)

Constructor for an empty set of histograms.

Parameters:
name_ Common histogram name
title_ Common histogram title
rol **ROList** object where "this" will be registered
binning1_ The 1st **BinningFun** that assigns entries to histograms
binning2_ The 2nd **BinningFun** that assigns entries to histograms
option_ An optional option.

Definition at line 31 of file MatrixOfH.C.

References SetOfH::getBinning(), BinningFun::getMaxBins(), and nbins2.

6.55.2.2 MatrixOfH::MatrixOfH (const MatrixOfH & *source*) `[private]`

Private copy constructor, left unimplemented: No copying!

Parameters:
source The source Matrix

6.55.3 Member Function Documentation

6.55.3.1 TH1 ∗ MatrixOfH::getHisto (IndexType *i*, IndexType *j*)

Returns a pointer to the histogram corresponding to bin i of the 1st and bin j of the 2nd binning

Parameters:
i Bin number of 1st binning

j Bin number of 2nd binning

Reimplemented in **MatrixOfH2F** (p. 193), **MatrixOfHistograms** (p. 201), and **MatrixOfHProf** (p. 210).

Definition at line 47 of file MatrixOfH.C.

References getBinNumber(), and ROList::getEntry().

6.55.3.2 const char * MatrixOfH::genBinTitle (IndexType *i*, IndexType *j*) const

Generate a bin title. Delete the returned string!

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Definition at line 73 of file MatrixOfH.C.

References SetOfH::getBinning(), BinningFun::getBinTitle(), and SetOfH::getTitle().

Referenced by MatrixOfHProf::initHisto(), MatrixOfHistograms::initHisto(), and MatrixOfH2F::initHisto().

6.55.3.3 const char * MatrixOfH::genBinName (IndexType *i*, IndexType *j*) const

Generate a bin name. Delete the returned string!

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Definition at line 102 of file MatrixOfH.C.

References BinningFun::getBinName(), SetOfH::getBinning(), and SetOfH::getName().

Referenced by MatrixOfHProf::initHisto(), MatrixOfHistograms::initHisto(), MatrixOfH2F::initHisto(), and MatrixOfHistograms::MatrixOfHistograms().

6.55.3.4 void MatrixOfH::initHisto (IndexType *i*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i The number under which to register

Implements **SetOfH** (p. 321).

Definition at line 43 of file MatrixOfH.C.

Referenced by initHistos().

6.55.3.5 virtual void MatrixOfH::initHisto (IndexType *i*, IndexType *j*) `[pure virtual]`

Books and registers one histogram with number i.

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Implemented in **MatrixOfH2F** (p. 194), **MatrixOfHistograms** (p. 201), and **MatrixOfHProf** (p. 210).

6.55.3.6 MatrixOfH::IndexType MatrixOfH::getBinNumber (IndexType *i*, IndexType *j*) const `[inline, protected]`

Get bin number for given bins in dimensions 1 and 2.

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Definition at line 110 of file MatrixOfH.h.

References ROList::IndexType, and nbins2.

Referenced by SFMatrixOfHProf::Fill(), SFMatrixOfHistograms::Fill(), SFMatrixOfH2F::Fill(), MatrixOfHProf::getHisto(), MatrixOfHistograms::getHisto(), MatrixOfH2F::getHisto(), getHisto(), MatrixOfHProf::initHisto(), MatrixOfHistograms::initHisto(), and MatrixOfH2F::initHisto().

6.55.3.7 MatrixOfH& MatrixOfH::operator= (const MatrixOfH & *source*) `[private]`

Private assignment, left unimplemented: No assignment!

Parameters:

source The source Matrix

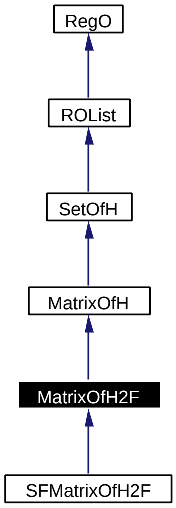
The documentation for this class was generated from the following files:

- **MatrixOfH.h**
- **MatrixOfH.C**

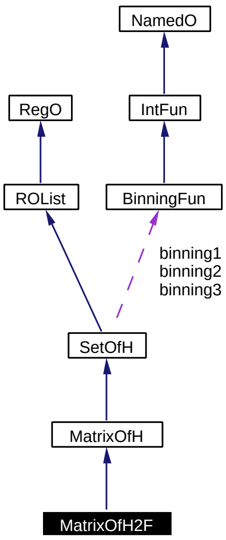
6.56 MatrixOfH2F Class Reference

`#include <MatrixOfH2F.h>`

Inheritance diagram for MatrixOfH2F:



Collaboration diagram for MatrixOfH2F:



6.56.1 Detailed Description

A 2-dimensional collection of 2-dimensional histograms that have the same axis.

A MatrixOfH2F contains a set of 1-dimensional TH2F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A MatrixOfH2F is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date

2005/11/26 13:51:01

Author

blist

Changelog:

- 14.6.05 BL: First version
- 26.11.05 BL: Make initHisto public

Definition at line 54 of file MatrixOfH2F.h.

Public Member Functions

- **MatrixOfH2F** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for an empty set histograms.

- **MatrixOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with equidistant x and y binnings.

- **MatrixOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

- **MatrixOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

- **MatrixOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

- **MatrixOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

- **MatrixOfH2F** (const char *name_, const char *title_, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms, taking 2 Binning objects.

- virtual ~**MatrixOfH2F** ()

  Virtual destructor.

- **RegH2F** * **getHisto** (**IndexType** i, **IndexType** j)
- virtual void **initHisto** (**IndexType** i, **IndexType** j)

  Books and registers one histogram with number i.

Private Member Functions

- **MatrixOfH2F** (const **MatrixOfH2F** &source)

  Private copy constructor, left unimplemented: No copying!

- **MatrixOfH2F** & **operator=** (const **MatrixOfH2F** &source)

  Private assignment, left unimplemented: No assignment!

Private Attributes

- Binning **xaxisbinning**

  The x axis binning.

- Binning **yaxisbinning**

  The y axis binning.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for an empty set histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 28 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.2 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with equidistant x and y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 38 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.3 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 56 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.4 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 73 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.5 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 90 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.6 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Int_t *nbinsy*, const Float_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH1F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in x (as for TH1F)

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 106 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.2.7 MatrixOfH2F::MatrixOfH2F (const char * *name_*, const char * *title_*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms, taking 2 Binning objects.

Parameters:

name_ TH2F name

title_ TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 122 of file MatrixOfH2F.C.

References MatrixOfH::initHistos().

6.56.3 Member Function Documentation

6.56.3.1 RegH2F * MatrixOfH2F::getHisto (IndexType *i*, IndexType *j*)

Returns a pointer to the histogram corresponding to bin i of the 1st and bin j of the 2nd binning

Parameters:

i Bin number of 1st binning

j Bin number of 2nd binning

Reimplemented from **MatrixOfH** (p. 185).

Definition at line 148 of file MatrixOfH2F.C.

References MatrixOfH::getBinNumber(), and ROList::getEntry().

6.56.3.2 void MatrixOfH2F::initHisto (IndexType *i*, IndexType *j*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Implements **MatrixOfH** (p. 187).

Definition at line 138 of file MatrixOfH2F.C.

References MatrixOfH::genBinName(), MatrixOfH::genBinTitle(), MatrixOfH::getBinNumber(), ROList::registerObject(), xaxisbinning, and yaxisbinning.

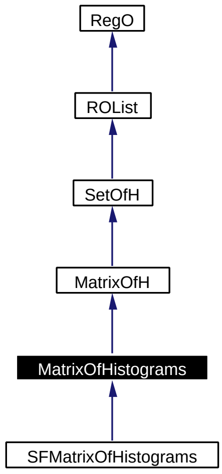
The documentation for this class was generated from the following files:

- **MatrixOfH2F.h**
- **MatrixOfH2F.C**

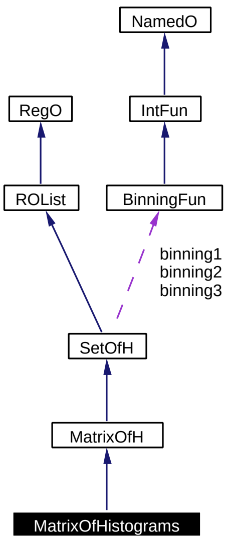
6.57 MatrixOfHistograms Class Reference

`#include <MatrixOfHistograms.h>`

Inheritance diagram for MatrixOfHistograms:



Collaboration diagram for MatrixOfHistograms:



6.57.1 Detailed Description

A 2-dimensional collection of 1-dimensional histograms that have the same axis.

A MatrixOfHistograms contains a set of 1-dimensional TH1F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A MatrixOfHistograms is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date

2005/11/26 13:51:01

Author

blist

Changelog:

- 22.6.04 BL: Upgrade documentation, use HListPoR and **BinningFunPoR**, introduce constructors using Binning objects, new method initHistos
- 30.6.04 Moved getHistInfo into **RegH1F**, added documentation
- 14.8.04 BL, JB: Add axisbinning, initHisto
- 27.5.04 BL: Derive from **MatrixOfH**, and therefore **SetOfH**;
- 27.5.04 BL: Derive from **MatrixOfH**, and therefore **SetOfH**; all Add, Divide etc methods are now residing solely in **SetOfH**
- 26.11.05 BL: Make initHisto public

Definition at line 63 of file MatrixOfHistograms.h.

Public Member Functions

- **MatrixOfHistograms** (const char *name_, const char *title_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for an empty set of histograms.

- **MatrixOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with equidistant axis binning.

- **MatrixOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **MatrixOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **MatrixOfHistograms** (const char ∗name_, const char ∗title_, const Binning &axisbinning_-, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms taking a Binning object.

- **MatrixOfHistograms** (const char ∗name_, const char ∗title_, TFile &file, const **ROList-PoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms to be read from a file.

- virtual ∼**MatrixOfHistograms** ()

  Virtual destructor.

- **RegH1F** ∗ **getSummary** (Option_t ∗**option**, const **ROListPoR** &rol)

  Returns a summary histogram.

- **SetOfHistograms** ∗ **getSummarySet** (const char ∗**name**, const char ∗**title**, int idim, Option_t ∗**option**, const **ROListPoR** &rol, const Float_t ∗xbinmargins=0)

  Returns a set of summary histograms.

- virtual **SetOfHistograms** ∗ **getSet** (const char ∗**name**, const char ∗**title**, int idim, const **ROListPoR** &rol, int lowbin=0, int highbin=0)

  Returns a histo set, where histos are summed over one dimension of the matrix.

- **RegH1F** ∗ **getHisto** (**IndexType** i, **IndexType** j)
- virtual void **initHisto** (**IndexType** i, **IndexType** j)

  Books and registers one histogram with number i.

Private Member Functions

- **MatrixOfHistograms** (const **MatrixOfHistograms** &source)

  Private copy constructor, left unimplemented: No copying!

- **MatrixOfHistograms** & **operator=** (const **MatrixOfHistograms** &source)

  Private assignment, left unimplemented: No assignment!

Private Attributes

- Binning **axisbinning**

  The axis binning.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 MatrixOfHistograms::MatrixOfHistograms (const char ∗ *name_*, const char ∗ *title_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for an empty set of histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 31 of file MatrixOfHistograms.C.

6.57.2.2 MatrixOfHistograms::MatrixOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 40 of file MatrixOfHistograms.C.

References MatrixOfH::initHistos().

6.57.2.3 MatrixOfHistograms::MatrixOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 54 of file MatrixOfHistograms.C.

References MatrixOfH::initHistos().

6.57.2.4 MatrixOfHistograms::MatrixOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 67 of file MatrixOfHistograms.C.

References MatrixOfH::initHistos().

6.57.2.5 MatrixOfHistograms::MatrixOfHistograms (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 80 of file MatrixOfHistograms.C.

References MatrixOfH::initHistos().

6.57.2.6 MatrixOfHistograms::MatrixOfHistograms (const char * *name_*, const char * *title_*, TFile & *file*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms to be read from a file.

Parameters:

name_ Common histogram name

title_ Common histogram title

file An input root file

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 92 of file MatrixOfHistograms.C.

References MatrixOfH::genBinName(), SetOfH::getBinning(), ROList::getEntries(), SetOfH::getName(), SetOfH::getTitle(), ROList::IndexType, ROListPoR::prol, and ROList::registerObject().

6.57.3 Member Function Documentation

6.57.3.1 RegH1F ∗ MatrixOfHistograms::getSummary (Option_t ∗ *option*, const ROListPoR & *rol*)

Returns a summary histogram.

The histogram contains on the x-axis a bin number and has as contents the contents or the mean of the individula histograms

Parameters:

option Option string Possible values:

- 1: 1-dim summary histo (default)
- 2: 2-dim histo
- C: Contents (default) => number 9999
- M: Mean => number 9998

rol List where to register the summary histo

Definition at line 142 of file MatrixOfHistograms.C.

References SetOfH::getBinning(), RegH1F::getHistInfo(), getHisto(), SetOfH::getName(), SetOfH::getTitle(), and ROList::IndexType.

6.57.3.2 SetOfHistograms ∗ MatrixOfHistograms::getSummarySet (const char ∗ *name*, const char ∗ *title*, int *idim*, Option_t ∗ *option*, const ROListPoR & *rol*, const Float_t ∗ *xbinmargins* = 0)

Returns a set of summary histograms.

Each histogram contains on the x-axis a bin number and has as contents the contents or the mean of the individula histograms. One can summarize over the first or second dimension of the matrix.

Parameters:

name Name of the summary histos

title Title of the summary histos

idim Dimension to use for the set (1 or 2)

option Option string Possible values:

- 1: 1-dim summary histo (default)
- 2: 2-dim histo
- C: Contents (default) => number 9999
- M: Mean => number 9998

rol List where to register the summary set

xbinmargins Set of x binmargins for summary histos

Definition at line 258 of file MatrixOfHistograms.C.

References SetOfH::getBinning(), RegH1F::getHistInfo(), getHisto(), SetOfHistograms::getHisto(), and ROList::IndexType.

6.57.3.3 SetOfHistograms ∗ MatrixOfHistograms::getSet (const char ∗ *name*, const char ∗ *title*, int *idim*, const ROListPoR & *rol*, int *lowbin* = 0, int *highbin* = 0) `[virtual]`

Returns a histo set, where histos are summed over one dimension of the matrix.

Parameters:

name Name for the summary histos

title Title for the summary histos

idim Dimension over which to sum (1 or 2)

rol List where to register the summary set

lowbin Optional number of lowest bin to sum

highbin Optional number of highest bin to sum

Definition at line 207 of file MatrixOfHistograms.C.

References SetOfH::genBinName(), SetOfH::genBinTitle(), SetOfH::getBinning(), getHisto(), and ROList::IndexType.

6.57.3.4 RegH1F ∗ MatrixOfHistograms::getHisto (IndexType *i*, IndexType *j*)

Returns a pointer to the histogram corresponding to bin i of the 1st and bin j of the 2nd binning

Parameters:

i Bin number of 1st binning

j Bin number of 2nd binning

Reimplemented from **MatrixOfH** (p. 185).

Definition at line 137 of file MatrixOfHistograms.C.

References MatrixOfH::getBinNumber(), and ROList::getEntry().

Referenced by getSet(), getSummary(), and getSummarySet().

6.57.3.5 void MatrixOfHistograms::initHisto (IndexType *i*, IndexType *j*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Implements **MatrixOfH** (p. 187).

Definition at line 123 of file MatrixOfHistograms.C.

References axisbinning, MatrixOfH::genBinName(), MatrixOfH::genBinTitle(), MatrixOfH::getBinNumber(), and ROList::registerObject().

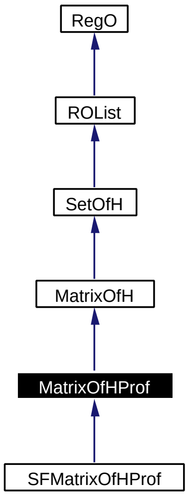
The documentation for this class was generated from the following files:

- **MatrixOfHistograms.h**
- **MatrixOfHistograms.C**

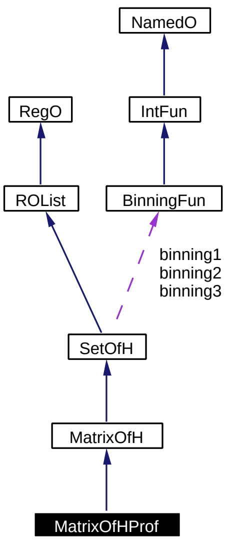
6.58 MatrixOfHProf Class Reference

`#include <MatrixOfHProf.h>`

Inheritance diagram for MatrixOfHProf:



Collaboration diagram for MatrixOfHProf:



6.58.1 Detailed Description

A 2-dimensional collection of 1-dimensional histograms that have the same axis.

A MatrixOfHProf contains a set of 1-dimensional THProfile histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A MatrixOfHProf is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date

2005/11/26 13:51:01

Author

blist

Changelog:

- 6.6.05 BL, JL: First version (derived from **MatrixOfHistograms**)
- 7.6.05 BL: Made compilable
- 26.11.05 BL: Make initHisto public

Definition at line 53 of file MatrixOfHProf.h.

Public Member Functions

- **MatrixOfHProf** (const char *name_, const char *title_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for an empty set of histograms.

- **MatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t *option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with equidistant axis binning.

- **MatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with equidistant axis binning.

- **MatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Option_t *option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **MatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **MatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Option_t *option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **MatrixOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Float_t ∗xbins, Axis_t ylow_, Axis_t yup_, Option_t ∗option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **MatrixOfHProf** (const char ∗name_, const char ∗title_, const Binning &axisbinning_, Option_t ∗option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms taking a Binning object.

- **MatrixOfHProf** (const char ∗name_, const char ∗title_, const Binning &axisbinning_, Axis_t ylow_, Axis_t yup_, Option_t ∗option_, const **ROListPoR** &rol, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_)

  Constructor for a set of histograms taking a Binning object.

- virtual ∼**MatrixOfHProf** ()

  Virtual destructor.

- **RegHProf** ∗ **getHisto** (**IndexType** i, **IndexType** j)
- virtual void **initHisto** (**IndexType** i, **IndexType** j)

  Books and registers one histogram with number i.

Private Member Functions

- **MatrixOfHProf** (const **MatrixOfHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **MatrixOfHProf** & **operator=** (const **MatrixOfHProf** &source)

  Private assignment, left unimplemented: No assignment!

Private Attributes

- Binning **axisbinning**

  The axis binning.

- Axis_t **ylow**

  Minimum y value (as for TProfile).

- Axis_t **yup**

  Maximum y value (as for TProfile).

6.58.2 Constructor & Destructor Documentation

6.58.2.1 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for an empty set of histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 31 of file MatrixOfHProf.C.

6.58.2.2 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xlow Lower edge (as for TProfile)

xup Upper edge (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 40 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.3 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xlow Lower edge (as for TProfile)

xup Upper edge (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 57 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.4 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 76 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.5 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 92 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.6 MatrixOfHProf::MatrixOfHProf (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Float_t ∗ *xbins*, Option_t ∗ *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 110 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.7 MatrixOfHProf::MatrixOfHProf (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Float_t ∗ *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t ∗ *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 126 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.8 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 144 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.2.9 MatrixOfHProf::MatrixOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*)

Constructor for a set of histograms taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

Definition at line 159 of file MatrixOfHProf.C.

References MatrixOfH::initHistos().

6.58.3 Member Function Documentation

6.58.3.1 RegHProf * MatrixOfHProf::getHisto (IndexType *i*, IndexType *j*)

Returns a pointer to the histogram corresponding to bin i of the 1st and bin j of the 2nd binning

Parameters:

i Bin number of 1st binning

j Bin number of 2nd binning

Reimplemented from **MatrixOfH** (p. 185).

Definition at line 222 of file MatrixOfHProf.C.

References MatrixOfH::getBinNumber(), and ROList::getEntry().

6.58.3.2 void MatrixOfHProf::initHisto (IndexType *i*, IndexType *j*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i Bin of 1st dimension

j Bin of 2nd dimension

Implements **MatrixOfH** (p. 187).

Definition at line 208 of file MatrixOfHProf.C.

References axisbinning, MatrixOfH::genBinName(), MatrixOfH::genBinTitle(), MatrixOfH::getBinNumber(), SetOfH::getOption(), ROList::registerObject(), ylow, and yup.

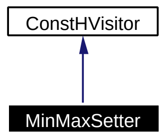
The documentation for this class was generated from the following files:

- **MatrixOfHProf.h**
- **MatrixOfHProf.C**

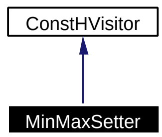
6.59 MinMaxSetter Class Reference

#include <HVisitors.h>

Inheritance diagram for MinMaxSetter:



Collaboration diagram for MinMaxSetter:



6.59.1 Detailed Description

An **HVisitor** that sets the minumum and maximum of all TH1 histograms.

Definition at line 206 of file HVisitors.h.

Public Member Functions

- **MinMaxSetter** (Double_t min_, Double_t max_)
- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- Double_t **min**

  The new minimum.

- Double_t **max**

  The new maximum.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 MinMaxSetter::MinMaxSetter (Double_t *min_*, Double_t *max_*) [inline]

Parameters:

*min_* The new minimum

max_ The new maximum

Definition at line 208 of file HVisitors.h.

References max, and min.

6.59.3 Member Function Documentation

6.59.3.1 virtual void MinMaxSetter::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 214 of file HVisitors.h.

References max, and min.

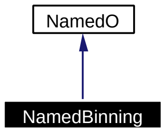
The documentation for this class was generated from the following file:

- **HVisitors.h**

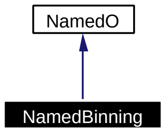
6.60 NamedBinning Class Reference

`#include <NamedBinning.h>`

Inheritance diagram for NamedBinning:



Collaboration diagram for NamedBinning:



6.60.1 Detailed Description

Abstract base class for function objects that sort events into bins.

A NamedBinning is a Binning with a name, which is entered into the list of named objects

Author: Jenny Böhme, Benno List

Date

2005/08/23 12:05:44

Author

blist

Changelog:

- 7.6.05 BL: First version (from **BinningFun**)
- 23.8.05 BL: Cosmetic changes

Definition at line 26 of file NamedBinning.h.

Public Member Functions

- **NamedBinning** ()

  Default: Create an empty binning.

- **NamedBinning** (int nbins, float xlow, float xup, const char ∗**name**=0)

  Create an equidistant binning.

- **NamedBinning** (int nbins, float xlow, float xup, const std::string &**name**)

  Create an equidistant binning.

- **NamedBinning** (int nbins, const float binedges_[ ], const char ∗**name**=0)

  Create an nonequidistant binning, taking floats.

- **NamedBinning** (int nbins, const float binedges_[ ], const std::string &**name**)

  Create an nonequidistant binning, taking floats.

- **NamedBinning** (int nbins, const double binedges_[ ], const char ∗**name**=0)

  Create an nonequidistant binning, taking doubles.

- **NamedBinning** (int nbins, const double binedges_[ ], const std::string &**name**)

  Create an nonequidistant binning, taking doubles.

- **NamedBinning** (const Binning &binning_, const char ∗**name**=0)

  Create a binning from a Binning object.

- **NamedBinning** (const Binning &binning_, const std::string &**name**)

  Create a binning from a Binning object.

- virtual int **getMaxBins** () const

  Gets the maximal return value of operator() plus 1, or the higest "sensible" value for a bin number.

Static Public Member Functions

- **NamedBinning** ∗ **getObject** (const std::string &**name**)

  Access function that returns a pointer to an **FloatFun** *object with this name.*

- **NamedBinning** ∗ **getObject** (const char ∗**name**)

  Access function that returns a pointer to an **NamedO** *object with this name.*

Protected Member Functions

- virtual ∼**NamedBinning** ()

  Protected destructor to ensure creation on the heap.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 NamedBinning::NamedBinning (int *nbins*, float *xlow*, float *xup*, const char ∗ *name* = 0) `[inline]`

Create an equidistant binning.

The binning will have nbins bins between xlow and xup.

Parameters:

nbins The number of bins

xlow The lowest bin edge

xup The highest bin edge

name The object name

Definition at line 34 of file NamedBinning.h.

References NamedO::str().

6.60.2.2 NamedBinning::NamedBinning (int *nbins*, float *xlow*, float *xup*, const std::string & *name*) `[inline]`

Create an equidistant binning.

The binning will have nbins bins between xlow and xup.

Parameters:

nbins The number of bins

xlow The lowest bin edge

xup The highest bin edge

name The object name

Definition at line 46 of file NamedBinning.h.

6.60.2.3 NamedBinning::NamedBinning (int *nbins*, const float *binedges_*[ ], const char * *name* = 0) `[inline]`

Create an nonequidistant binning, taking floats.

The binning will have nbins bins with bin edges given by binedges_. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges_ An array with bin edges

name The object name

Definition at line 63 of file NamedBinning.h.

References NamedO::str().

6.60.2.4 NamedBinning::NamedBinning (int *nbins*, const float *binedges_*[ ], const std::string & *name*) `[inline]`

Create an nonequidistant binning, taking floats.

The binning will have nbins bins with bin edges given by binedges_. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges_ An array with bin edges

name The object name

Definition at line 80 of file NamedBinning.h.

6.60.2.5 NamedBinning::NamedBinning (int *nbins*, const double *binedges_*[ ], const char * *name* = 0) `[inline]`

Create an nonequidistant binning, taking doubles.

The binning will have nbins bins with binedges given by binedges_. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges_ An array with bin edges

name The object name

Definition at line 97 of file NamedBinning.h.

References NamedO::str().

6.60.2.6 NamedBinning::NamedBinning (int *nbins*, const double *binedges_*[ ], const std::string & *name*) `[inline]`

Create an nonequidistant binning, taking doubles.

The binning will have nbins bins with bin edges given by binedges. binedges_[i] and binedges_[i+1] are the lower and upper edges of bin i. It is assumed that the binedge values are monotonically increasing.

Parameters:

nbins The number of bins

binedges_ An array with bin edges

name The object name

Definition at line 115 of file NamedBinning.h.

6.60.2.7 NamedBinning::NamedBinning (const Binning & *binning_*, const char * *name* = 0) `[inline]`

Create a binning from a Binning object.

Parameters:

binning_ The binning

name The object name

Definition at line 124 of file NamedBinning.h.

References NamedO::str().

6.60.2.8 NamedBinning::NamedBinning (const Binning & *binning_*, const std::string & *name*) [inline]

Create a binning from a Binning object.

Parameters:

binning_ The binning

name The object name

Definition at line 136 of file NamedBinning.h.

6.60.3 Member Function Documentation

6.60.3.1 NamedBinning* NamedBinning::getObject (const std::string & *name*) [inline, static]

Access function that returns a pointer to an **FloatFun** object with this name.

This access function must never be called within the constructor of any class that is derived from NamedBinning!

Parameters:

name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 156 of file NamedBinning.h.

6.60.3.2 NamedBinning* NamedBinning::getObject (const char * *name*) [inline, static]

Access function that returns a pointer to an **NamedO** object with this name.

This access function must never be called within the constructor of any class that is derived from NamedBinning!

Parameters:

name The name

Reimplemented from **NamedO** (p. 222).

Definition at line 166 of file NamedBinning.h.

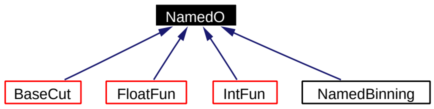
The documentation for this class was generated from the following file:

- **NamedBinning.h**

6.61 NamedO Class Reference

`#include <NamedO.h>`

Inheritance diagram for NamedO:



6.61.1 Detailed Description

Base class for heap objects that have a name.

Classes derived from NamedO should implement a **getName()** method. under the name returned by getName each new object is registered, and can be looked up by name using the **NamedO::getObject** function.

Changelog:

- 29.5.05 BL: First version
- 31.5.05 BL: Add data member name
- 6.6.05 BL: Add data member definition, added Init class
- 7.6.05 BL: Added **str(double)**, and printing method
- 29.6.05 BL: Declared class Init as friend to make some compilers happy

Definition at line 28 of file NamedO.h.

Public Member Functions

- **NamedO** (const char *name_="?")
  Constructor from a C style string, serves as default constructor.
- **NamedO** (const std::string &name_)
  Constructor from a C++ string.
- **NamedO** (const std::string &name_, const std::string &definition_)
  Constructor from two C++ strings.
- virtual const char * **getName** () const
  Returns pointer to name (pointer must NOT be deleted!).
- virtual const char * **getDefinition** () const
  Returns pointer to definition (pointer must NOT be deleted!).
- virtual **NamedO** & **setName** (const std::string &newname)
  Method to be called when the name is changed.

- virtual **NamedO** & **setName** (const char ∗newname)
- virtual **NamedO** & **setDefinition** (const std::string &newdef)

  Method to change the definition.

- virtual std::ostream & **print** (std::ostream &os) const

  Method for printing.

- virtual void **destroy** ()

  Replaces delete, which cannot be called because of protected destructor.

Static Public Member Functions

- **NamedO** ∗ **getObject** (const std::string &**name**)

  Access function that returns a pointer to an NamedO object with this name.

- **NamedO** ∗ **getObject** (const char ∗**name**)

  Access function that returns a pointer to an NamedO object with this name.

- template<class T> T ∗ **getObject** (const std::string &**name**)
- template<class T> T ∗ **getObject** (const char ∗**name**)
- std::ostream & **listObjects** (std::ostream &os)

  Outputs a list of all objects registered so far.

- std::string **str** (char c)

  Converts a char into a string.

- std::string **str** (const char ∗s)

  Converts a C style string into a string.

- std::string **str** (int i)

  Converts an int into a string.

- std::string **str** (int i, int wide)

  Converts an int into a string, with a minimum number of digits (leading zeroes).

- std::string **str** (float f)

  Converts an float into a string.

- std::string **str** (double d)

  Converts a double into a string.

- std::string **str** (bool b)

  Converts a bool into a string.

Protected Member Functions

- virtual ~**NamedO** ()

  Protected destructor ensures creation on the heap.

Private Member Functions

- void **reg** ()

  Register an object.

- void **unreg** ()

  Unregister an object.

- **NamedO** (const **NamedO** &source)

  Private copy constructor, left unimplemented: No copying!

- **NamedO** & **operator=** (const **NamedO** &source)

  Private assignment operator, left unimplemented: No assignment!

Static Private Member Functions

- void **initMap** ()

  Internal function to initialize the maps.

Private Attributes

- std::string **name**

  The object's name.

- std::string **definition**

  The object's name.

Friends

- class **NamedO::Init**

Related Functions

(Note that these are not member functions.)

- std::ostream & **operator<<** (std::ostream &os, const **NamedO** &rhs)

  Output operator.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 NamedO::NamedO (const char * *name_* = "?") [explicit]

Constructor from a C style string, serves as default constructor.

Parameters:
*name_* The object's name

Definition at line 65 of file NamedO.C.

References reg().

6.61.2.2 NamedO::NamedO (const std::string & *name_*) [explicit]

Constructor from a C++ string.

Parameters:
*name_* The object's name

Definition at line 72 of file NamedO.C.

References reg().

6.61.2.3 NamedO::NamedO (const std::string & *name_*, const std::string & *definition_*)

Constructor from two C++ strings.

Parameters:
*name_* The object's name
*definition_* The object's definition

Definition at line 79 of file NamedO.C.

References reg().

6.61.3 Member Function Documentation

6.61.3.1 NamedO & NamedO::setName (const std::string & *newname*) [virtual]

Method to be called when the name is changed.

Parameters:
newname The new name

Definition at line 147 of file NamedO.C.

References name, and reg().

Referenced by JBLH1SomeArrayIterator< H1SomeArrayPtr, ArrayRow >::JBLH1SomeArrayIterator(), JBLH1SomeBankIterator< H1SomeBankPtr, H1SomeRow >::JBLH1SomeBankIterator(), and setName().

6.61.3.2 NamedO & NamedO::setName (const char ∗ *newname*) [virtual]

Parameters:

newname The new name

Definition at line 156 of file NamedO.C.

References setName().

6.61.3.3 NamedO & NamedO::setDefinition (const std::string & *newdef*) [virtual]

Method to change the definition.

Parameters:

newdef The new definition

Definition at line 160 of file NamedO.C.

References definition.

Referenced by CutBinning::genDefinition().

6.61.3.4 NamedO ∗ NamedO::getObject (const std::string & *name*) [static]

Access function that returns a pointer to an NamedO object with this name.

This access function must never be called within the constructor of any class that is derived from NamedO!

Parameters:

name The name

Reimplemented in **BaseCut** (p. 37), **FloatFun** (p. 92), **IntFun** (p. 123), and **NamedBinning** (p. 217).

Definition at line 171 of file NamedO.C.

References getName().

Referenced by getObject().

6.61.3.5 NamedO ∗ NamedO::getObject (const char ∗ *name*) [static]

Access function that returns a pointer to an NamedO object with this name.

This access function must never be called within the constructor of any class that is derived from NamedO!

Parameters:

name The name

Reimplemented in **BaseCut** (p. 37), **FloatFun** (p. 92), **IntFun** (p. 124), and **NamedBinning** (p. 217).

Definition at line 183 of file NamedO.C.

References getObject().

6.61.3.6 template<class T> T* NamedO::getObject (const std::string & *name*) [inline, static]

Parameters:

name The name

Reimplemented in **BaseCut** (p. 37), **FloatFun** (p. 92), **IntFun** (p. 123), and **NamedBinning** (p. 217).

Definition at line 76 of file NamedO.h.

References getObject().

6.61.3.7 template<class T> T* NamedO::getObject (const char * *name*) [inline, static]

Parameters:

name The name

Reimplemented in **BaseCut** (p. 37), **FloatFun** (p. 92), **IntFun** (p. 124), and **NamedBinning** (p. 217).

Definition at line 82 of file NamedO.h.

References getObject().

6.61.3.8 std::string NamedO::str (char *c*) [static]

Converts a char into a string.

Parameters:

c The character

Definition at line 188 of file NamedO.C.

Referenced by BinningFun::BinningFun(), ConstBaseCut::ConstBaseCut(), ConstFun::ConstFun(), ConstIntFun::ConstIntFun(), CutBinning::CutBinning(), JBLH1BankFloatFunInt::JBLH1BankFloatFunInt(), JBLH1BankIntFun::JBLH1BankIntFun(), JBLH1FloatFun< H1SomePtr >::JBLH1FloatFun(), JBLH1IntFun< H1SomePtr >::JBLH1IntFun(), NamedBinning::NamedBinning(), FloatFun::operator-(), and RangeIterator::RangeIterator().

6.61.3.9 std::string NamedO::str (const char * *s*) [static]

Converts a C style string into a string.

Parameters:

s A C style string

Definition at line 194 of file NamedO.C.

6.61.3.10 std::string NamedO::str (int *i*) [static]

Converts an int into a string.

Parameters:

i The integer

Definition at line 198 of file NamedO.C.

6.61.3.11 std::string NamedO::str (int *i*, int *wide*) [static]

Converts an int into a string, with a minimum number of digits (leading zeroes).

Parameters:

i The integer,

wide The field width

Definition at line 204 of file NamedO.C.

6.61.3.12 std::string NamedO::str (float *f*) [static]

Converts an float into a string.

Parameters:

f The float

Definition at line 212 of file NamedO.C.

6.61.3.13 std::string NamedO::str (double *d*) [static]

Converts a double into a string.

Parameters:

d The double

Definition at line 218 of file NamedO.C.

6.61.3.14 std::string NamedO::str (bool *b*) [static]

Converts a bool into a string.

Parameters:

b The bool

Definition at line 224 of file NamedO.C.

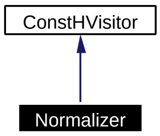
The documentation for this class was generated from the following files:

- **NamedO.h**
- **NamedO.C**

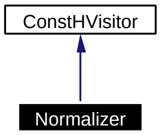
6.62 Normalizer Class Reference

`#include <HVisitors.h>`

Inheritance diagram for Normalizer:



Collaboration diagram for Normalizer:



6.62.1 Detailed Description

An **HVisitor** that normalizes all TH1 objects to a constant.

If overflow bins are to be included, this works currently only for 1D histograms

Definition at line 42 of file HVisitors.h.

Public Member Functions

- **Normalizer** (Double_t c_=1, bool includeOverflow_=true)

  Constructor.

- virtual void **visit** (**RegO** &h) const

  The visit function.

Private Attributes

- Double_t **c**

  The new norm.

- bool **includeOverflow**

  Include overflows?

6.62.2 Constructor & Destructor Documentation

6.62.2.1 Normalizer::Normalizer (Double_t *c_* = 1, bool *includeOverflow_* = true) `[inline]`

Constructor.

Parameters:

c_ The new norm

includeOverflow_ Include overflows?

Definition at line 45 of file HVisitors.h.

References c, and includeOverflow.

6.62.3 Member Function Documentation

6.62.3.1 virtual void Normalizer::visit (RegO & *h*) const `[inline, virtual]`

The visit function.

Parameters:

h The object to be visited

Implements **ConstHVisitor** (p. 60).

Definition at line 50 of file HVisitors.h.

References c, and includeOverflow.

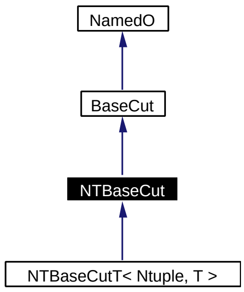
The documentation for this class was generated from the following file:

- **HVisitors.h**

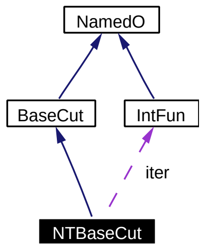
6.63 NTBaseCut Class Reference

`#include <NTBaseCut.h>`

Inheritance diagram for NTBaseCut:



Collaboration diagram for NTBaseCut:



6.63.1 Detailed Description

Returns a bool value from an Ntuple (abstract base class for **NTBaseCutT** templates).

Author: Benno List, Jenny List

Date
2005/06/04 16:27:51

Author
blist

Changelog:

- 4.6.05 BL: First version (taken from **NTFloatFun**)

Definition at line 28 of file NTBaseCut.h.

Public Member Functions

- **NTBaseCut** (const std::string &name_)
  Constructor from a name.

- **NTBaseCut** (const std::string &name_, const **IntFun** *iter_)

  Constructor from a name and an iterator.

- **NTBaseCut** (const std::string &name_, int offset_)

  Constructor from a name and an offsets.

- virtual **NTBaseCut** & **operator**[ ] (const **IntFunPoR** &iter_) const=0

  Returns a new object with a different iterator.

- virtual **NTBaseCut** & **operator**[ ] (int offset_) const=0

  Returns a new object with a different offset.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ∼**NTBaseCut** ()

  Protected destructor ensures creation on the heap.

- std::string **getBaseName** () const

  Returns the part of the name before the "[".

Protected Attributes

- const **IntFun** * **iter**

  An optional **FillIterator**.

- int **offset**

  An optional offset, i.e. array index.

Related Functions

(Note that these are not member functions.)

- template<class Ntuple, class T> **NTBaseCutT**< Ntuple, T > & **ntbasecut** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_="?", const **IntFunPoR** &iter_=0)

  returns a **BaseCut** *from an ntuple data member*

- template<class Ntuple, class T> **NTBaseCutT**< Ntuple, T > & **ntbasecut** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_="?", int offset_)

  returns a **BaseCut** *from an ntuple data member*

6.63.2 Constructor & Destructor Documentation

6.63.2.1 NTBaseCut::NTBaseCut (const std::string & *name_*)

Constructor from a name.

Parameters:
*name_* The name

Definition at line 20 of file NTBaseCut.C.

References NamedO::getName().

6.63.2.2 NTBaseCut::NTBaseCut (const std::string & *name_*, const IntFun * *iter_*)

Constructor from a name and an iterator.

Parameters:
*name_* The name
*iter_* The Iterator

Definition at line 30 of file NTBaseCut.C.

References NamedO::getName().

6.63.2.3 NTBaseCut::NTBaseCut (const std::string & *name_*, int *offset_*)

Constructor from a name and an offsets.

Parameters:
*name_* The name
*offset_* The integer offset

Definition at line 40 of file NTBaseCut.C.

References NamedO::getName().

6.63.3 Member Function Documentation

6.63.3.1 virtual NTBaseCut& NTBaseCut::operator[] (const IntFunPoR & *iter_*) const `[pure virtual]`

Returns a new object with a different iterator.

Parameters:
*iter_* the new iterator

Implemented in **NTBaseCutT< Ntuple, T >** (p. 234).

6.63.3.2 virtual NTBaseCut& NTBaseCut::operator[] (int *offset_*) const [pure virtual]

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implemented in **NTBaseCutT< Ntuple, T >** (p. 235).

6.63.4 Friends And Related Function Documentation

6.63.4.1 template<class Ntuple, class T> NTBaseCutT< Ntuple, T > & ntbasecut (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", const IntFunPoR & *iter_* = 0) [related]

returns a **BaseCut** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 197 of file NTBaseCut.h.

6.63.4.2 template<class Ntuple, class T> NTBaseCutT< Ntuple, T > & ntbasecut (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", int *offset_*) [related]

returns a **BaseCut** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset into an array

Definition at line 210 of file NTBaseCut.h.

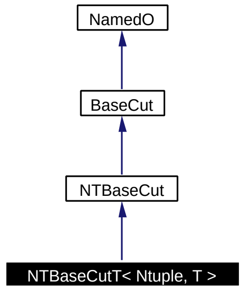
The documentation for this class was generated from the following files:

- **NTBaseCut.h**
- **NTBaseCut.C**

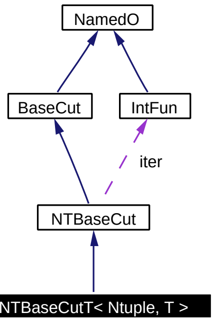
6.64 NTBaseCutT< Ntuple, T > Class Template Reference

`#include <NTBaseCut.h>`

Inheritance diagram for NTBaseCutT< Ntuple, T >:



Collaboration diagram for NTBaseCutT< Ntuple, T >:



6.64.1 Detailed Description

template<class Ntuple, class T = bool> class NTBaseCutT< Ntuple, T >

Returns a bool value from an Ntuple.

An NTBaseCutT collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a value from a given branch.

The first template argument is the class name of the ntuple class that was automatically generated by TTree::MakeClass. The second (optional) template argument is the type of the branch variable; this is useful if the branch contains not bool objects, but double or even int values.

Author: Benno List, Jenny Böhme

Date

2005/06/04 16:27:51

Author

blist

Changelog:

- 15.7.04 BL: Added method **getIterator()**
- 20.7.04 BL: Use **IntFun** instead of **FillIterator**
- 8.4.05 BL: Add optional offset, and operator[]
- 6.5.04 BL: Adapted to name in base class, added new constructors needed for operator[]

Definition at line 94 of file NTBaseCut.h.

Public Member Functions

- **NTBaseCutT** (const Ntuple &nt, const char ∗branchname, const **IntFunPoR** &iter_=0)
  Constructor from a branch name given as string.
- **NTBaseCutT** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_="?", const **IntFunPoR** &iter_=0)
  Constructor from a member pointer.
- **NTBaseCutT** (const Ntuple &nt, const char ∗branchname, int offset_)
  Constructor from a branch name given as string, with offset.
- **NTBaseCutT** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_, int offset_)
  Constructor from a member pointer, with offset.
- **NTBaseCutT** (const **NTBaseCutT**< Ntuple, T > &source, const **IntFunPoR** &iter_)
  Constructor from an existing NTBaseCutT object, plus a new iterator.
- **NTBaseCutT** (const **NTBaseCutT**< Ntuple, T > &source, const int offset_)
  Constructor from an existing NTBaseCutT object, plus a new offset.
- virtual bool **operator()** () const
  Returns a bool value from an ntuple row.
- **NTBaseCutT**< Ntuple, T > & **operator[ ]** (const **IntFunPoR** &iter_) const
  Returns a new object with a different iterator.
- **NTBaseCutT**< Ntuple, T > & **operator[ ]** (int offset_) const
  Returns a new object with a different offset.

Protected Member Functions

- virtual ∼**NTBaseCutT** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const T ∗ **branchAddress**
  The branch adress.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const Ntuple & *nt*, const char ∗ *branchname*, const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a branch name given as string.

Parameters:

nt The ntuple

branchname The branch name

iter_ An optional iterator

Definition at line 97 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress.

6.64.2.2 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const Ntuple & *nt*, T Ntuple::∗ *p_mem*, const char ∗ *name_* = "?", const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a member pointer.

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 112 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress.

6.64.2.3 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const Ntuple & *nt*, const char ∗ *branchname*, int *offset_*) `[inline]`

Constructor from a branch name given as string, with offset.

Parameters:

nt The ntuple

branchname The branch name

offset_ The offset

Definition at line 124 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress.

6.64.2.4 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_*, int *offset_*) [inline]

Constructor from a member pointer, with offset.

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ The offset

Definition at line 139 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress.

6.64.2.5 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const NTBaseCutT< Ntuple, T > & *source*, const IntFunPoR & *iter_*) [inline]

Constructor from an existing NTBaseCutT object, plus a new iterator.

Parameters:

iter_ The new iterator

Definition at line 151 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress, and NTBaseCut::getBaseName().

6.64.2.6 template<class Ntuple, class T = bool> NTBaseCutT< Ntuple, T >::NTBaseCutT (const NTBaseCutT< Ntuple, T > & *source*, const int *offset_*) [inline]

Constructor from an existing NTBaseCutT object, plus a new offset.

Parameters:

offset_ The new offset

Definition at line 158 of file NTBaseCut.h.

References NTBaseCutT< Ntuple, T >::branchAddress, and NTBaseCut::getBaseName().

6.64.3 Member Function Documentation

6.64.3.1 template<class Ntuple, class T = bool> NTBaseCutT<Ntuple, T>& NTBaseCutT< Ntuple, T >::operator[] (const IntFunPoR & *iter_*) const [inline, virtual]

Returns a new object with a different iterator.

Parameters:

iter_ the new iterator

Implements **NTBaseCut** (p. 229).

Definition at line 173 of file NTBaseCut.h.

6.64.3.2 template<class Ntuple, class T = bool> NTBaseCutT<Ntuple, T>& NTBaseCutT< Ntuple, T >::operator[] (int *offset_*) const [inline, virtual]

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implements **NTBaseCut** (p. 230).

Definition at line 179 of file NTBaseCut.h.

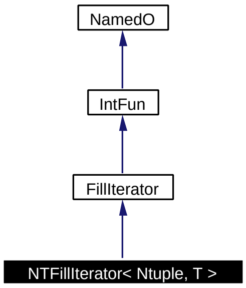
The documentation for this class was generated from the following file:

- **NTBaseCut.h**

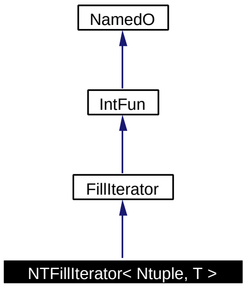
6.65 NTFillIterator< Ntuple, T > Class Template Reference

```
#include <NTFillIterator.h>
```

Inheritance diagram for NTFillIterator< Ntuple, T >:



Collaboration diagram for NTFillIterator< Ntuple, T >:



6.65.1 Detailed Description

template<class Ntuple, class T = Int_t> class NTFillIterator< Ntuple, T >

A **FillIterator** from an ntuple variable.

An NTFillIterator collaborates with a class Ntuple, which automatically generated by TTree::MakeClass. The branch name given must correspond to an Int_t that represent the number of entries (jets, tracks, whatever) over which one wants to iterate.

The first template argument is the class name of the ntuple class that was automatically generated by TTree::MakeClass. The second (optional) template argument is the type of the branch variable; this is useful if the branch contains not Int_t objects, but short or char values.

Author: Benno List, Jenny Böhme

Date

2005/06/04 16:27:51

Author

blist

Changelog:

- 6.5.04 BL: Adapted to name in base class

Definition at line 43 of file NTFillIterator.h.

Public Member Functions

- **NTFillIterator** (const Ntuple &nt_, const char *branchname)

  Constructor using a branch name.

- **NTFillIterator** (const Ntuple &nt_, T Ntuple::*p_mem, const char *objectname="?")

  Constructor using a member pointer.

- virtual int **operator()** () const

  Returns iterator value, starting at 0; "-1" means "invalid".

- virtual bool **next** ()

  Increments iterator; returns false if iterator cannot be incremented.

- virtual bool **reset** ()

  Resets iterator; returns false if iterator value range is empty.

- virtual bool **isValid** () const

  Returns whether current value of iterator is valid.

Protected Member Functions

- virtual ~**NTFillIterator** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const Ntuple & **nt**

  The ntuple.

- const T * **branchAddress**

  The branch adress.

- int **index**

  The current index value.

Related Functions

(Note that these are not member functions.)

- **NTFillIterator**< Ntuple, T > & **ntfilliterator** (const Ntuple &**nt**, T Ntuple::*p_mem, const char *objectname="?")

  returns a **FillIterator** *from an ntuple data member*

6.65.2 Constructor & Destructor Documentation

6.65.2.1 template<class Ntuple, class T = Int_t> NTFillIterator< Ntuple, T >::NTFillIterator (const Ntuple & *nt_*, const char * *branchname*) `[inline]`

Constructor using a branch name.

Parameters:

nt_ The ntuple

branchname The branch name

Definition at line 46 of file NTFillIterator.h.

References NTFillIterator< Ntuple, T >::branchAddress, NamedO::getName(), NTFillIterator< Ntuple, T >::index, and NTFillIterator< Ntuple, T >::nt.

6.65.2.2 template<class Ntuple, class T = Int_t> NTFillIterator< Ntuple, T >::NTFillIterator (const Ntuple & *nt_*, T Ntuple::* *p_mem*, const char * *objectname* = "?") `[inline]`

Constructor using a member pointer.

Parameters:

nt_ The ntuple

p_mem Member pointer

objectname The object's name

Definition at line 66 of file NTFillIterator.h.

References NTFillIterator< Ntuple, T >::branchAddress, NamedO::getName(), NTFillIterator< Ntuple, T >::index, and NTFillIterator< Ntuple, T >::nt.

6.65.3 Friends And Related Function Documentation

6.65.3.1 template<class Ntuple, class T = Int_t> NTFillIterator< Ntuple, T > & ntfilliterator (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *objectname* = "?") `[related]`

returns a **FillIterator** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

objectname The object's name

Definition at line 118 of file NTFillIterator.h.

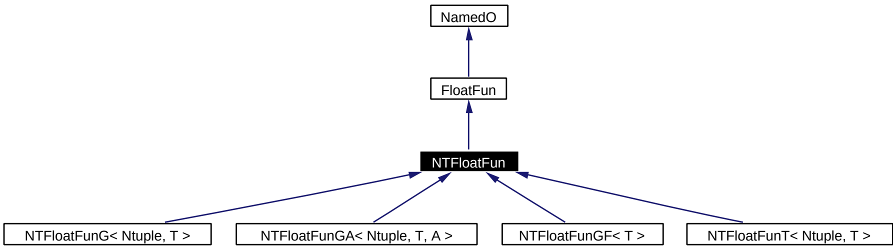
The documentation for this class was generated from the following file:

- **NTFillIterator.h**

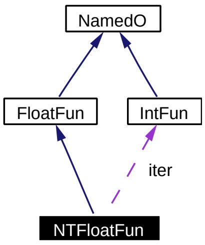
6.66 NTFloatFun Class Reference

`#include <NTFloatFun.h>`

Inheritance diagram for NTFloatFun:



Collaboration diagram for NTFloatFun:



6.66.1 Detailed Description

Returns a Float_FF value from an Ntuple (abstract base class for **NTFloatFunT** templates).

Author: Benno List, Jenny List

Date
2006/10/25 15:06:59

Author
msteder

Changelog:

- 8.4.05 BL: First version
- 6.5.04 BL: Adapted to name in base class, added new method **getBaseName()**, removed methods setIterator and setOffset, created new constructors
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**
- 5.9.06 BL, TZ: Add ntfloatfun call for const member function getters
- 25.10.06 BL: Add ntfloatfun call for getters with arguments (class **NTFloatFunGA**)

Definition at line 35 of file NTFloatFun.h.

Public Member Functions

- **NTFloatFun** (const std::string &name_)
  Constructor from a name.
- **NTFloatFun** (const std::string &name_, const **IntFun** ∗iter_)
  Constructor from a name and an iterator.
- **NTFloatFun** (const std::string &name_, int offset_)
  Constructor from a name and an offsets.
- virtual **NTFloatFun** & **operator**[ ] (const **IntFunPoR** &iter_) const=0
  Returns a new object with a different iterator.
- virtual **NTFloatFun** & **operator**[ ] (int offset_) const=0
  Returns a new object with a different offset.
- virtual const **FillIterator** ∗ **getIterator** () const
  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ∼**NTFloatFun** ()
  Protected destructor ensures creation on the heap.
- std::string **getBaseName** () const
  Returns the part of the name before the "[".

Protected Attributes

- const **IntFun** ∗ **iter**
  An optional **FillIterator**.
- int **offset**
  An optional offset, i.e. array index.

Related Functions

(Note that these are not member functions.)

- template<class Ntuple, class T> **NTFloatFunT**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_="?", const **IntFunPoR** &iter_=0)
  returns a **FloatFun** *from an ntuple data member*
- template<class Ntuple, class T> **NTFloatFunT**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_="?", int offset_)

returns a **FloatFun** *from an ntuple data member*

- template<class Ntuple, class T> **NTFloatFunG**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::∗p_mem)(), const char ∗name_="?")

  returns a **FloatFun** *from an ntuple access function*

- template<class Ntuple, class T> **NTFloatFunG**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::∗p_mem)() const, const char ∗name_="?")

  returns a **FloatFun** *from an ntuple access function*

- template<class Ntuple, class T, class A> **NTFloatFunGA**< Ntuple, T, A > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::∗p_mem)(A), A arg_, const char ∗name_="?")

  returns a **FloatFun** *from an ntuple access function with an argument*

- template<class Ntuple, class T, class A> **NTFloatFunGA**< Ntuple, T, A > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::∗p_mem)(A) const, A arg_, const char ∗name_="?")

  returns a **FloatFun** *from an ntuple data member*

- template<class T> **NTFloatFunGF**< T > & **ntfloatfun** (T(∗p_mem)(), const char ∗name_="?")

  returns a **FloatFun** *from an ntuple data member*

6.66.2 Constructor & Destructor Documentation

6.66.2.1 NTFloatFun::NTFloatFun (const std::string & *name_*)

Constructor from a name.

Parameters:

name_ The name

Definition at line 24 of file NTFloatFun.C.

References NamedO::getName().

6.66.2.2 NTFloatFun::NTFloatFun (const std::string & *name_*, const IntFun ∗ *iter_*)

Constructor from a name and an iterator.

Parameters:

name_ The name

iter_ The Iterator

Definition at line 34 of file NTFloatFun.C.

References NamedO::getName().

6.66.2.3 NTFloatFun::NTFloatFun (const std::string & *name_*, int *offset_*)

Constructor from a name and an offsets.

Parameters:

name_ The name

offset_ The integer offset

Definition at line 44 of file NTFloatFun.C.

References NamedO::getName().

6.66.3 Member Function Documentation

6.66.3.1 virtual NTFloatFun& NTFloatFun::operator[] (const IntFunPoR & *iter_*) const `[pure virtual]`

Returns a new object with a different iterator.

Parameters:

iter_ the new iterator

Implemented in **NTFloatFunT< Ntuple, T >** (p. 258), **NTFloatFunG< Ntuple, T >** (p. 248), **NTFloatFunGA< Ntuple, T, A >** (p. 251), and **NTFloatFunGF< T >** (p. 254).

6.66.3.2 virtual NTFloatFun& NTFloatFun::operator[] (int *offset_*) const `[pure virtual]`

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implemented in **NTFloatFunT< Ntuple, T >** (p. 259), **NTFloatFunG< Ntuple, T >** (p. 248), **NTFloatFunGA< Ntuple, T, A >** (p. 251), and **NTFloatFunGF< T >** (p. 254).

6.66.4 Friends And Related Function Documentation

6.66.4.1 template<class Ntuple, class T> NTFloatFunT< Ntuple, T > & ntfloatfun (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", const IntFunPoR & *iter_* = 0) `[related]`

returns a **FloatFun** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 390 of file NTFloatFun.h.

6.66.4.2 template<class Ntuple, class T> NTFloatFunT< Ntuple, T > & ntfloatfun (const Ntuple & *nt*, T Ntuple::∗ *p_mem*, const char ∗ *name_* = "?", int *offset_*) `[related]`

returns a **FloatFun** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset into an array

Definition at line 403 of file NTFloatFun.h.

6.66.4.3 template<class Ntuple, class T> NTFloatFunG< Ntuple, T > & ntfloatfun (const Ntuple & *nt*, T(Ntuple::∗ *p_mem*)(), const char ∗ *name_* = "?") `[related]`

returns a **FloatFun** from an ntuple access function

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

Definition at line 416 of file NTFloatFun.h.

6.66.4.4 template<class Ntuple, class T> NTFloatFunG< Ntuple, T > & ntfloatfun (const Ntuple & *nt*, T(Ntuple::∗ *p_mem*)() const, const char ∗ *name_* = "?") `[related]`

returns a **FloatFun** from an ntuple access function

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

Definition at line 427 of file NTFloatFun.h.

6.66.4.5 template<class Ntuple, class T, class A> NTFloatFunGA< Ntuple, T, A > & ntfloatfun (const Ntuple & *nt*, T(Ntuple::∗ *p_mem*)(A), A *arg_*, const char ∗ *name_* = "?") `[related]`

returns a **FloatFun** from an ntuple access function with an argument

Parameters:

nt The ntuple

p_mem Member pointer

arg_ The argument value

name_ The object's name

Definition at line 439 of file NTFloatFun.h.

6.66.4.6 template<class Ntuple, class T, class A> NTFloatFunGA< Ntuple, T, A > & ntfloatfun (const Ntuple & *nt*, T(Ntuple::* *p_mem*)(A) const, A *arg_*, const char * *name_* = "?") [related]

returns a **FloatFun** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

arg_ The argument value

name_ The object's name

Definition at line 451 of file NTFloatFun.h.

6.66.4.7 template<class T> NTFloatFunGF< T > & ntfloatfun (T(* *p_mem*)(), const char * *name_* = "?") [related]

returns a **FloatFun** from an ntuple data member

Parameters:

p_mem Member pointer

name_ The object's name

Definition at line 464 of file NTFloatFun.h.

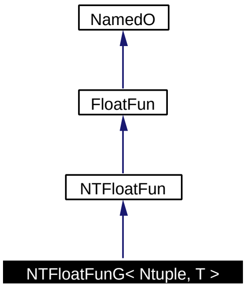
The documentation for this class was generated from the following files:

- **NTFloatFun.h**
- **NTFloatFun.C**

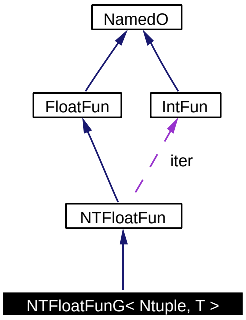
6.67 NTFloatFunG< Ntuple, T > Class Template Reference

`#include <NTFloatFun.h>`

Inheritance diagram for NTFloatFunG< Ntuple, T >:



Collaboration diagram for NTFloatFunG< Ntuple, T >:



6.67.1 Detailed Description

template<class Ntuple, class T = Float_t> class NTFloatFunG< Ntuple, T >

Returns a Float_FF value from an Ntuple getter method.

An NTFloatFunG collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a Float_t value from a given branch.

Author: Benno List, Jenny Böhme

Date

2006/10/25 15:06:59

Author

msteder

Changelog:

- 30.5.06 BL: First version

Definition at line 216 of file NTFloatFun.h.

Public Types

- typedef T(Ntuple::* **Getter**)()
- typedef T(Ntuple::* **ConstGetter**)() const

Public Member Functions

- **NTFloatFunG** (const Ntuple &nt_, Getter p_mem, const char *name_="?")

  Constructor from a member pointer.

- **NTFloatFunG** (const Ntuple &nt_, ConstGetter p_mem, const char *name_="?")
- virtual Float_FF **operator()** () const

  Returns a Float_FF value from an ntuple row.

- virtual **NTFloatFun** & **operator**[] (const **IntFunPoR** &iter_) const

  Returns a new object with a different iterator.

- virtual **NTFloatFun** & **operator**[] (int offset_) const

  Returns a new object with a different offset.

Protected Member Functions

- virtual ~**NTFloatFunG** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- Ntuple & **nt**
- Getter **getMethod**

6.67.2 Constructor & Destructor Documentation

6.67.2.1 template<class Ntuple, class T = Float_t> NTFloatFunG< Ntuple, T >::NTFloatFunG (const Ntuple & *nt_*, Getter *p_mem*, const char * *name_* = "?") `[inline]`

Constructor from a member pointer.

Parameters:

nt_ The ntuple

p_mem Member pointer

name_ The object's name

Definition at line 222 of file NTFloatFun.h.

6.67.2.2 template<class Ntuple, class T = Float_t> NTFloatFunG< Ntuple, T >::NTFloatFunG (const Ntuple & *nt_*, ConstGetter *p_mem*, const char * *name_* = "?") `[inline]`

Parameters:

*nt_* The ntuple

p_mem Member pointer

*name_* The object's name

Definition at line 229 of file NTFloatFun.h.

6.67.3 Member Function Documentation

6.67.3.1 template<class Ntuple, class T = Float_t> virtual NTFloatFun& NTFloatFunG< Ntuple, T >::operator[] (const IntFunPoR & *iter_*) const `[inline, virtual]`

Returns a new object with a different iterator.

Parameters:

*iter_* the new iterator

Implements **NTFloatFun** (p. 243).

Definition at line 244 of file NTFloatFun.h.

6.67.3.2 template<class Ntuple, class T = Float_t> virtual NTFloatFun& NTFloatFunG< Ntuple, T >::operator[] (int *offset_*) const `[inline, virtual]`

Returns a new object with a different offset.

Parameters:

*offset_* the new offset

Implements **NTFloatFun** (p. 243).

Definition at line 249 of file NTFloatFun.h.

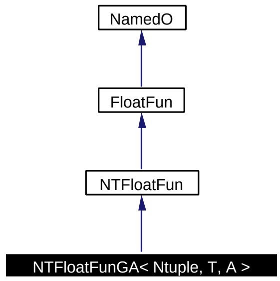
The documentation for this class was generated from the following file:

- **NTFloatFun.h**

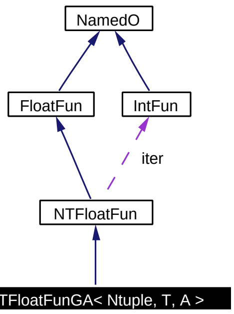
6.68 NTFloatFunGA< Ntuple, T, A > Class Template Reference

`#include <NTFloatFun.h>`

Inheritance diagram for NTFloatFunGA< Ntuple, T, A >:



Collaboration diagram for NTFloatFunGA< Ntuple, T, A >:



6.68.1 Detailed Description

template<class Ntuple, class T = Float_t, class A = Float_t> class NTFloatFunGA< Ntuple, T, A >

Returns a Float_FF value from an Ntuple getter method with an argument.

An NTFloatFunGA collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a Float_t value from a given branch.

Author: Benno List, Jenny Böhme

Date
2006/10/25 15:06:59

Author
msteder

Changelog:

- 25.10.06 BL: First version

Definition at line 280 of file NTFloatFun.h.

Public Types

- typedef T(Ntuple::∗ **Getter**)(A)
- typedef T(Ntuple::∗ **ConstGetter**)(A) const

Public Member Functions

- **NTFloatFunGA** (const Ntuple &nt_, Getter p_mem, A arg_, const char ∗name_="?")

  Constructor from a member pointer.

- **NTFloatFunGA** (const Ntuple &nt_, ConstGetter p_mem, A arg_, const char ∗name_="?")
- virtual Float_FF **operator()** () const

  Returns a Float_FF value from an ntuple row.

- virtual **NTFloatFun** & **operator**[] (const **IntFunPoR** &iter_) const

  Returns a new object with a different iterator.

- virtual **NTFloatFun** & **operator**[] (int offset_) const

  Returns a new object with a different offset.

Protected Member Functions

- virtual ∼**NTFloatFunGA** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- Ntuple & **nt**
- Getter **getMethod**
- A **arg**

6.68.2 Constructor & Destructor Documentation

6.68.2.1 template<class Ntuple, class T = Float_t, class A = Float_t> NTFloatFunGA< Ntuple, T, A >::NTFloatFunGA (const Ntuple & *nt_*, Getter *p_mem*, A *arg_*, const char ∗ *name_* = "?") `[inline]`

Constructor from a member pointer.

Parameters:

nt_ The ntuple

p_mem Member pointer

arg_ The argument of the getter function

name_ The object's name

Definition at line 286 of file NTFloatFun.h.

6.68.2.2 template<class Ntuple, class T = Float_t, class A = Float_t> NTFloatFunGA< Ntuple, T, A >::NTFloatFunGA (const Ntuple & *nt_*, ConstGetter *p_mem*, A *arg_*, const char * *name_* = "?") [inline]

Parameters:

nt_ The ntuple

p_mem Member pointer

arg_ The argument of the getter function

name_ The object's name

Definition at line 294 of file NTFloatFun.h.

6.68.3 Member Function Documentation

6.68.3.1 template<class Ntuple, class T = Float_t, class A = Float_t> virtual NTFloatFun& NTFloatFunGA< Ntuple, T, A >::operator[] (const IntFunPoR & *iter_*) const [inline, virtual]

Returns a new object with a different iterator.

Parameters:

iter_ the new iterator

Implements **NTFloatFun** (p. 243).

Definition at line 311 of file NTFloatFun.h.

6.68.3.2 template<class Ntuple, class T = Float_t, class A = Float_t> virtual NTFloatFun& NTFloatFunGA< Ntuple, T, A >::operator[] (int *offset_*) const [inline, virtual]

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implements **NTFloatFun** (p. 243).

Definition at line 316 of file NTFloatFun.h.

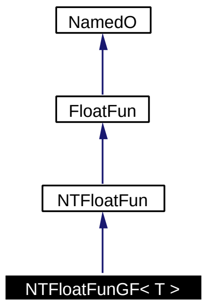
The documentation for this class was generated from the following file:

- **NTFloatFun.h**

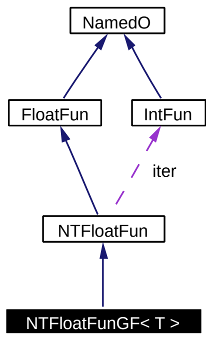
6.69 NTFloatFunGF< T > Class Template Reference

`#include <NTFloatFun.h>`

Inheritance diagram for NTFloatFunGF< T >:



Collaboration diagram for NTFloatFunGF< T >:



6.69.1 Detailed Description

template<class T = Float_t> class NTFloatFunGF< T >

Returns a Float_FF value from free getter function.

An **NTFloatFunG** collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a Float_t value from a given branch.

Author: Benno List, Jenny Böhme

Date

2006/10/25 15:06:59

Author

msteder

Changelog:

- 30.5.06 BL: First version

Definition at line 348 of file NTFloatFun.h.

Public Types

- typedef T(* **Getter**)()

Public Member Functions

- **NTFloatFunGF** (Getter p_mem, const char *name_="?")

  Constructor from a member pointer.

- virtual Float_FF **operator()** () const

  Returns a Float_FF value from an ntuple row.

- virtual **NTFloatFun** & **operator**[ ] (const **IntFunPoR** &iter_) const

  Returns a new object with a different iterator.

- virtual **NTFloatFun** & **operator**[ ] (int offset_) const

  Returns a new object with a different offset.

Protected Member Functions

- virtual ~**NTFloatFunGF** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- Getter **getMethod**

6.69.2 Constructor & Destructor Documentation

6.69.2.1 template<class T = Float_t> NTFloatFunGF< T >::NTFloatFunGF (Getter *p_mem*, const char * *name_* = "?") `[inline]`

Constructor from a member pointer.

Parameters:

p_mem Member pointer

name_ The object's name

Definition at line 353 of file NTFloatFun.h.

6.69.3 Member Function Documentation

6.69.3.1 template<class T = Float_t> virtual NTFloatFun& NTFloatFunGF< T >::operator[] (const IntFunPoR & *iter_*) const `[inline, virtual]`

Returns a new object with a different iterator.

Parameters:

iter_ the new iterator

Implements **NTFloatFun** (p. 243).

Definition at line 367 of file NTFloatFun.h.

6.69.3.2 template<class T = Float_t> virtual NTFloatFun& NTFloatFunGF< T >::operator[] (int *offset_*) const `[inline, virtual]`

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implements **NTFloatFun** (p. 243).

Definition at line 372 of file NTFloatFun.h.

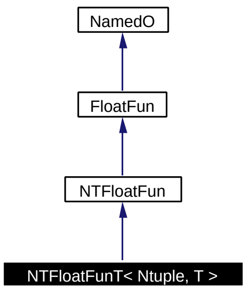
The documentation for this class was generated from the following file:

- **NTFloatFun.h**

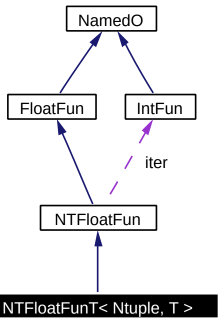
6.70 NTFloatFunT< Ntuple, T > Class Template Reference

`#include <NTFloatFun.h>`

Inheritance diagram for NTFloatFunT< Ntuple, T >:



Collaboration diagram for NTFloatFunT< Ntuple, T >:



6.70.1 Detailed Description

template<class Ntuple, class T = Float_t> class NTFloatFunT< Ntuple, T >

Returns a Float_FF value from an Ntuple.

An NTFloatFunT collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a Float_t value from a given branch.

The first template argument is the class name of the ntuple class that was automatically generated by TTree::MakeClass. The second (optional) template argument is the type of the branch variable; this is useful if the branch contains not Float_t objects, but double or even int values.

Author: Benno List, Jenny Böhme

Date

2006/10/25 15:06:59

Author

msteder

Changelog:

- 15.7.04 BL: Added method **getIterator()**
- 20.7.04 BL: Use **IntFun** instead of **FillIterator**
- 8.4.05 BL: Add optional offset, and operator[]
- 6.5.04 BL: Adapted to name in base class, added new constructors needed for operator[]

Definition at line 101 of file NTFloatFun.h.

Public Member Functions

- **NTFloatFunT** (const Ntuple &nt, const char *branchname, const **IntFunPoR** &iter_=0)

  Constructor from a branch name given as string.

- **NTFloatFunT** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_="?", const **IntFunPoR** &iter_=0)

  Constructor from a member pointer.

- **NTFloatFunT** (const Ntuple &nt, const char *branchname, int offset_)

  Constructor from a branch name given as string, with offset.

- **NTFloatFunT** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_, int offset_)

  Constructor from a member pointer, with offset.

- **NTFloatFunT** (const **NTFloatFunT**< Ntuple, T > &source, const **IntFunPoR** &iter_)

  Constructor from an existing NTFloatFunT object, plus a new iterator.

- **NTFloatFunT** (const **NTFloatFunT**< Ntuple, T > &source, const int offset_)

  Constructor from an existing NTFloatFunT object, plus a new offset.

- virtual Float_FF **operator()** () const

  Returns a Float_FF value from an ntuple row.

- **NTFloatFunT**< Ntuple, T > & **operator[]** (const **IntFunPoR** &iter_) const

  Returns a new object with a different iterator.

- **NTFloatFunT**< Ntuple, T > & **operator[]** (int offset_) const

  Returns a new object with a different offset.

Protected Member Functions

- virtual ~**NTFloatFunT** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const T * **branchAddress**

  The branch adress.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const Ntuple & *nt*, const char * *branchname*, const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a branch name given as string.

Parameters:

nt The ntuple

branchname The branch name

iter_ An optional iterator

Definition at line 104 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress.

6.70.2.2 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a member pointer.

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 119 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress.

6.70.2.3 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const Ntuple & *nt*, const char * *branchname*, int *offset_*) `[inline]`

Constructor from a branch name given as string, with offset.

Parameters:

nt The ntuple

branchname The branch name

offset_ The offset

Definition at line 131 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress.

6.70.2.4 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_*, int *offset_*) [inline]

Constructor from a member pointer, with offset.

Parameters:

nt The ntuple

p_mem Member pointer

*name_* The object's name

*offset_* The offset

Definition at line 146 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress.

6.70.2.5 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const NTFloatFunT< Ntuple, T > & *source*, const IntFunPoR & *iter_*) [inline]

Constructor from an existing NTFloatFunT object, plus a new iterator.

Parameters:

*iter_* The new iterator

Definition at line 158 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress, and NTFloatFun::getBaseName().

6.70.2.6 template<class Ntuple, class T = Float_t> NTFloatFunT< Ntuple, T >::NTFloatFunT (const NTFloatFunT< Ntuple, T > & *source*, const int *offset_*) [inline]

Constructor from an existing NTFloatFunT object, plus a new offset.

Parameters:

*offset_* The new offset

Definition at line 165 of file NTFloatFun.h.

References NTFloatFunT< Ntuple, T >::branchAddress, and NTFloatFun::getBaseName().

6.70.3 Member Function Documentation

6.70.3.1 template<class Ntuple, class T = Float_t> NTFloatFunT<Ntuple, T>& NTFloatFunT< Ntuple, T >::operator[] (const IntFunPoR & *iter_*) const [inline, virtual]

Returns a new object with a different iterator.

Parameters:

*iter_* the new iterator

Implements **NTFloatFun** (p. 243).

Definition at line 180 of file NTFloatFun.h.

6.70.3.2 template<class Ntuple, class T = Float_t> NTFloatFunT<Ntuple, T>& NTFloatFunT< Ntuple, T >::operator[] (int *offset_*) const `[inline, virtual]`

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implements **NTFloatFun** (p. 243).

Definition at line 186 of file NTFloatFun.h.

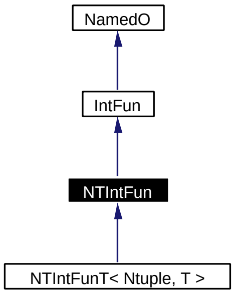
The documentation for this class was generated from the following file:

- **NTFloatFun.h**

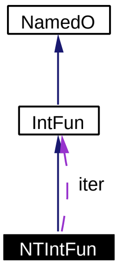
6.71 NTIntFun Class Reference

`#include <NTIntFun.h>`

Inheritance diagram for NTIntFun:



Collaboration diagram for NTIntFun:



6.71.1 Detailed Description

Returns a int value from an Ntuple (abstract base class for **NTIntFunT** templates).

Author: Benno List, Jenny List

Date

2005/06/04 16:27:51

Author

blist

Changelog:

- 8.4.05 BL: First version

Definition at line 32 of file NTIntFun.h.

Public Member Functions

- **NTIntFun** (const std::string &name_)

Constructor from a name.

- **NTIntFun** (const std::string &name_, const **IntFun** *iter_)

  Constructor from a name and an iterator.

- **NTIntFun** (const std::string &name_, int offset_)

  Constructor from a name and an offsets.

- virtual **NTIntFun** & **operator**[ ] (const **IntFunPoR** &iter_) const=0

  Returns a new object with a different iterator.

- virtual **NTIntFun** & **operator**[ ] (int offset_) const=0

  Returns a new object with a different offset.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ~**NTIntFun** ()

  Protected destructor ensures creation on the heap.

- std::string **getBaseName** () const

  Returns the part of the name before the "[".

Protected Attributes

- const **IntFun** * **iter**

  An optional **FillIterator**.

- int **offset**

  An optional offset, i.e. array index.

Related Functions

(Note that these are not member functions.)

- template<class Ntuple, class T> **NTIntFunT**< Ntuple, T > & **ntintfun** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_="?", const **IntFunPoR** &iter_=0)

  returns a **IntFun** *from an ntuple data member*

- template<class Ntuple, class T> **NTIntFunT**< Ntuple, T > & **ntintfun** (const Ntuple &nt, T Ntuple::*p_mem, const char *name_="?", int offset_)

  returns a **IntFun** *from an ntuple data member*

6.71.2 Constructor & Destructor Documentation

6.71.2.1 NTIntFun::NTIntFun (const std::string & *name_*)

Constructor from a name.

Parameters:
- ***name_*** The name

Definition at line 19 of file NTIntFun.C.

References NamedO::getName().

6.71.2.2 NTIntFun::NTIntFun (const std::string & *name_*, const IntFun ∗ *iter_*)

Constructor from a name and an iterator.

Parameters:
- ***name_*** The name
- ***iter_*** The Iterator

Definition at line 29 of file NTIntFun.C.

References NamedO::getName().

6.71.2.3 NTIntFun::NTIntFun (const std::string & *name_*, int *offset_*)

Constructor from a name and an offsets.

Parameters:
- ***name_*** The name
- ***offset_*** The integer offset

Definition at line 39 of file NTIntFun.C.

References NamedO::getName().

6.71.3 Member Function Documentation

6.71.3.1 virtual NTIntFun& NTIntFun::operator[] (const IntFunPoR & *iter_*) const `[pure virtual]`

Returns a new object with a different iterator.

Parameters:
- ***iter_*** the new iterator

Implemented in **NTIntFunT< Ntuple, T >** (p. 267).

6.71.3.2 virtual NTIntFun& NTIntFun::operator[] (int *offset_*) const [pure virtual]

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implemented in **NTIntFunT< Ntuple, T >** (p. 268).

6.71.4 Friends And Related Function Documentation

6.71.4.1 template<class Ntuple, class T> NTIntFunT< Ntuple, T > & ntintfun (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", const IntFunPoR & *iter_* = 0) [related]

returns a **IntFun** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 206 of file NTIntFun.h.

6.71.4.2 template<class Ntuple, class T> NTIntFunT< Ntuple, T > & ntintfun (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", int *offset_*) [related]

returns a **IntFun** from an ntuple data member

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset into an array

Definition at line 219 of file NTIntFun.h.

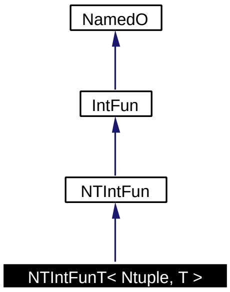
The documentation for this class was generated from the following files:

- **NTIntFun.h**
- **NTIntFun.C**

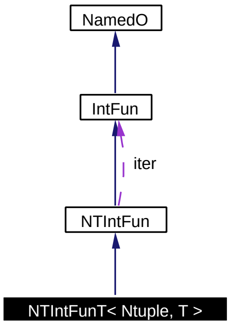
6.72 NTIntFunT< Ntuple, T > Class Template Reference

`#include <NTIntFun.h>`

Inheritance diagram for NTIntFunT< Ntuple, T >:



Collaboration diagram for NTIntFunT< Ntuple, T >:



6.72.1 Detailed Description

template<class Ntuple, class T = Int_t> class NTIntFunT< Ntuple, T >

Returns a int value from an Ntuple.

An **NTIntFun** collaborates with a class Ntuple, which automatically generated by TTree::MakeClass, and returns a int value from a given branch.

The first template argument is the class name of the ntuple class that was automatically generated by TTree::MakeClass. The second (optional) template argument is the type of the branch variable; this is useful if the branch contains not Int_t objects, but short or char values.

Author: Benno List, Jenny Böhme

Date

2005/06/04 16:27:51

Author

blist

Changelog:

- 15.7.04 BL: Added iterator and method **getIterator()**
- 20.7.04 BL: Use **IntFun** instead of **FillIterator**
- 14.3.05 BL, JL: Add template for ntintfun access to arrays
- 6.4.05 BL: Add optional offset, and operator[]
- 8.4.05 BL: Renamed NTIntFunT, and derived from **NTIntFun**
- 6.5.04 BL: Adapted to name in base class, added new constructors needed for operator[]

Definition at line 102 of file NTIntFun.h.

Public Member Functions

- **NTIntFunT** (const Ntuple &nt, const char ∗branchname, const **IntFunPoR** &iter_=0)
  Constructor from a branch name given as string.
- **NTIntFunT** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_="?", const **IntFunPoR** &iter_=0)
  Constructor from a member pointer.
- **NTIntFunT** (const Ntuple &nt, const char ∗branchname, int offset_)
  Constructor from a branch name given as string, with offset.
- **NTIntFunT** (const Ntuple &nt, T Ntuple::∗p_mem, const char ∗name_, int offset_)
  Constructor from a member pointer, with offset.
- **NTIntFunT** (const **NTIntFunT**< Ntuple, T > &source, const **IntFunPoR** &iter_)
  Constructor from an existing NTIntFunT object, plus a new iterator.
- **NTIntFunT** (const **NTIntFunT**< Ntuple, T > &source, const int offset_)
  Constructor from an existing NTIntFunT object, plus a new offset.
- virtual int **operator()** () const
  Returns an int value from an ntuple row.
- **NTIntFunT**< Ntuple, T > & **operator[ ]** (const **IntFunPoR** &iter_) const
  Returns a new object with a different iterator.
- **NTIntFunT**< Ntuple, T > & **operator[ ]** (int offset_) const
  Returns a new object with a different offset.

Protected Member Functions

- virtual ∼**NTIntFunT** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const T * **branchAddress**
  The branch adress.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const Ntuple & *nt*, const char * *branchname*, const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a branch name given as string.

Parameters:
nt The ntuple
branchname The branch name
iter_ An optional iterator

Definition at line 105 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress.

6.72.2.2 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_* = "?", const IntFunPoR & *iter_* = 0) `[inline]`

Constructor from a member pointer.

Parameters:
nt The ntuple
p_mem Member pointer
name_ The object's name
iter_ An optional iterator

Definition at line 120 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress.

6.72.2.3 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const Ntuple & *nt*, const char * *branchname*, int *offset_*) `[inline]`

Constructor from a branch name given as string, with offset.

Parameters:
nt The ntuple
branchname The branch name
offset_ The offset

Definition at line 132 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress.

6.72.2.4 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const Ntuple & *nt*, T Ntuple::* *p_mem*, const char * *name_*, int *offset_*) [inline]

Constructor from a member pointer, with offset.

Parameters:

nt The ntuple

p_mem Member pointer

*name_* The object's name

*offset_* The offset

Definition at line 147 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress.

6.72.2.5 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const NTIntFunT< Ntuple, T > & *source*, const IntFunPoR & *iter_*) [inline]

Constructor from an existing NTIntFunT object, plus a new iterator.

Parameters:

*iter_* The new iterator

Definition at line 159 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress, and NTIntFun::getBaseName().

6.72.2.6 template<class Ntuple, class T = Int_t> NTIntFunT< Ntuple, T >::NTIntFunT (const NTIntFunT< Ntuple, T > & *source*, const int *offset_*) [inline]

Constructor from an existing NTIntFunT object, plus a new offset.

Parameters:

*offset_* The new offset

Definition at line 166 of file NTIntFun.h.

References NTIntFunT< Ntuple, T >::branchAddress, and NTIntFun::getBaseName().

6.72.3 Member Function Documentation

6.72.3.1 template<class Ntuple, class T = Int_t> NTIntFunT<Ntuple, T>& NTIntFunT< Ntuple, T >::operator[] (const IntFunPoR & *iter_*) const [inline, virtual]

Returns a new object with a different iterator.

Parameters:

*iter_* the new iterator

Implements **NTIntFun** (p. 262).

Definition at line 180 of file NTIntFun.h.

References IntFunPoR::pif.

6.72.3.2 template<class Ntuple, class T = Int_t> NTIntFunT<Ntuple, T>& NTIntFunT< Ntuple, T >::operator[] (int *offset_*) const `[inline, virtual]`

Returns a new object with a different offset.

Parameters:

offset_ the new offset

Implements **NTIntFun** (p. 263).

Definition at line 186 of file NTIntFun.h.

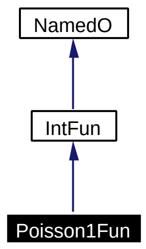
The documentation for this class was generated from the following file:

- **NTIntFun.h**

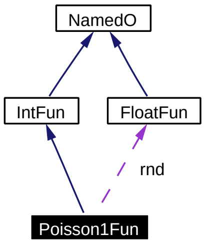
6.73 Poisson1Fun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for Poisson1Fun:



Collaboration diagram for Poisson1Fun:



6.73.1 Detailed Description

A **IntFun** that returns a Poisson-distributed value from a uniform random number.

Definition at line 128 of file RandomFuns.h.

Public Member Functions

- **Poisson1Fun** (const **FloatFun** &rnd_, double mean_, const char *name_=0)

  Constructor from a C style string, serves as default constructor.

- **Poisson1Fun** (const **FloatFun** &rnd_, double mean_, const std::string &name_)

  Constructor from a C++ string.

- virtual int **operator()** () const

  Returns a poisson distributed value.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ~**Poisson1Fun** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const **FloatFun** & **rnd**

  The random function.

- double **mean**

6.73.2 Constructor & Destructor Documentation

6.73.2.1 Poisson1Fun::Poisson1Fun (const FloatFun & *rnd_*, double *mean_*, const char * *name_* = 0) [explicit]

Constructor from a C style string, serves as default constructor.

Parameters:

rnd_ The random function

mean_ The mean value

name_ An optional name

Definition at line 135 of file RandomFuns.C.

6.73.2.2 Poisson1Fun::Poisson1Fun (const FloatFun & *rnd_*, double *mean_*, const std::string & *name_*) [inline, explicit]

Constructor from a C++ string.

Parameters:

rnd_ The random function

mean_ The mean value

name_ An optional name

Definition at line 137 of file RandomFuns.h.

References mean, and rnd.

6.73.3 Member Data Documentation

6.73.3.1 double Poisson1Fun::mean [protected]

The mean value

Definition at line 154 of file RandomFuns.h.

Referenced by operator()(), and Poisson1Fun().

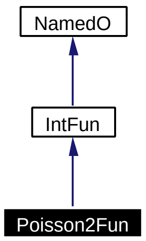
The documentation for this class was generated from the following files:

- **RandomFuns.h**
- **RandomFuns.C**

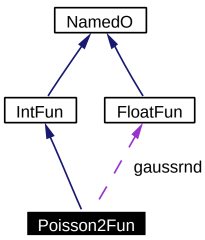
6.74 Poisson2Fun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for Poisson2Fun:



Collaboration diagram for Poisson2Fun:



6.74.1 Detailed Description

A **IntFun** that returns a Poisson-distributed value from a uniform random number (gaussian approximation).

Definition at line 161 of file RandomFuns.h.

Public Member Functions

- **Poisson2Fun** (const **FloatFun** &rnd_, double mean_, const char *name_=0)

  Constructor from a C style string, serves as default constructor.

- **Poisson2Fun** (const **FloatFun** &rnd_, double mean_, const std::string &name_)

  Constructor from a C++ string.

- virtual int **operator()** () const

  Returns a poisson distributed value.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Protected Member Functions

- virtual ~**Poisson2Fun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- **FloatFun** & **gaussrnd**

6.74.2 Constructor & Destructor Documentation

6.74.2.1 Poisson2Fun::Poisson2Fun (const FloatFun & *rnd_*, double *mean_*, const char * *name_* = 0) `[explicit]`

Constructor from a C style string, serves as default constructor.

Parameters:

*rnd_* The random function

*mean_* The mean value

*name_* An optional name

Definition at line 157 of file RandomFuns.C.

6.74.2.2 Poisson2Fun::Poisson2Fun (const FloatFun & *rnd_*, double *mean_*, const std::string & *name_*) `[explicit]`

Constructor from a C++ string.

Parameters:

*rnd_* The random function

*mean_* The mean value

*name_* An optional name

Definition at line 162 of file RandomFuns.C.

6.74.3 Member Data Documentation

6.74.3.1 FloatFun& Poisson2Fun::gaussrnd `[protected]`

The gaussian random function

Definition at line 182 of file RandomFuns.h.

Referenced by getIterator(), operator()(), and ~Poisson2Fun().

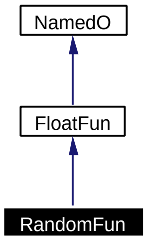
The documentation for this class was generated from the following files:

- **RandomFuns.h**
- **RandomFuns.C**

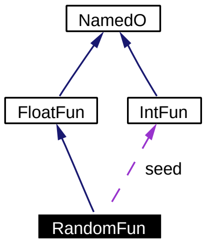
6.75 RandomFun Class Reference

`#include <RandomFuns.h>`

Inheritance diagram for RandomFun:



Collaboration diagram for RandomFun:



6.75.1 Detailed Description

A **FloatFun** that returns a Random value, taking an intfun as seed.

Use different index values for different objects to get different results for the same seed

Definition at line 39 of file RandomFuns.h.

Public Member Functions

- **RandomFun** (const **IntFun** &seed_, int index_, const char *name_=0)
- **RandomFun** (const **IntFun** &seed_, int index_, const std::string &name_)

  Constructor from a C++ string.

- **RandomFun** (const **IntFun** &seed1_, const **IntFun** &seed2_, int index_, const char *name_-=0)
- **RandomFun** (const **IntFun** &seed1_, const **IntFun** &seed2_, int index_, const std::string &name_)
- virtual Float_FF **operator()** () const

  Returns a random number that depends on **seed()** *and index.*

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator, if any.

Static Public Member Functions

- long long **iterate** (long long i)
- long long **toInt** (Float_FF d)
- Float_FF **toDouble** (long long i)
- Float_FF **generateNext** (Float_FF d)

Protected Member Functions

- virtual ~**RandomFun** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const **IntFun** & **seed**

  The seed function object.

- int **index**

  The index.

- bool **cleanup**

Static Private Attributes

- const long long **a** = 16807LL
- const long long **c** = 0LL
- const long long **m** = 2147483647LL

6.75.2 Constructor & Destructor Documentation

6.75.2.1 RandomFun::RandomFun (const IntFun & *seed_*, int *index_*, const char * *name_* = 0) [explicit]

Parameters:

seed_ The seed function object

index_ The index

name_ The object's name

Definition at line 40 of file RandomFuns.C.

6.75.2.2 RandomFun::RandomFun (const IntFun & *seed_*, int *index_*, const std::string & *name_*)

Constructor from a C++ string.

Parameters:

seed_ The seed function object

index_ The index

name_ The object's name

Definition at line 48 of file RandomFuns.C.

6.75.2.3 RandomFun::RandomFun (const IntFun & *seed1_*, const IntFun & *seed2_*, int *index_*, const char * *name_* = 0)

Parameters:

seed1_ The 1st seed function object

seed2_ The 2nd seed function object

index_ The index

name_ The object's name

Definition at line 58 of file RandomFuns.C.

6.75.2.4 RandomFun::RandomFun (const IntFun & *seed1_*, const IntFun & *seed2_*, int *index_*, const std::string & *name_*)

Parameters:

seed1_ The 1st seed function object

seed2_ The 2nd seed function object

index_ The index

name_ The object's name

Definition at line 66 of file RandomFuns.C.

6.75.3 Member Data Documentation

6.75.3.1 bool RandomFun::cleanup `[protected]`

Flag for clsen up

Definition at line 83 of file RandomFuns.h.

Referenced by ~RandomFun().

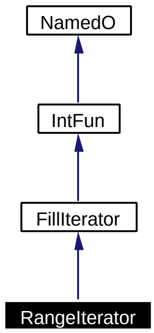
The documentation for this class was generated from the following files:

- **RandomFuns.h**
- **RandomFuns.C**

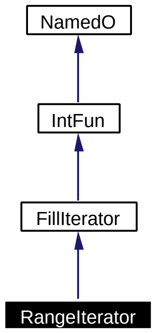
6.76 RangeIterator Class Reference

`#include <FillIterators.h>`

Inheritance diagram for RangeIterator:



Collaboration diagram for RangeIterator:



6.76.1 Detailed Description

FillIterator that iterates over a constant range of values.

Changelog

- 6.7.04 BL: First version
- 4.6.05 BL: Added name

Author: Benno List, Jenny Böhme

Date
2005/06/09 16:42:59

Author
rweber

Definition at line 21 of file FillIterators.h.

Public Member Functions

- **RangeIterator** (int min_, int max_)
  Constructor.
- **RangeIterator** (int min_, int max_, const std::string &name_)
- virtual int **operator()** () const
  Returns iterator value, starting at min.
- virtual bool **next** ()
  Increments iterator; returns false if iterator cannot be incremented.
- virtual bool **reset** ()
  Resets iterator; returns false if iterator value range is empty.
- virtual bool **isValid** () const
  Returns whether current value of iterator is valid.

Protected Member Functions

- virtual ~**RangeIterator** ()
  Protected destructor to ensure creation on the heap.

Private Attributes

- int **min**
  Minimum value.
- int **max**
  Maximum value.
- int **index**
  Current value.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 RangeIterator::RangeIterator (int *min_*, int *max_*) `[inline]`

Constructor.

Parameters:

min_ Minimum value

max_ Maximum value

Definition at line 24 of file FillIterators.h.

References index, max, min, and NamedO::str().

6.76.2.2 RangeIterator::RangeIterator (int *min_*, int *max_*, const std::string & *name_*) [inline]

Parameters:

min_ Minimum value

max_ Maximum value

name_ The object's name

Definition at line 32 of file FillIterators.h.

References index, max, and min.

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

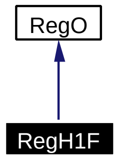
- **FillIterators.h**

6.77 RegH1F Class Reference

`#include <RegH1F.h>`

Inheritance diagram for RegH1F:

Collaboration diagram for RegH1F:



6.77.1 Detailed Description

A registered 1-dimensional TH1F histogram.

Author: Benno List, Jenny Böhme

Date

2005/11/11 17:21:16

Author

rweber

Changelog:

- 4.6.04 BL: Add new constructor using a Binning object
- 8.6.04 BL: Use HListPoR
- 30.6.04 Moved getHistInfo from **MatrixOfHistograms** here
- 12.5.05 BL: Added Divide methods
- 15.6.05 BL: Signature of the Divide method corrected (const TH1∗)

Definition at line 36 of file RegH1F.h.

Public Member Functions

- **RegH1F** (const char *name, const char *title, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl=0)

  Constructor with equidistant binning.

- **RegH1F** (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant binning, taking float bin edges.

- **RegH1F** (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant binning, taking double bin edges.

- **RegH1F** (const char *name, const char *title, const Binning &binning, const **ROListPoR** &hhl=0)

  Constructor taking a Binning object.

- **RegH1F** (const char *name, TFile &file, const **ROListPoR** &hhl)

  Constructor reading histogram from a file.

- **RegH1F** (const char *name, const char *title, const TH1F &source, const **ROListPoR** &hhl=0)

  Sort of "renaming copy constructor".

- virtual ~**RegH1F** ()

  Virtual destructor.

- virtual Stat_t **GetSumOfErrors2** () const

  Returns the sum of error squares excluding under/overflow bins.

- void **getHistInfo** (Option_t *option, Stat_t &content, Stat_t &error2) const

  Return content or mean and error.

- virtual void **Divide** (TF1 *f1, Double_t c1=1)

  *Divide by a function (as for TH1): this = this/(c1*f1).*

- virtual void **Divide** (const TH1 *h1)

  Divide by a histogram (as for TH1): this = this/h1.

- virtual void **Divide** (const TH1 *h1, const TH1 *h2, Double_t c1=1, Double_t c2=1, Option_t *option=0)

  *Replace contents by the division of h1 by h2: this = c1*h1/(c2*h2).*

- virtual void **Divide** (const TH1 *h1, const TH1 *h2, const TH1 *h3, Double_t c1=1, Double_t c2=1)

  *Replace contents by the division of h1 by h2, with h3 as overlap histogram: this = c1*h1/(c2*h2).*

- virtual void **DrawFunc** (Option_t *option, const char *fname)

  Enable a specific Fit Function to be plotted with the Histogram.

Private Member Functions

- **RegH1F** (const **RegH1F** &source)

  Private copy constructor, left unimplemented: No copying!

- **RegH1F** & **operator=** (const **RegH1F** &source)

  Private assignment operator, left unimplemented: No assignment!

6.77.2 Constructor & Destructor Documentation

6.77.2.1 RegH1F::RegH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl* = 0)

Constructor with equidistant binning.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

Definition at line 49 of file RegH1F.C.

6.77.2.2 RegH1F::RegH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Float_t * *xbins*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant binning, taking float bin edges.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

Definition at line 60 of file RegH1F.C.

6.77.2.3 RegH1F::RegH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant binning, taking double bin edges.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

Definition at line 70 of file RegH1F.C.

6.77.2.4 RegH1F::RegH1F (const char ∗ *name*, const char ∗ *title*, const Binning & *binning*, const ROListPoR & *hhl* = 0)

Constructor taking a Binning object.

Parameters:

name TH1F name

title TH1F title

binning Defines the histogram binning

hhl **ROList** object where "this" will be registered

Definition at line 80 of file RegH1F.C.

6.77.2.5 RegH1F::RegH1F (const char ∗ *name*, TFile & *file*, const ROListPoR & *hhl*)

Constructor reading histogram from a file.

Parameters:

name TH1F name

file File to read from

hhl **ROList** object where "this" will be registered

Definition at line 110 of file RegH1F.C.

6.77.2.6 RegH1F::RegH1F (const char ∗ *name*, const char ∗ *title*, const TH1F & *source*, const ROListPoR & *hhl* = 0)

Sort of "renaming copy constructor".

Parameters:

name New TH1F name

title New TH1F title

source Histogram to be copied

hhl **ROList** object where "this" will be registered

Definition at line 127 of file RegH1F.C.

6.77.3 Member Function Documentation

6.77.3.1 void RegH1F::getHistInfo (Option_t * *option*, Stat_t & *content*, Stat_t & *error2*) const

Return content or mean and error.

Parameters:
- ***option*** An option. "M" means "return mean" instead of contents
- ***content*** The content (return value)
- ***error2*** The squared error (return value)

Definition at line 155 of file RegH1F.C.

References GetSumOfErrors2().

Referenced by SetOfHistograms::getSummary(), MatrixOfHistograms::getSummary(), and MatrixOfHistograms::getSummarySet().

6.77.3.2 void RegH1F::Divide (TF1 * *f1*, Double_t *c1* = 1) [virtual]

Divide by a function (as for TH1): this = this/(c1*f1).

Parameters:
- ***f1*** The function f1
- ***c1*** An optional multiplier to f1

Definition at line 177 of file RegH1F.C.

Referenced by SetOfH::Divide().

6.77.3.3 void RegH1F::Divide (const TH1 * *h1*) [virtual]

Divide by a histogram (as for TH1): this = this/h1.

Parameters:
- ***h1*** The histogram h1

Definition at line 181 of file RegH1F.C.

6.77.3.4 void RegH1F::Divide (const TH1 * *h1*, const TH1 * *h2*, Double_t *c1* = 1, Double_t *c2* = 1, Option_t * *option* = 0) [virtual]

Replace contents by the division of h1 by h2: this = c1*h1/(c2*h2).

Parameters:
- ***h1*** The first histogram h1
- ***h2*** The second histogram h2
- ***c1*** An optional multiplier to h1
- ***c2*** An optional multiplier to h2
- ***option*** An option. "B" means binomial errors, "S" means subset, i.e. binomial errors for weighted events

Definition at line 185 of file RegH1F.C.

6.77.3.5 void RegH1F::Divide (const TH1 * *h1*, const TH1 * *h2*, const TH1 * *h3*, Double_t *c1* = 1, Double_t *c2* = 1) `[virtual]`

Replace contents by the division of h1 by h2, with h3 as overlap histogram: this = c1*h1/(c2*h2).

In the most general case where h1 contains a sum of event weights w_i and h2 a sum of event weights v_i, h3 should contain the sum w_i*v_i for those events which fall into the same x bin; h3 is the overlap histogram.

Parameters:

- ***h1*** The first histogram h1
- ***h2*** The second histogram h2
- ***h3*** The third histogram h3
- ***c1*** An optional multiplier to h1
- ***c2*** An optional multiplier to h2

Definition at line 289 of file RegH1F.C.

6.77.3.6 void RegH1F::DrawFunc (Option_t * *option*, const char * *fname*) `[virtual]`

Enable a specific Fit Function to be plotted with the Histogram.

If the Histogram was fitted with the Option "0", the Fit Function is globally disabled for drawing. This function enables it for the active drawing and disables it again afterwards. Note: The Draw is done via DrawCopy ! Author: Ronnie Weber

Parameters:

- ***option*** Draw Options for Histogram
- ***fname*** Name of the function

Definition at line 379 of file RegH1F.C.

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

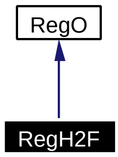
- **RegH1F.h**
- **RegH1F.C**

6.78 RegH2F Class Reference

`#include <RegH2F.h>`

Inheritance diagram for RegH2F:

Collaboration diagram for RegH2F:



6.78.1 Detailed Description

A registered 2-dimensional TH2F histogram.

Author: Jenny Böhme, Benno List

Date

2005/06/10 09:22:11

Author

blist

Changelog:

- 9.6.04 BL: Use HListPoR, documentation ugraded, added constructor with Binning objects

Definition at line 30 of file RegH2F.h.

Public Member Functions

- **RegH2F** (const char ∗name, const char ∗title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl=0)

  Constructor with equidistant x and y binnings.

- **RegH2F** (const char ∗name, const char ∗title, Int_t nbinsx, const Double_t ∗xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant x and equidistant y binnings.

- **RegH2F** (const char ∗name, const char ∗title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t ∗ybins, const **ROListPoR** &hhl=0)

  Constructor with equidistant x and non-equidistant y binnings.

- **RegH2F** (const char ∗name, const char ∗title, Int_t nbinsx, const Double_t ∗xbins, Int_t nbinsy, const Double_t ∗ybins, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant x and y binnings taking doubles.

- **RegH2F** (const char ∗name, const char ∗title, Int_t nbinsx, const Float_t ∗xbins, Int_t nbinsy, const Float_t ∗ybins, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant x and y binnings taking floats.

- **RegH2F** (const char ∗name, const char ∗title, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl=0)

  Constructor taking 2 Binning objects.

Private Member Functions

- **RegH2F** (const **RegH2F** &source)

  Private copy constructor, left unimplemented: No copying!

- **RegH2F** & **operator=** (const **RegH2F** &source)

  Private assignment operator, left unimplemented: No assignment!

6.78.2 Constructor & Destructor Documentation

6.78.2.1 RegH2F::RegH2F (const char ∗ *name*, const char ∗ *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl* = 0)

Constructor with equidistant x and y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

Definition at line 32 of file RegH2F.C.

6.78.2.2 RegH2F::RegH2F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant x and equidistant y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

Definition at line 43 of file RegH2F.C.

6.78.2.3 RegH2F::RegH2F (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl* = 0)

Constructor with equidistant x and non-equidistant y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

Definition at line 54 of file RegH2F.C.

6.78.2.4 RegH2F::RegH2F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant x and y binnings taking doubles.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

Definition at line 65 of file RegH2F.C.

6.78.2.5 RegH2F::RegH2F (const char ∗ *name*, const char ∗ *title*, Int_t *nbinsx*, const Float_t ∗ *xbins*, Int_t *nbinsy*, const Float_t ∗ *ybins*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant x and y binnings taking floats.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

Definition at line 75 of file RegH2F.C.

6.78.2.6 RegH2F::RegH2F (const char ∗ *name*, const char ∗ *title*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl* = 0)

Constructor taking 2 Binning objects.

Parameters:

name TH2F name

title TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

Definition at line 85 of file RegH2F.C.

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

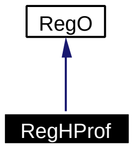
- **RegH2F.h**
- **RegH2F.C**

6.79 RegHProf Class Reference

`#include <RegHProf.h>`

Inheritance diagram for RegHProf:

Collaboration diagram for RegHProf:



6.79.1 Detailed Description

A registered 1-dimensional TProfile histogram.

Author: Jenny Böhme, Benno List

Date
2004/06/15 10:54:52

Author
blist

Changelog:

- 9.6.04 BL: Use HListPoR, documentation ugraded, added constructor with Binning objects

Definition at line 30 of file RegHProf.h.

Public Member Functions

- **RegHProf** (const char *name, const char *title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with equidistant binning.

- **RegHProf** (const char *name, const char *title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow, Axis_t yup, Option_t *option, const **ROListPoR** &hhl=0)

Constructor with equidistant binning and limited y range.

- **RegHProf** (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant binning, taking float bin edges.

- **RegHProf** (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant binning, taking double bin edges.

- **RegHProf** (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Axis_t ylow, Axis_t yup, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with non-equidistant binning and limited y range.

- **RegHProf** (const char *name, const char *title, const Binning &binning, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with Binning object.

- **RegHProf** (const char *name, const char *title, const Binning &binning, Axis_t ylow, Axis_t yup, Option_t *option, const **ROListPoR** &hhl=0)

  Constructor with Binning object and limited y range.

Private Member Functions

- void **initBinning** (const Binning &binning)

  helper function to set binning right

- **RegHProf** (const **RegHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **RegHProf** & **operator=** (const **RegHProf** &source)

  Private assignment operator, left unimplemented: No assignment!

6.79.2 Constructor & Destructor Documentation

6.79.2.1 RegHProf::RegHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with equidistant binning.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 32 of file RegHProf.C.

6.79.2.2 RegHProf::RegHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow*, Axis_t *yup*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with equidistant binning and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 42 of file RegHProf.C.

6.79.2.3 RegHProf::RegHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Float_t * *xbins*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant binning, taking float bin edges.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 53 of file RegHProf.C.

6.79.2.4 RegHProf::RegHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant binning, taking double bin edges.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 62 of file RegHProf.C.

6.79.2.5 RegHProf::RegHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Axis_t *ylow*, Axis_t *yup*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with non-equidistant binning and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 71 of file RegHProf.C.

6.79.2.6 RegHProf::RegHProf (const char * *name*, const char * *title*, const Binning & *binning*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with Binning object.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

binning Defines the histogram binning

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 81 of file RegHProf.C.

References initBinning().

6.79.2.7 RegHProf::RegHProf (const char * *name*, const char * *title*, const Binning & *binning*, Axis_t *ylow*, Axis_t *yup*, Option_t * *option*, const ROListPoR & *hhl* = 0)

Constructor with Binning object and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

binning Defines the histogram binning

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

Definition at line 93 of file RegHProf.C.

References initBinning().

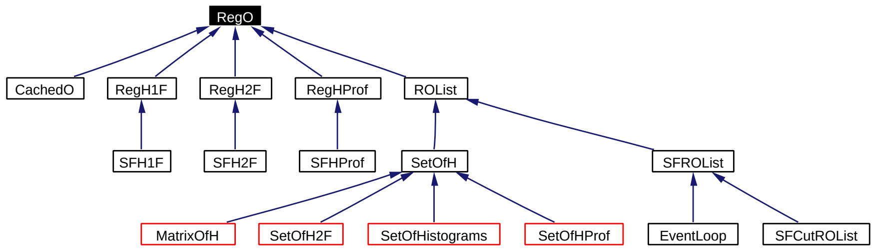
The documentation for this class was generated from the following files:

- **RegHProf.h**
- **RegHProf.C**

6.80 RegO Class Reference

`#include <RegO.h>`

Inheritance diagram for RegO:



6.80.1 Detailed Description

Mix-in class for registered objects.

A RegO object can register itself in a **ROList** object, which holds an array of pointers to RegO objects.

Author: Benno List, Jenny Böhme

Date

2004/06/30 15:31:46

Author

blist

Changelog:

- 10.6.04 BL First version (adapted from RegH)

Definition at line 27 of file RegO.h.

Public Member Functions

- **RegO** ()

  Default constructor.

- **RegO** (const **ROListPoR** &rol)

  Constructor that registers this object in an **ROList** *object.*

- virtual ~**RegO** ()

  Virtual destructor to ensure that dynamic casts work.

Private Member Functions

- **RegO** (const **RegO** &rhs)

  Private copy constructor, left unimplemented: No copying!

- **RegO** & **operator=** (const **RegO** &rhs)

  Private assignment operator, left unimplemented: No assignment!

6.80.2 Constructor & Destructor Documentation

6.80.2.1 RegO::RegO (const ROListPoR & *rol*)

Constructor that registers this object in an **ROList** object.

Parameters:

rol The **ROList** object where this is registered

Definition at line 27 of file RegO.C.

References ROListPoR::prol, and ROList::registerObject().

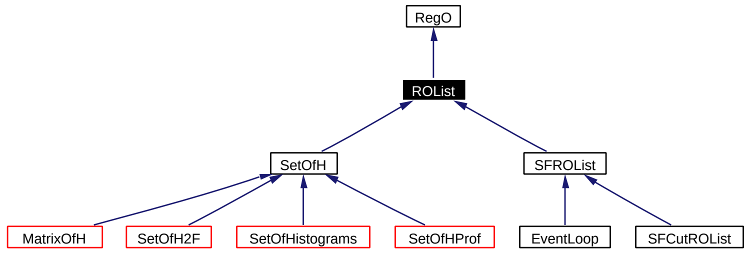
The documentation for this class was generated from the following files:

- **RegO.h**
- **RegO.C**

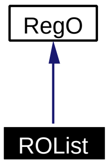
6.81 ROList Class Reference

`#include <ROList.h>`

Inheritance diagram for ROList:



Collaboration diagram for ROList:



6.81.1 Detailed Description

Stores a List of registered Histos (**RegO**), also recursively.

An ROList stores pointers to **RegO** objects, and to other ROList objects. Its main purpose is to allow operations on all of the stored objects.

One possibility to operate on all objects is to use an **HVisitor** object, which will be called with all stored objects in turn (see Gamma, Helm, Johnson, Vlissides: Design Patterns for a discussion of the Visitor pattern). Special methods include **Write()**, which writes out all histograms to a file, and Draw(...), which draws histograms on a canvas.

A ROList is itself a **RegO**, which means that we can build trees of such lists (see the "Compound" pattern in Gamma, Helm, Johnson, Vlissides: Design Patterns).

To iterate over all entries, do something like this:

```
void iterate_over_ROList (const ROList& rol) {
  for (ROList::const_iterator iter = rol.begin(); iter != rol.end(); ++iter) {
    RegO *ro = rol.getEntry (iter);
    // do something with ro here...
  }
*
```

An older way to iterate is this:

```
void iterate_over_ROList (const ROList& rol) {
  ROList::IndexType i = rol.getFirstEntry();
  RegO *ro = rol.getEntry (i);
```

```
  while (ro) {
    // do something with ro here...
    ro = rol.getNextEntry (i);   // updates i and ro!
  }
*
```

Author: Jenny Böhme, Benno List

Date

2006/05/15 13:28:18

Author

zimmerto

Changelog:

- 10.6.04 BL First version (adapted from HList)
- 14.8.04 BL, JB: Use map to store object references, allow nonconsecutive numbers
- 2.5.05 BL: Added **getFirstEntry()** and getNextEntry(i)
- 27.5.05 BL: Added begin(), end(), rbegin(), rend(), made iterator types publicly available
- 27.5.05 BL: Added "recursive" flag to Draw(...)
- 10.7.05 BL: Added DrawCutFlow

Definition at line 76 of file ROList.h.

Public Types

- typedef unsigned long long **IndexType**

  The type that is used to enumerate the objects internally.

- typedef MapType::const_iterator **const_iterator**
- typedef MapType::iterator **iterator**
- typedef MapType::const_reverse_iterator **const_reverse_iterator**
- typedef MapType::reverse_iterator **reverse_iterator**

Public Member Functions

- **ROList** ()

  Default constructor: creates an empty ROList.

- **ROList** (const **ROListPoR** &mother)

  Creates an empty ROList and registers it with another list.

- int **registerObject** (**RegO** ∗h)

  Registers a **RegO** *object.*

- int **registerObject** (**IndexType** n, **RegO** ∗h)

  Registers a **RegO** *object under a given number.*

- virtual void **Write** ()

  Writes out all histograms (if they are derived from TH1).

- virtual **ROList** & **visit** (**HVisitor** &v)

  Visits all histograms.

- virtual **ROList** & **visit** (const **ConstHVisitor** &v)

  Visits all histograms.

- virtual int **Draw** (TPad *pad, TVirtualPS *ps=0, Option_t *option=0, bool across=true, Int_t nx=1, Int_t ny=1, Float_t xmargin=0.01, Float_t ymargin=0.01, Int_t color=0, Int_t log=0, bool recursive=false, int izone=0)

  Draws all histograms (if they are derived from TH1).

- virtual int **Draw** (TPad *pad, TVirtualPS *ps, Option_t *option, bool across, Int_t nx, Int_t ny, Int_t log, bool recursive=false, int izone=0)
- virtual int **DrawFunc** (TPad *pad, TVirtualPS *ps=0, Option_t *option=0, const char *fname="", bool across=true, Int_t nx=1, Int_t ny=1, Float_t xmargin=0.01, Float_t ymargin=0.01, Int_t color=0, Int_t log=0, bool recursive=false, int izone=0)

  Enable a specific Fit Function to be plotted with the Histogram.

- virtual int **DrawFunc** (TPad *pad, TVirtualPS *ps, Option_t *option, const char *fname, bool across, Int_t nx, Int_t ny, Int_t log, bool recursive=false, int izone=0)
- virtual void **DrawCutFlow** (TPad *pad, TVirtualPS *ps=0, Option_t *option=0, Option_t *sameoption=0, bool across=true, Int_t nx=1, Int_t ny=1, Float_t xmargin=0.01, Float_t ymargin=0.01, Int_t color=0, Int_t log=0, bool recursive=false, bool backward=false)

  Draws all histograms (if they are derived from TH1), with the following histogram overlaid.

- virtual void **DrawCutFlow** (TPad *pad, TVirtualPS *ps, Option_t *option, Option_t *sameoption, bool across, Int_t nx, Int_t ny, Int_t log, bool recursive=false, bool backward=false)

  Draws all histograms (if they are derived from TH1).

- **IndexType getEntries** () const

  Return the number of entries in the list.

- **IndexType getHighestEntry** () const

  Returns the highest number plus 1 of any entry in the map.

- **IndexType getFirstEntry** () const

  Returns the lowest number of any entry in the map.

- **RegO** * **getEntry** (**IndexType** i) const

  Returns pointer to entry with number i, 0 if entry does not exist.

- **RegO** * **getNextEntry** (**IndexType** &i) const

  Returns pointer to next entry with number greater than i, 0 if entry does not exist.

- **RegO** * **getEntry** (const_iterator iter) const

Returns pointer to entry with number i, 0 if entry does not exist.

- **RegO** * **getNextEntry** (const_iterator &i) const

  Returns pointer to next entry with number greater than i, 0 if entry does not exist.

- iterator **begin** ()
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- iterator **lower_bound** (**IndexType** i)
- const_iterator **lower_bound** (**IndexType** i) const
- iterator **upper_bound** (**IndexType** i)
- const_iterator **upper_bound** (**IndexType** i) const
- **IndexType getIndex** (iterator iter) const

  Get index that corresponds to an iterator.

- **IndexType getIndex** (const_iterator iter) const

Private Types

- typedef std::map< **IndexType**, **RegO** * > **MapType**
- enum { **LINEAR** = 0, **LOGY** = 1, **LOGX** = 2, **LOGXY** = 3 }

Private Attributes

- MapType **theMap**

  the map of pointers to registered histos

6.81.2 Constructor & Destructor Documentation

6.81.2.1 ROList::ROList (const ROListPoR & *mother*)

Creates an empty ROList and registers it with another list.

Parameters:
  mother The "mother" list

Definition at line 48 of file ROList.C.

References ROListPoR::prol, and registerObject().

6.81.3 Member Function Documentation

6.81.3.1 int ROList::registerObject (RegO * *h*)

Registers a **RegO** object.

Parameters:

h The object to be registered

Definition at line 57 of file ROList.C.

References getEntries(), getHighestEntry(), and theMap.

Referenced by SetOfHProf::initHisto(), SetOfHistograms::initHisto(), SetOfH2F::initHisto(), MatrixOfHProf::initHisto(), MatrixOfHistograms::initHisto(), MatrixOfH2F::initHisto(), MatrixOfHistograms::MatrixOfHistograms(), RegO::RegO(), ROList(), and SetOfH::SetOfH().

6.81.3.2 int ROList::registerObject (IndexType *n*, RegO * *h*)

Registers a **RegO** object under a given number.

Parameters:

n The number under which to register the object

h The object to be registered

Definition at line 69 of file ROList.C.

References getEntries(), and theMap.

6.81.3.3 ROList & ROList::visit (HVisitor & *v*) `[virtual]`

Visits all histograms.

Calls v(RegO& RegO) for each of the registered histograms See Gamma, Helm, Johnson, Vlissides: "Design Patterns" for a discussion of the Visitor pattern.

Parameters:

v The visitor object

Definition at line 92 of file ROList.C.

References HVisitor::visit().

Referenced by visit(), and Write().

6.81.3.4 int ROList::Draw (TPad * *pad*, TVirtualPS * *ps* = 0, Option_t * *option* = 0, bool *across* = true, Int_t *nx* = 1, Int_t *ny* = 1, Float_t *xmargin* = 0.01, Float_t *ymargin* = 0.01, Int_t *color* = 0, Int_t *log* = 0, bool *recursive* = false, int *izone* = 0) `[virtual]`

Draws all histograms (if they are derived from TH1).

- pad is usually a TCanvas object and MUST be provided,

- ps is usually a TPostscript object
- option is an option string like "HE0"
- across determines whether the histograms are arranged left-to-right (across=true) or top-to-bottom on the page
- nx, ny are the number of histos in horizontal (nx) and vertical (ny) direction on a page
- xmargin, ymargin are the corresponding values used by TPad::Divide.
- color is passed to TPad::Divide.
- log determines whether the histos shall be plotted logarithmically.

Parameters:

pad usually a TCanvas object and MUST be provided
ps usually a TPostscript object
option an option string like "HE0"
across arrange histos left-to-right (true) or top-to-bottom
nx passed to TPad::Divide.
ny passed to TPad::Divide.
xmargin passed to TPad::Divide.
ymargin passed to TPad::Divide.
color passed to TPad::Divide.
log determines whether the histos shall be plotted logarithmically
recursive determines whether histos in Sets and Matrices shall also be plotted
izone start at this pad

Definition at line 120 of file ROList.C.

References getEntries(), and IndexType.

Referenced by Draw().

6.81.3.5 int ROList::Draw (TPad ∗ *pad*, TVirtualPS ∗ *ps*, Option_t ∗ *option*, bool *across*, Int_t *nx*, Int_t *ny*, Int_t *log*, bool *recursive* = false, int *izone* = 0) `[virtual]`

Draws all histograms (if they are derived from TH1) start at zone izone

Parameters:

pad usually a TCanvas object and MUST be provided
ps usually a TPostscript object
option an option string like "HE0"
across arrange histos left-to-right (true) or top-to-bottom
nx passed to TPad::Divide.
ny passed to TPad::Divide.
log determines whether the histos shall be plotted logarithmically
recursive determines whether histos in Sets and Matrices shall also be plotted
izone start at this pad

Definition at line 172 of file ROList.C.

References Draw().

6.81.3.6 int ROList::DrawFunc (TPad ∗ *pad*, TVirtualPS ∗ *ps* = 0, Option_t ∗ *option* = 0, const char ∗ *fname* = "", bool *across* = true, Int_t *nx* = 1, Int_t *ny* = 1, Float_t *xmargin* = 0.01, Float_t *ymargin* = 0.01, Int_t *color* = 0, Int_t *log* = 0, bool *recursive* = false, int *izone* = 0) `[virtual]`

Enable a specific Fit Function to be plotted with the Histogram.

If the Histogram was fitted with the Option "0", the Fit Function is globally disabled for drawing. This function enables it for the active drawing and disables it again afterwards. Note: The Draw is done via DrawCopy ! Inputs:

- pad is usually a TCanvas object and MUST be provided,
- ps is usually a TPostscript object
- option is an option string like "HE0"
- fname is the name of the function to be plotted
- across determines whether the histograms are arranged left-to-right (across=true) or top-to-bottom on the page
- nx, ny are the number of histos in horizontal (nx) and vertical (ny) direction on a page
- xmargin, ymargin are the corresponding values used by TPad::Divide.
- color is passed to TPad::Divide.
- log determines whether the histos shall be plotted logarithmically. Author: Ronnie Weber

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

fname Name of the function

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

xmargin passed to TPad::Divide.

ymargin passed to TPad::Divide.

color passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

recursive determines whether histos in Sets and Matrices shall also be plotted

izone start at this pad

Definition at line 275 of file ROList.C.

References getEntries(), and IndexType.

Referenced by DrawFunc().

6.81.3.7 int ROList::DrawFunc (TPad * *pad*, TVirtualPS * *ps*, Option_t * *option*, const char * *fname*, bool *across*, Int_t *nx*, Int_t *ny*, Int_t *log*, bool *recursive* = false, int *izone* = 0) [virtual]

Draws all histograms (if they are derived from TH1) and the function fname start at zone izone

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

fname Name of the function

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

recursive determines whether histos in Sets and Matrices shall also be plotted

izone start at this pad

Definition at line 341 of file ROList.C.

References DrawFunc().

6.81.3.8 void ROList::DrawCutFlow (TPad * *pad*, TVirtualPS * *ps* = 0, Option_t * *option* = 0, Option_t * *sameoption* = 0, bool *across* = true, Int_t *nx* = 1, Int_t *ny* = 1, Float_t *xmargin* = 0.01, Float_t *ymargin* = 0.01, Int_t *color* = 0, Int_t *log* = 0, bool *recursive* = false, bool *backward* = false) [virtual]

Draws all histograms (if they are derived from TH1), with the following histogram overlaid.

- pad is usually a TCanvas object and MUST be provided,
- ps is usually a TPostscript object
- option is an option string like "HE0"
- across determines whether the histograms are arranged left-to-right (across=true) or top-to-bottom on the page
- nx, ny are the number of histos in horizontal (nx) and vertical (ny) direction on a page
- xmargin, ymargin are the corresponding values used by TPad::Divide.
- color is passed to TPad::Divide.
- log determines whether the histos shall be plotted logarithmically.

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

sameoption an option string like "HE0 SAME"

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

xmargin passed to TPad::Divide.

ymargin passed to TPad::Divide.

color passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

recursive determines whether histos in Sets and Matrices shall also be plotted

backward determines whether the cuts are compared "forward" or "backward"

Definition at line 180 of file ROList.C.

References getEntries(), and IndexType.

Referenced by DrawCutFlow().

6.81.3.9 void ROList::DrawCutFlow (TPad * *pad*, TVirtualPS * *ps*, Option_t * *option*, Option_t * *sameoption*, bool *across*, Int_t *nx*, Int_t *ny*, Int_t *log*, bool *recursive* = false, bool *backward* = false) `[virtual]`

Draws all histograms (if they are derived from TH1).

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

sameoption an option string like "HE0 SAME"

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

recursive determines whether histos in Sets and Matrices shall also be plotted

backward determines whether the cuts are compared "forward" or "backward"

Definition at line 265 of file ROList.C.

References DrawCutFlow().

6.81.3.10 RegO * ROList::getEntry (IndexType *i*) const `[inline]`

Returns pointer to entry with number i, 0 if entry does not exist.

Parameters:

i The entry number

Definition at line 314 of file ROList.h.

References theMap.

Referenced by SetOfH::Add(), SFEffSetOfHistograms::Divide(), SetOfH::Divide(), SFEffSetOfHistograms::Draw(), SFSetOfHProf::Fill(), SFSetOfHistograms::Fill(), SFSetOfH2F::Fill(), SFROList::Fill(), SFMatrixOfHProf::Fill(), SFMatrixOfHistograms::Fill(), SFMatrixOfH2F::Fill(), SFEffSetOfHistograms::Fill(), SetOfHProf::getHisto(), SetOfHistograms::getHisto(), SetOfH2F::getHisto(), SetOfH::getHisto(), MatrixOfHProf::getHisto(), MatrixOfHistograms::getHisto(), MatrixOfH2F::getHisto(), MatrixOfH::getHisto(), EventLoop::loop(), and SetOfH::Multiply().

6.81.3.11 RegO ∗ ROList::getNextEntry (IndexType & *i*) const [inline]

Returns pointer to next entry with number greater than i, 0 if entry does not exist.

Parameters:

i The entry number (will be increased!)

Definition at line 320 of file ROList.h.

References theMap.

6.81.3.12 RegO ∗ ROList::getEntry (const_iterator *iter*) const [inline]

Returns pointer to entry with number i, 0 if entry does not exist.

Parameters:

iter An Iterator

Definition at line 329 of file ROList.h.

6.81.3.13 RegO ∗ ROList::getNextEntry (const_iterator & *i*) const [inline]

Returns pointer to next entry with number greater than i, 0 if entry does not exist.

Parameters:

iter An iterator (will be increased)

Definition at line 333 of file ROList.h.

References theMap.

6.81.3.14 ROList::IndexType ROList::getIndex (iterator *iter*) const [inline]

Get index that corresponds to an iterator.

Parameters:

iter An Iterator

Definition at line 380 of file ROList.h.

References getHighestEntry(), IndexType, and theMap.

Referenced by SetOfH::Add(), SetOfH::Divide(), SetOfHistograms::getSummary(), and SetOfH::Multiply().

6.81.3.15 ROList::IndexType ROList::getIndex (const_iterator *iter*) const `[inline]`

Parameters:

iter An Iterator

Definition at line 384 of file ROList.h.

References getHighestEntry(), IndexType, and theMap.

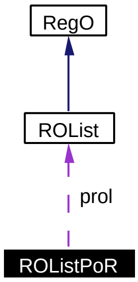
The documentation for this class was generated from the following files:

- **ROList.h**
- **ROList.C**

6.82 ROListPoR Class Reference

`#include <ROListPoR.h>`

Collaboration diagram for ROListPoR:



6.82.1 Detailed Description

ROList Pointer or Reference.

Automagically converts between pointers and references to **ROList** objects

Author: Benno List, Jenny Boehme

Date
2004/06/29 13:03:38

by:

Author
blist

Changelog:

- 10.6.06 BL First version (adapted from HListPoR)

Definition at line 30 of file ROListPoR.h.

Public Member Functions

- **ROListPoR** (**ROList** &rol_)
  Constructor from a reference.
- **ROListPoR** (**ROList** *prol_=0)
  Constructor from a pointer.

Public Attributes

- **ROList** * **prol**
  The pointer to the **ROList**.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 ROListPoR::ROListPoR (ROList & *rol_*) [inline]

Constructor from a reference.

Parameters:
*rol_* A reference to a **ROList**

Definition at line 33 of file ROListPoR.h.

References prol.

6.82.2.2 ROListPoR::ROListPoR (ROList * *prol_* = 0) [inline]

Constructor from a pointer.

Parameters:
*prol_* A pointer to a **ROList**

Definition at line 37 of file ROListPoR.h.

References prol.

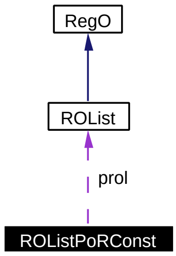
The documentation for this class was generated from the following file:

- **ROListPoR.h**

6.83 ROListPoRConst Class Reference

`#include <ROListPoR.h>`

Collaboration diagram for ROListPoRConst:



6.83.1 Detailed Description

Constant **ROList** Pointer or Reference.

Automagically converts between pointers and references to const **ROList** objects

Author: Benno List, Jenny Boehme

Date

2004/06/29 13:03:38

by:

Author

blist

Definition at line 55 of file ROListPoR.h.

Public Member Functions

- **ROListPoRConst** (const **ROList** &rol_)

  Constructor from a reference.

- **ROListPoRConst** (const **ROList** *prol_=0)

  Constructor from a pointer.

Public Attributes

- const **ROList** * **prol**

  The pointer to the const **ROList**.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 ROListPoRConst::ROListPoRConst (const ROList & *rol_*) [inline]

Constructor from a reference.

Parameters:

*rol_* A reference to a const **ROList**

Definition at line 58 of file ROListPoR.h.

References prol.

6.83.2.2 ROListPoRConst::ROListPoRConst (const ROList * *prol_* = 0) [inline]

Constructor from a pointer.

Parameters:

*prol_* A pointer to a const **ROList**

Definition at line 62 of file ROListPoR.h.

References prol.

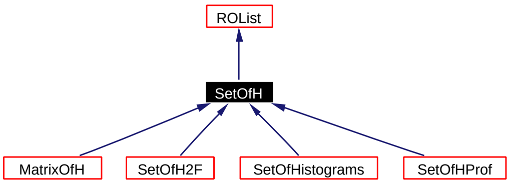
The documentation for this class was generated from the following file:

- **ROListPoR.h**

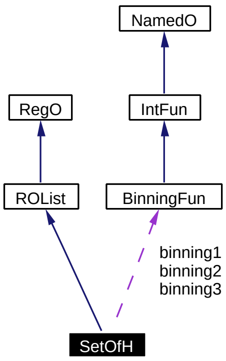
6.84 SetOfH Class Reference

`#include <SetOfH.h>`

Inheritance diagram for SetOfH:



Collaboration diagram for SetOfH:



6.84.1 Detailed Description

Base class for collection of histograms that have the same axis/axes.

A SetOfH contains a set of histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SetOfH is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Benno List

Date

2005/11/26 13:51:02

Author

blist

Changelog:

- 21.5.05 BL: First version (from **SetOfHProf**)
- 27.5.05 BL: Make data members private, remove getNBins()
- 27.5.05 BL: Reinsert Divide method with overlap correction
- 15.6.05 BL: Change signatures of Divide, Multiply to be compatible to TH1 signatures
- 26.11.05 BL: Make initHisto public

Definition at line 53 of file SetOfH.h.

Public Member Functions

- **SetOfH** (const char ∗name_, const char ∗title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, Option_t ∗option_=0)

  Constructor for a 1-dimensional Set.

- **SetOfH** (const char ∗name_, const char ∗title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, Option_t ∗option_=0)

  Constructor for a 2-dimensional Set (a Matrix).

- virtual ∼**SetOfH** ()

  Destructor.

- TH1 ∗ **getHisto** (unsigned int i)

  Returns histogram number i (corresponding to bin i of the **BinningFun***).*

- virtual void **Divide** (TF1 ∗f1, Double_t c1=1)

  Divide all histos by the same function (as for TProfile): this = this/(c1∗f1).

- virtual void **Divide** (const TH1 ∗h1)

  Divide all histos by the same histogram (as for TProfile): this = this/h1.

- virtual void **Divide** (const TH1 ∗h1, const TH1 ∗h2, Double_t c1=1, Double_t c2=1, Option_t ∗**option**=0)

  Replace contents of all histograms by the division of h1 by h2: this = c1∗h1/(c2∗h2).

- virtual void **Divide** (const **SetOfH** ∗soh)

  Divide each histo by corresponding histo of soh: this = this/soh.

- virtual void **Divide** (const **SetOfH** ∗soh1, const **SetOfH** ∗soh2, Double_t c1=1, Double_t c2=1, Option_t ∗**option**=0)

  Replace each histo by ratio of corresponding histos in soh1 and soh2: this = c1∗soh1/(c2∗soh2).

- virtual void **Divide** (const **SetOfH** ∗soh1, const **SetOfH** ∗soh2, const **SetOfH** ∗soh3, Double_t c1=1, Double_t c2=1)

  Replace each histo by the division of h1 by h2, with h3 as overlap histogram: this = c1∗h1/(c2∗h2).

- virtual void **Multiply** (TF1 ∗f1, Double_t c1=1)

  Multiply all histos by the same function (as for TProfile): this = this∗(c1∗f1).

- virtual void **Multiply** (const TH1 ∗h1)

  Multiply all histos by the same function (as for TProfile): this = this∗(c1∗f1).

- virtual void **Multiply** (const TH1 ∗h1, const TH1 ∗h2, Double_t c1=1, Double_t c2=1, Option_t ∗**option**=0)

  Replace contents of all histograms by the product of h1 by h2: this = c1∗h1/(c2∗h2).

- virtual void **Multiply** (const **SetOfH** ∗soh)

  Multiply each histo by corresponding histo of soh: this = this∗soh.

- virtual void **Multiply** (const **SetOfH** ∗soh1, const **SetOfH** ∗soh2, Double_t c1=1, Double_t c2=1, Option_t ∗**option**=0)

  Multiply each histo by product of corresponding histos in soh1 and soh2: this = (c1∗soh1)∗(c2∗soh2).

- virtual void **Add** (const **SetOfH** ∗soh, Double_t c1=1)

  Add to each histo corresponding histo of soh: this = this+c1∗soh.

- virtual const char ∗ **genBinTitle** (**IndexType** i) const

  Generate a bin title string; the returned string must be deleted by the caller!

- virtual const char ∗ **genBinName** (**IndexType** i) const

  Generate a bin name string; the returned string must be deleted by the caller!

- const char ∗ **getTitle** () const

  Get the title string. Don't delete the returned string!

- const char ∗ **getName** () const

  Get the name string. Don't delete the returned string!

- const char ∗ **getOption** () const

  Get the option string. Don't delete the returned string!

- const **BinningFun** ∗ **getBinning** () const

  Get the **BinningFun**.

- **BinningFun** ∗ **getBinning** ()

  Get the **BinningFun**.

- const **BinningFun** ∗ **getBinning** (int i) const

  Get the **BinningFun** *of a certain dimension (1, 2 [for* **MatrixOfH**], *3 [for CubeOfH]).*

- **BinningFun** ∗ **getBinning** (int i)

  Get the **BinningFun** *of a certain dimension (1, 2 [for* **MatrixOfH**], *3 [for CubeOfH]).*

- virtual void **initHisto** (**IndexType** i)=0

  Books and registers one histogram with number i.

Protected Member Functions

- virtual void **initHistos** ()

  Book all the histos , generate their names and titles.

Private Member Functions

- void **init** (const char *name_, const char *title_, Option_t *option_)

  Copy name and title to local store.

- **SetOfH** (const **SetOfH** &source)

  Private copy constructor, left unimplemented: No copying!

- **SetOfH** & **operator=** (const **SetOfH** &source)

  Private assignment operator, left unimplemented: No assignment!

Private Attributes

- char * **name**

  Common histogram name.

- char * **title**

  Common histogram title.

- **BinningFun** * **binning1**

  The 1st **BinningFun** *that assigns entries to histograms.*

- **BinningFun** * **binning2**

  The 2nd **BinningFun** *that assigns entries to histograms.*

- **BinningFun** * **binning3**

  The 3rd **BinningFun** *that assigns entries to histograms.*

- char * **option**

  Option string (as for TProfile).

6.84.2 Constructor & Destructor Documentation

6.84.2.1 SetOfH::SetOfH (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, Option_t * *option_* = 0)

Constructor for a 1-dimensional Set.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning1_ The **BinningFun** that assigns entries to histograms

option_ An optional option.

Definition at line 37 of file SetOfH.C.

References init(), ROListPoR::prol, and ROList::registerObject().

6.84.2.2 SetOfH::SetOfH (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, Option_t * *option_* = 0)

Constructor for a 2-dimensional Set (a Matrix).

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

option_ An optional option.

Definition at line 49 of file SetOfH.C.

References init(), ROListPoR::prol, and ROList::registerObject().

6.84.3 Member Function Documentation

6.84.3.1 TH1 * SetOfH::getHisto (unsigned int *i*)

Returns histogram number i (corresponding to bin i of the **BinningFun**).

Parameters:

i The number of the histogram to be returned

Definition at line 89 of file SetOfH.C.

References ROList::getEntries(), and ROList::getEntry().

6.84.3.2 void SetOfH::Divide (TF1 * *f1*, Double_t *c1* = 1) `[virtual]`

Divide all histos by the same function (as for TProfile): this = this/(c1*f1).

Parameters:

f1 The function f1

c1 An optional multiplier to f1

Definition at line 134 of file SetOfH.C.

References ROList::getEntry().

6.84.3.3 void SetOfH::Divide (const TH1 * *h1*) [virtual]

Divide all histos by the same histogram (as for TProfile): this = this/h1.

Parameters:

h1 The histogram h1

Definition at line 140 of file SetOfH.C.

References ROList::getEntry().

6.84.3.4 void SetOfH::Divide (const TH1 * *h1*, const TH1 * *h2*, Double_t *c1* = 1, Double_t *c2* = 1, Option_t * *option* = 0) [virtual]

Replace contents of all histograms by the division of h1 by h2: this = c1*h1/(c2*h2).

This operation is somewhat pointless for a **SetOfHProf**, it exists only to have the same Divide methods as for a TProfile

Parameters:

h1 The first histogram h1

h2 The second histogram h2

c1 An optional multiplier to h1

c2 An optional multiplier to h2

option An option. "B" means binomial errors

Definition at line 146 of file SetOfH.C.

References ROList::getEntry().

6.84.3.5 void SetOfH::Divide (const SetOfH * *soh*) [virtual]

Divide each histo by corresponding histo of soh: this = this/soh.

Parameters:

soh Histograms by which to divide

Definition at line 152 of file SetOfH.C.

References ROList::getEntries(), ROList::getEntry(), ROList::getIndex(), and getName().

6.84.3.6 void SetOfH::Divide (const SetOfH * *soh1*, const SetOfH * *soh2*, Double_t *c1* = 1, Double_t *c2* = 1, Option_t * *option* = 0) [virtual]

Replace each histo by ratio of corresponding histos in soh1 and soh2: this = c1*soh1/(c2*soh2).

Parameters:

soh1 The first set of histos soh1

soh2 The second set of histos moh2

c1 An optional multiplier to soh1

c2 An optional multiplier to soh2

option An option. "B" means binomial errors

Definition at line 168 of file SetOfH.C.

References ROList::getEntries(), ROList::getEntry(), ROList::getIndex(), and getName().

6.84.3.7 void SetOfH::Divide (const SetOfH ∗ *soh1*, const SetOfH ∗ *soh2*, const SetOfH ∗ *soh3*, Double_t *c1* = 1, Double_t *c2* = 1) `[virtual]`

Replace each histo by the division of h1 by h2, with h3 as overlap histogram: this = c1∗h1/(c2∗h2).

In the most general case where soh1 contains a sum of event weights w_i and soh2 a sum of event weights v_i, h3 should contain the sum w_i∗v_i for those events which fall into the same x bin of the same histogram; soh3 is the overlap set of histograms.

Parameters:

soh1 The first set of histos soh1

soh2 The second set of histos moh2

soh3 The third set of histos moh2

c1 An optional multiplier to soh1

c2 An optional multiplier to soh2

Definition at line 187 of file SetOfH.C.

References RegH1F::Divide(), ROList::getEntries(), ROList::getEntry(), ROList::getIndex(), and getName().

6.84.3.8 void SetOfH::Multiply (TF1 ∗ *f1*, Double_t *c1* = 1) `[virtual]`

Multiply all histos by the same function (as for TProfile): this = this∗(c1∗f1).

Parameters:

f1 The function f1

c1 An optional multiplier to f1

Definition at line 209 of file SetOfH.C.

References ROList::getEntry().

6.84.3.9 void SetOfH::Multiply (const TH1 ∗ *h1*) `[virtual]`

Multiply all histos by the same function (as for TProfile): this = this∗(c1∗f1).

An optional multiplier to f1

Parameters:

h1 The function f1

Definition at line 215 of file SetOfH.C.

References ROList::getEntry().

6.84.3.10 void SetOfH::Multiply (const TH1 * *h1*, const TH1 * *h2*, Double_t *c1* = 1, Double_t *c2* = 1, Option_t * *option* = 0) `[virtual]`

Replace contents of all histograms by the product of h1 by h2: this = c1*h1/(c2*h2).

This operation is somewhat pointless for a SetOfH, it exists only to have the same Multiply methods as for a TProfile

Parameters:

- ***h1*** The first matrix of histos moh1
- ***h2*** The firsecondst matrix of histos moh2
- ***c1*** An optional multiplier to moh1
- ***c2*** An optional multiplier to moh2
- ***option*** An option that is not used, as for TH1

Definition at line 221 of file SetOfH.C.

References ROList::getEntry().

6.84.3.11 void SetOfH::Multiply (const SetOfH * *soh*) `[virtual]`

Multiply each histo by corresponding histo of soh: this = this*soh.

Parameters:

- ***soh*** Histograms by which to multiply

Definition at line 227 of file SetOfH.C.

References ROList::getEntries(), ROList::getEntry(), ROList::getIndex(), and getName().

6.84.3.12 void SetOfH::Multiply (const SetOfH * *soh1*, const SetOfH * *soh2*, Double_t *c1* = 1, Double_t *c2* = 1, Option_t * *option* = 0) `[virtual]`

Multiply each histo by product of corresponding histos in soh1 and soh2: this = (c1*soh1)*(c2*soh2).

Parameters:

- ***soh1*** The first set of histos soh1
- ***soh2*** The second set of histos soh2
- ***c1*** An optional multiplier to soh1
- ***c2*** An optional multiplier to soh2
- ***option*** An option that is not used, as for TH1

Definition at line 244 of file SetOfH.C.

References ROList::getEntries(), ROList::getEntry(), ROList::getIndex(), and getName().

6.84.3.13 void SetOfH::Add (const SetOfH * *soh*, Double_t *c1* = 1) `[virtual]`

Add to each histo corresponding histo of soh: this = this+c1*soh.

Parameters:

soh Histograms to add

c1 An optional multiplier to soh

Definition at line 264 of file SetOfH.C.

References ROList::getEntries(), ROList::getEntry(), and ROList::getIndex().

6.84.3.14 const char * SetOfH::genBinTitle (IndexType *i*) const `[virtual]`

Generate a bin title string; the returned string must be deleted by the caller!

Parameters:

i The bin number

Definition at line 94 of file SetOfH.C.

References binning1, BinningFun::getBinTitle(), and title.

Referenced by MatrixOfHistograms::getSet(), SetOfHProf::initHisto(), SetOfHistograms::initHisto(), and SetOfH2F::initHisto().

6.84.3.15 const char * SetOfH::genBinName (IndexType *i*) const `[virtual]`

Generate a bin name string; the returned string must be deleted by the caller!

Parameters:

i The bin number

Definition at line 109 of file SetOfH.C.

References binning1, BinningFun::getBinName(), and name.

Referenced by MatrixOfHistograms::getSet(), SetOfHProf::initHisto(), SetOfHistograms::initHisto(), SetOfH2F::initHisto(), and SetOfHistograms::SetOfHistograms().

6.84.3.16 const BinningFun * SetOfH::getBinning (int *i*) const `[inline]`

Get the **BinningFun** of a certain dimension (1, 2 [for **MatrixOfH**], 3 [for CubeOfH]).

Parameters:

i The dimension (1 or 2)

Definition at line 234 of file SetOfH.h.

References binning1, binning2, and binning3.

6.84.3.17 BinningFun * SetOfH::getBinning (int *i*) `[inline]`

Get the **BinningFun** of a certain dimension (1, 2 [for **MatrixOfH**], 3 [for CubeOfH]).

Parameters:

i The dimension (1 or 2)

Definition at line 243 of file SetOfH.h.

References binning1, binning2, and binning3.

6.84.3.18 virtual void SetOfH::initHisto (IndexType *i*) `[pure virtual]`

Books and registers one histogram with number i.

Parameters:

i The number under which to register

Implemented in **MatrixOfH** (p. 186), **SetOfH2F** (p. 327), **SetOfHistograms** (p. 335), and **SetOfHProf** (p. 343).

Referenced by initHistos().

6.84.3.19 void SetOfH::init (const char * *name_*, const char * *title_*, Option_t * *option_*) `[private]`

Copy name and title to local store.

Parameters:

name_ Common histogram name

title_ Common histogram title

option_ Option string (as for TProfile)

Definition at line 62 of file SetOfH.C.

References name, option, and title.

Referenced by SetOfH().

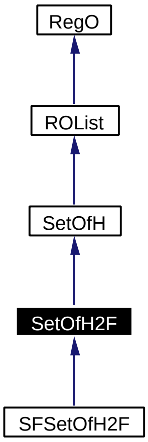
The documentation for this class was generated from the following files:

- **SetOfH.h**
- **SetOfH.C**

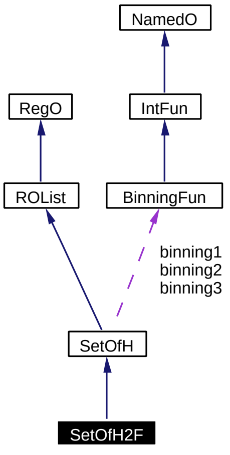
6.85 SetOfH2F Class Reference

`#include <SetOfH2F.h>`

Inheritance diagram for SetOfH2F:



Collaboration diagram for SetOfH2F:



6.85.1 Detailed Description

A collection of profile histograms that have the same axis.

A SetOfH2F contains a set of TH2F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SetOfH2F is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Benno List

Date

2005/11/26 13:51:02

Author

blist

Changelog:

- 10.6.5.05 BL: First version (from **SetOfHProf**)

Definition at line 49 of file SetOfH2F.h.

Public Member Functions

- **SetOfH2F** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for an empty set histograms.

- **SetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with equidistant x and y binnings.

- **SetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

- **SetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

- **SetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

- **SetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

- **SetOfH2F** (const char *name_, const char *title_, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms, taking 2 Binning objects.

- virtual ~**SetOfH2F** ()

  Destructor.

- **RegH2F** ∗ **getHisto** (**IndexType** i)

  Returns histogram number i (corresponding to bin i of the **BinningFun***).*

- void **initHisto** (**IndexType** i)

  Books and registers one histogram with number i.

Protected Member Functions

- void **initialize** (**ROList** &hhl)

Protected Attributes

- Binning **xaxisbinning**

  The x axis binning.

- Binning **yaxisbinning**

  The y axis binning.

Private Member Functions

- **SetOfH2F** (const **SetOfH2F** &source)

  Private copy constructor, left unimplemented: No copying!

- **SetOfH2F** & **operator=** (const **SetOfH2F** &source)

  Private assignment operator, left unimplemented: No assignment!

6.85.2 Constructor & Destructor Documentation

6.85.2.1 SetOfH2F::SetOfH2F (const char ∗ *name_*, const char ∗ *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for an empty set histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 28 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.2 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with equidistant x and y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 37 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.3 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 54 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.4 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 70 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.5 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 86 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.6 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Int_t *nbinsy*, const Float_t * *ybins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH1F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in x (as for TH1F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 101 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.2.7 SetOfH2F::SetOfH2F (const char * *name_*, const char * *title_*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms, taking 2 Binning objects.

Parameters:

name_ TH2F name

title_ TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 116 of file SetOfH2F.C.

References SetOfH::initHistos().

6.85.3 Member Function Documentation

6.85.3.1 RegH2F * SetOfH2F::getHisto (IndexType *i*)

Returns histogram number i (corresponding to bin i of the **BinningFun**).

Parameters:

i The number of the histogram to be returned

Definition at line 142 of file SetOfH2F.C.

References ROList::getEntry().

6.85.3.2 void SetOfH2F::initHisto (IndexType *i*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i The number under which to register

Implements **SetOfH** (p. 321).

Definition at line 130 of file SetOfH2F.C.

References SetOfH::genBinName(), SetOfH::genBinTitle(), ROList::registerObject(), xaxisbinning, and yaxisbinning.

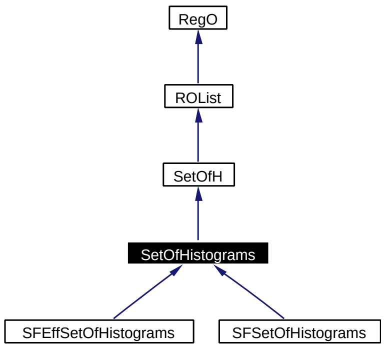
The documentation for this class was generated from the following files:

- **SetOfH2F.h**
- **SetOfH2F.C**

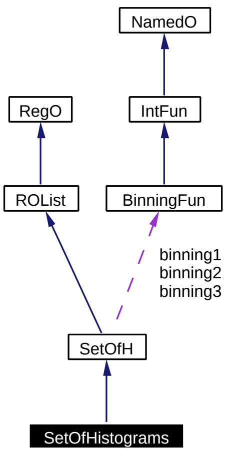
6.86 SetOfHistograms Class Reference

`#include <SetOfHistograms.h>`

Inheritance diagram for SetOfHistograms:



Collaboration diagram for SetOfHistograms:



6.86.1 Detailed Description

A collection of 1-dimensional histograms that have the same axis.

A SetOfHistograms contains a set of 1-dimensional TH1F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SetOfHistograms is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date
2006/06/30 11:05:00

Author
blist

Changelog:

- 9.6.04 BL: Upgrade documentation, use HListPoR and **BinningFunPoR**, introduce constructors using Binning objects
- 22.6.04 BL: New private method initHistos, more documentation
- 14.8.04 BL, JB: Add axisbinning, initHisto
- 12.5.05 BL, JB: Add new Divide method with overlap set of histograms
- 26.5.05 BL: Derived from **SetOfH** instead of **ROList**
- 26.11.05 BL: Make initHisto public

Definition at line 58 of file SetOfHistograms.h.

Public Member Functions

- **SetOfHistograms** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for an empty set histograms.

- **SetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with equidistant axis binning.

- **SetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SetOfHistograms** (const char *name_, const char *title_, const Binning &axisbinning_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor taking a Binning object.

- **SetOfHistograms** (const char *name_, const char *title_, TFile &file, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms to be read from a file.

- virtual ~**SetOfHistograms** ()

  Destructor.

- **RegH1F** ∗ **getHisto** (**IndexType** i)

  Returns histogram number i (corresponding to bin i of the **BinningFun***).*

- **RegH1F** ∗ **getHisto** (iterator i)

  Returns histogram corresponding to an iterator.

- const **RegH1F** ∗ **getHisto** (const_iterator i) const

  Returns histogram corresponding to an iterator.

- **RegH1F** ∗ **getSummary** (Option_t ∗option_, const **ROListPoR** &hhl)

  Returns a summary histogram,.

- **RegH1F** ∗ **getSummary** (Option_t ∗option_, const char ∗fname, int parnumber, int hnumber, const char ∗tcomment, const **ROListPoR** &hhl)

  Returns a summary histogram from fit function parameters.

- virtual **RegH1F** ∗ **getSum** (const char ∗**name**, const char ∗**title**, const **ROListPoR** &hhl, int lowbin=0, int highbin=0) const

  Returns a histogram that is the sum of all histos in the set.

- virtual void **initHisto** (**IndexType** i)

  Books and registers one histogram with number i.

Protected Member Functions

- void **initialize** (**ROList** &hhl)

Protected Attributes

- Binning **axisbinning**

  The axis binning.

Private Member Functions

- **SetOfHistograms** (const **SetOfHistograms** &source)

  Private copy constructor, left unimplemented: No copying!

- **SetOfHistograms** & **operator=** (const **SetOfHistograms** &source)

  Private assignment operator, left unimplemented: No assignment!

6.86.2 Constructor & Destructor Documentation

6.86.2.1 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for an empty set histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 35 of file SetOfHistograms.C.

6.86.2.2 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 43 of file SetOfHistograms.C.

References SetOfH::initHistos().

6.86.2.3 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 57 of file SetOfHistograms.C.

References SetOfH::initHistos().

6.86.2.4 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 70 of file SetOfHistograms.C.

References SetOfH::initHistos().

6.86.2.5 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 82 of file SetOfHistograms.C.

References SetOfH::initHistos().

6.86.2.6 SetOfHistograms::SetOfHistograms (const char * *name_*, const char * *title_*, TFile & *file*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms to be read from a file.

Parameters:

name_ Common histogram name

title_ Common histogram title

file An input root file

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 94 of file SetOfHistograms.C.

References SetOfH::genBinName(), SetOfH::getBinning(), and ROList::IndexType.

6.86.3 Member Function Documentation

6.86.3.1 RegH1F * SetOfHistograms::getHisto (IndexType *i*)

Returns histogram number i (corresponding to bin i of the **BinningFun**).

Parameters:
i The number of the histogram to be returned

Definition at line 122 of file SetOfHistograms.C.

References ROList::getEntry().

Referenced by getSum(), getSummary(), and MatrixOfHistograms::getSummarySet().

6.86.3.2 RegH1F * SetOfHistograms::getHisto (iterator *i*)

Returns histogram corresponding to an iterator.

Parameters:
i The iterator

Definition at line 126 of file SetOfHistograms.C.

References ROList::getEntry().

6.86.3.3 const RegH1F * SetOfHistograms::getHisto (const_iterator *i*) const

Returns histogram corresponding to an iterator.

Parameters:
i The iterator

Definition at line 130 of file SetOfHistograms.C.

References ROList::getEntry().

6.86.3.4 RegH1F * SetOfHistograms::getSummary (Option_t * *option_*, const ROListPoR & *hhl*)

Returns a summary histogram,.

The histogram contains on the x-axis a bin number and has as contents the contents or the mean of the individula histograms

Parameters:
*option_* Option string Possible values:

- 1: 1-dim summary histo (default)
- 2: 2-dim histo
- C: Contents (default) => number 9999
- M: Mean => number 9998

hhl List where to register the summary histo

Definition at line 139 of file SetOfHistograms.C.

References SetOfH::getBinning(), RegH1F::getHistInfo(), getHisto(), ROList::getIndex(), SetOfH::getName(), and SetOfH::getTitle().

6.86.3.5 RegH1F * SetOfHistograms::getSummary (Option_t * *option_*, const char * *fname*, int *parnumber*, int *hnumber*, const char * *tcomment*, const ROListPoR & *hhl*)

Returns a summary histogram from fit function parameters.

The histogram contains on the x-axis a bin number and has as contents the value of a fit function parameter

Parameters:

option_ Option string (at present unused)

fname The function name

parnumber The parameter number to extract

hnumber A nuber to add to the id

tcomment A comment to add to the title

hhl List where to register the summary histo

Definition at line 185 of file SetOfHistograms.C.

References SetOfH::getBinning(), getHisto(), ROList::getIndex(), SetOfH::getName(), and SetOfH::getTitle().

6.86.3.6 RegH1F * SetOfHistograms::getSum (const char * *name*, const char * *title*, const ROListPoR & *hhl*, int *lowbin* = 0, int *highbin* = 0) const `[virtual]`

Returns a histogram that is the sum of all histos in the set.

Parameters:

name Name for the summary histo

title Title for the summary histo

hhl List where to register the summary set

lowbin Optional number of lowest bin to sum

highbin Optional number of highest bin to sum

Definition at line 227 of file SetOfHistograms.C.

References ROList::getHighestEntry(), getHisto(), and ROList::IndexType.

6.86.3.7 void SetOfHistograms::initHisto (IndexType *i*) `[virtual]`

Books and registers one histogram with number i.

Parameters:

i The number under which to register

Implements **SetOfH** (p. 321).

Definition at line 111 of file SetOfHistograms.C.

References axisbinning, SetOfH::genBinName(), SetOfH::genBinTitle(), and ROList::registerObject().

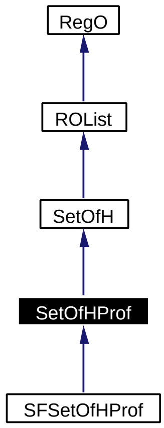
The documentation for this class was generated from the following files:

- **SetOfHistograms.h**
- **SetOfHistograms.C**

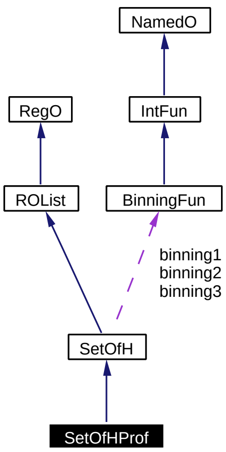
6.87 SetOfHProf Class Reference

`#include <SetOfHProf.h>`

Inheritance diagram for SetOfHProf:



Collaboration diagram for SetOfHProf:



6.87.1 Detailed Description

A collection of profile histograms that have the same axis.

A SetOfHProf contains a set of THProfile histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SetOfHProf is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Benno List

Date
2005/11/26 13:51:02

Author
blist

Changelog:

- 13.5.05 BL: First version (from **SetOfHistograms**)
- 26.5.05 BL: Derived from **SetOfH** instead of **ROList**
- 26.11.05 BL: Make initHisto public

Definition at line 51 of file SetOfHProf.h.

Public Member Functions

- **SetOfHProf** (const char ∗name_, const char ∗title_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for an empty set histograms.

- **SetOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with equidistant binning.

- **SetOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow_, Axis_t yup_, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with equidistant binning and limited y range.

- **SetOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Float_t ∗xbins, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant binning, taking float bin edges.

- **SetOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Double_t ∗xbins, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant binning, taking double bin edges.

- **SetOfHProf** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Double_t ∗xbins, Axis_t ylow_, Axis_t yup_, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with non-equidistant binning and limited y range.

- **SetOfHProf** (const char ∗name_, const char ∗title_, const Binning &axisbinning_, Option_t ∗option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with Binning object.

- **SetOfHProf** (const char *name_, const char *title_, const Binning &axisbinning_, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &hhl, const **BinningFunPoR** &binning_)

  Constructor for a set of histograms with Binning object and limited y range.

- virtual ~**SetOfHProf** ()

  Destructor.

- **RegHProf** * **getHisto** (**IndexType** i)

  Returns histogram number i (corresponding to bin i of the **BinningFun***).*

- void **initHisto** (**IndexType** i)

  Books and registers one histogram with number i.

Protected Member Functions

- void **initialize** (**ROList** &hhl)

Protected Attributes

- Binning **axisbinning**

  The axis binning.

- Axis_t **ylow**

  Minimum y value (as for TProfile).

- Axis_t **yup**

  Maximum y value (as for TProfile).

Private Member Functions

- **SetOfHProf** (const **SetOfHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **SetOfHProf** & **operator=** (const **SetOfHProf** &source)

  Private assignment operator, left unimplemented: No assignment!

6.87.2 Constructor & Destructor Documentation

6.87.2.1 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for an empty set histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 31 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.2 SetOfHProf::SetOfHProf (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t ∗ *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with equidistant binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 42 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.3 SetOfHProf::SetOfHProf (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow_*, Axis_t *yup_*, Option_t ∗ *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with equidistant binning and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 57 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.4 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Option_t * *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant binning, taking float bin edges.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 74 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.5 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant binning, taking double bin edges.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 88 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.6 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with non-equidistant binning and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 102 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.7 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Option_t * *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 118 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.2.8 SetOfHProf::SetOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *hhl*, const BinningFunPoR & *binning_*)

Constructor for a set of histograms with Binning object and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

binning_ The **BinningFun** that assigns entries to histograms

Definition at line 131 of file SetOfHProf.C.

References SetOfH::initHistos().

6.87.3 Member Function Documentation

6.87.3.1 RegHProf ∗ SetOfHProf::getHisto (IndexType *i*)

Returns histogram number i (corresponding to bin i of the **BinningFun**).

Parameters:
i The number of the histogram to be returned

Definition at line 160 of file SetOfHProf.C.

References ROList::getEntry().

6.87.3.2 void SetOfHProf::initHisto (IndexType *i*) `[virtual]`

Books and registers one histogram with number i.

Parameters:
i The number under which to register

Implements **SetOfH** (p. 321).

Definition at line 148 of file SetOfHProf.C.

References axisbinning, SetOfH::genBinName(), SetOfH::genBinTitle(), SetOfH::getOption(), ROList::registerObject(), ylow, and yup.

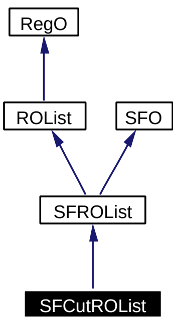
The documentation for this class was generated from the following files:

- **SetOfHProf.h**
- **SetOfHProf.C**

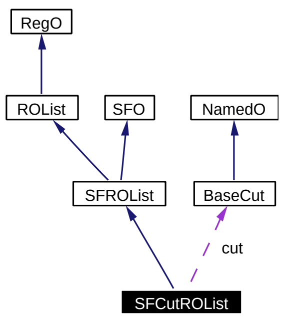
6.88 SFCutROList Class Reference

`#include <SFCutROList.h>`

Inheritance diagram for SFCutROList:



Collaboration diagram for SFCutROList:



6.88.1 Detailed Description

A list of Self-Filling objects with a cut.

Author: Benno List, Jenny Böhme

Date
2005/07/18 07:04:25

Author
blist

Changelog:

- 18.7.05 BL: First version, adapted from class **SFROList**

Definition at line 34 of file SFCutROList.h.

Public Member Functions

- **SFCutROList** ()
  Default constructor: creates an empty SFCutROList.
- **SFCutROList** (const **ROListPoR** &mother)
  Creates an empty SFCutROList and registers it with another list.
- virtual ~**SFCutROList** ()
  Destructor.
- virtual void **Fill** ()
  Fills all self-filling objects.
- **BaseCut** ∗ **getCut** ()
  Returns the cut object.
- const **BaseCut** ∗ **getCut** () const
  Returns the cut object as constant.
- **SFCutROList** & **setCut** (const **BaseCutPoR** &newcut)
  Sets the cut object.

Private Attributes

- **BaseCut** ∗ **cut**
  The cut object.

6.88.2 Constructor & Destructor Documentation

6.88.2.1 SFCutROList::SFCutROList (const ROListPoR & *mother*)

Creates an empty SFCutROList and registers it with another list.

Parameters:
  mother The "mother" list

Definition at line 31 of file SFCutROList.C.

6.88.3 Member Function Documentation

6.88.3.1 SFCutROList & SFCutROList::setCut (const BaseCutPoR & *newcut*)

Sets the cut object.

Parameters:
  newcut The new cut object

Definition at line 51 of file SFCutROList.C.

References cut, BaseCut::getIterator(), NamedO::getName(), and BaseCutPoR::pbc.

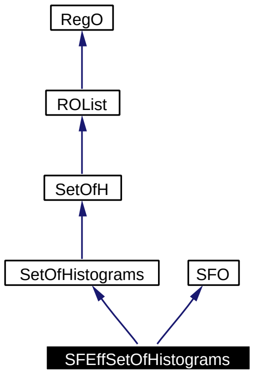
The documentation for this class was generated from the following files:

- **SFCutROList.h**
- **SFCutROList.C**

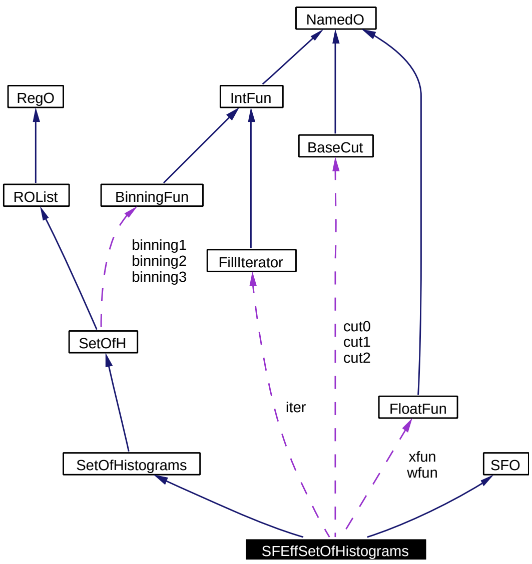
6.89 SFEffSetOfHistograms Class Reference

`#include <SFEffSetOfHistograms.h>`

Inheritance diagram for SFEffSetOfHistograms:



Collaboration diagram for SFEffSetOfHistograms:



6.89.1 Detailed Description

A self-filling collection of 1-D histos with the same axis for efficiency calculation.

Like a **SetOfHistograms**, a SFEffSetOfHistograms contains a set of 1-dimensional TH1F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created automatically.

Three histos are booked, one using **BaseCut** object cut1, one using cut2, the third will be an efficiency derived by dividing histo2 by histo1.

Author: Benno List, Jenny Böhme

Date
2004/07/15 00:07:02

Author
blist

Changelog:

- 14.6.04 BL: First version

Definition at line 39 of file SFEffSetOfHistograms.h.

Public Member Functions

- **SFEffSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut0_, const **BaseCutPoR** &cut1_, const **BaseCutPoR** &cut2_, const **FloatFunPoR** &wfun_, const **FillIteratorPoR** &iter_=0, Option_t *div_option_=0)

  Constructor for a set of histograms with equidistant axis binning.

- **SFEffSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut0_, const **BaseCutPoR** &cut1_, const **BaseCutPoR** &cut2_, const **FloatFunPoR** &wfun_, const **FillIteratorPoR** &iter_=0, Option_t *div_option_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SFEffSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut0_, const **BaseCutPoR** &cut1_, const **BaseCutPoR** &cut2_, const **FloatFunPoR** &wfun_, const **FillIteratorPoR** &iter_=0, Option_t *div_option_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SFEffSetOfHistograms** (const char *name_, const char *title_, const Binning &**axisbinning**, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut0_, const **BaseCutPoR** &cut1_, const **BaseCutPoR** &cut2_, const **FloatFunPoR** &wfun_, const **FillIteratorPoR** &iter_=0, Option_t *div_option_=0)

  Constructor taking a Binning object.

- virtual ~**SFEffSetOfHistograms** ()

  Virtual destructor.

- virtual void **Draw** (TPad *pad, TVirtualPS *ps=0, Option_t ***option**=0, bool across=true, Int_t nx=1, Int_t ny=1, Float_t xmargin=0.01, Float_t ymargin=0.01, Int_t color=0, bool log=false)

  Draws all histograms (if they are derived from TH1).

- virtual void **Draw** (TPad *pad, TVirtualPS *ps, Option_t ***option**, bool across, Int_t nx, Int_t ny, bool log)

  Draws all histograms (if they are derived from TH1).

- virtual void **Draw** (Option_t ***option**=0)

Draws the efficiency histo onto an existing pad.

- virtual void **Fill** ()

  The Self-filler method.

- virtual void **Divide** ()

  Divides histo 2 by histo 1 to get an efficiency histo 3.

Protected Member Functions

- void **checkIterators** (const char ∗**name**)

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot).

- **BaseCut** ∗ **cut0**

  optional Function object defining a cut for histo 1 (0 means: no cut)

- **BaseCut** ∗ **cut1**

  optional Function object defining a cut for histo 1 (0 means: no cut)

- **BaseCut** ∗ **cut2**

  optional Function object defining a cut for histo 1 (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

- char ∗ **div_option**

  An option. "B" means binomial errors.

Private Member Functions

- **SFEffSetOfHistograms** (const **SFEffSetOfHistograms** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFEffSetOfHistograms** & **operator=** (const **SFEffSetOfHistograms** &source)

  Private assignment operator, left unimplemented: No assignment!

- void **copy_option** (Option_t ∗div_option_)

  Copies option string.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 SFEffSetOfHistograms::SFEffSetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut0_*, const BaseCutPoR & *cut1_*, const BaseCutPoR & *cut2_*, const FloatFunPoR & *wfun_*, const FillIteratorPoR & *iter_* = 0, Option_t * *div_option_* = 0)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut0_ optional Function object defining a cut for both histos (0 means: no cut)

cut1_ optional Function object defining a cut for histo 1 (0 means: no cut)

cut2_ optional Function object defining a cut for histo 2 (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

div_option_ An option. "B" means binomial errors

Definition at line 69 of file SFEffSetOfHistograms.C.

References copy_option().

6.89.2.2 SFEffSetOfHistograms::SFEffSetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut0_*, const BaseCutPoR & *cut1_*, const BaseCutPoR & *cut2_*, const FloatFunPoR & *wfun_*, const FillIteratorPoR & *iter_* = 0, Option_t * *div_option_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut0_ optional Function object defining a cut for both histos (0 means: no cut)

cut1_ optional Function object defining a cut for histo 1 (0 means: no cut)

cut2_ optional Function object defining a cut for histo 2 (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

div_option_ An option. "B" means binomial errors

Definition at line 90 of file SFEffSetOfHistograms.C.

References copy_option().

6.89.2.3 SFEffSetOfHistograms::SFEffSetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut0_*, const BaseCutPoR & *cut1_*, const BaseCutPoR & *cut2_*, const FloatFunPoR & *wfun_*, const FillIteratorPoR & *iter_* = 0, Option_t * *div_option_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut0_ optional Function object defining a cut for both histos (0 means: no cut)

cut1_ optional Function object defining a cut for histo 1 (0 means: no cut)

cut2_ optional Function object defining a cut for histo 2 (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

div_option_ An option. "B" means binomial errors

Definition at line 110 of file SFEffSetOfHistograms.C.

References copy_option().

6.89.2.4 SFEffSetOfHistograms::SFEffSetOfHistograms (const char * *name_*, const char * *title_*, const Binning & *axisbinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut0_*, const BaseCutPoR & *cut1_*, const BaseCutPoR & *cut2_*, const FloatFunPoR & *wfun_*, const FillIteratorPoR & *iter_* = 0, Option_t * *div_option_* = 0)

Constructor taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning Defines the histogram binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut0_ optional Function object defining a cut for both histos (0 means: no cut)

cut1_ optional Function object defining a cut for histo 1 (0 means: no cut)

cut2_ optional Function object defining a cut for histo 2 (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

div_option_ An option. "B" means binomial errors

Definition at line 130 of file SFEffSetOfHistograms.C.

References copy_option().

6.89.3 Member Function Documentation

6.89.3.1 void SFEffSetOfHistograms::Draw (TPad * *pad*, TVirtualPS * *ps* = 0, Option_t * *option* = 0, bool *across* = true, Int_t *nx* = 1, Int_t *ny* = 1, Float_t *xmargin* = 0.01, Float_t *ymargin* = 0.01, Int_t *color* = 0, bool *log* = false) [virtual]

Draws all histograms (if they are derived from TH1).

- pad is usually a TCanvas object and MUST be provided,
- ps is usually a TPostscript object
- option is an option string like "HE0"
- across determines whether the histograms are arranged left-to-right (across=true) or top-to-bottom on the page
- nx, ny are the number of histos in horizontal (nx) and vertical (ny) direction on a page
- xmargin, ymargin are the corresponding values used by TPad::Divide.
- color is passed to TPad::Divide.
- log determines whether the histos shall be plotted logarithmically.

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

xmargin passed to TPad::Divide.

ymargin passed to TPad::Divide.

color passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

Definition at line 177 of file SFEffSetOfHistograms.C.

Referenced by Draw().

6.89.3.2 void SFEffSetOfHistograms::Draw (TPad ∗ *pad*, TVirtualPS ∗ *ps*, Option_t ∗ *option*, bool *across*, Int_t *nx*, Int_t *ny*, bool *log*) [virtual]

Draws all histograms (if they are derived from TH1).

Parameters:

pad usually a TCanvas object and MUST be provided

ps usually a TPostscript object

option an option string like "HE0"

across arrange histos left-to-right (true) or top-to-bottom

nx passed to TPad::Divide.

ny passed to TPad::Divide.

log determines whether the histos shall be plotted logarithmically

Definition at line 196 of file SFEffSetOfHistograms.C.

References Draw().

6.89.3.3 void SFEffSetOfHistograms::Fill () [virtual]

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 154 of file SFEffSetOfHistograms.C.

References cut0, cut1, cut2, ROList::getEntry(), iter, FillIterator::next(), FillIterator::reset(), wfun, and xfun.

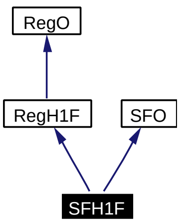
The documentation for this class was generated from the following files:

- **SFEffSetOfHistograms.h**
- **SFEffSetOfHistograms.C**

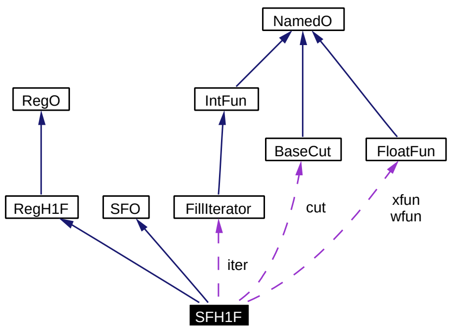
6.90 SFH1F Class Reference

`#include <SFH1F.h>`

Inheritance diagram for SFH1F:



Collaboration diagram for SFH1F:



6.90.1 Detailed Description

A self-filling 1-dimensional TH1F histogram.

A self-filling SFH1F object is a TH1F that registers itself in a **ROList** histogram list when it is created. It has a **Fill()** method that fills one or several entries into the histogram. It uses **FloatFun** and **BaseCut** objects to know what to fill, and possibly **FillIterator** objects in the case that several entries are to be made in one **Fill()** call.

Author: Benno List, Jenny Böhme

Date

2005/06/07 17:03:24

Author

blist

Changelog:

- 4.6.04 BL: Add new constructor using a Binning object
- 8.6.04 BL: Use HListPoR
- 10.6.04 BL: Derived from class **SFO** instead of SFH

- 15.7.04 BL: Added method **checkIterators()**
- 15.7.04 BL: Added methods getXFun etc

Definition at line 47 of file SFH1F.h.

Public Member Functions

- **SFH1F** (const char ∗name, const char ∗title, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with equidistant binning.

- **SFH1F** (const char ∗name, const char ∗title, Int_t nbinsx, const Float_t ∗xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** iter_=0)

  Constructor with non-equidistant binning, taking float bin edges.

- **SFH1F** (const char ∗name, const char ∗title, Int_t nbinsx, const Double_t ∗xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, **FillIteratorPoR** iter_=0)

  Constructor with non-equidistant binning, taking double bin edges.

- **SFH1F** (const char ∗name, const char ∗title, const Binning &binning, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, **FillIteratorPoR** iter_=0)

  Constructor taking a Binning object.

- **SFH1F** (const char ∗name, const char ∗title, const TH1F &source, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, **FillIteratorPoR** iter_=0)

  "steals" binning from a TH1F -> contents are IGNORED

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char *name)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char *name, const char *fun1name, Fun1 *fun1, const char *fun2name, Fun2 *fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char *name, const char *funname, Fun *fun)

  Function object.

Protected Attributes

- **FloatFun** * **xfun**

  Function object defining abscissa (what to plot).

- **BaseCut** * **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** * **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** * **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFH1F** (const **SFH1F** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFH1F** & **operator=** (const **SFH1F** &source)

  Private assignment operator, left unimplemented: No assignment!

6.90.2 Constructor & Destructor Documentation

6.90.2.1 SFH1F::SFH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with equidistant binning.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 36 of file SFH1F.C.

References checkIterators().

6.90.2.2 SFH1F::SFH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Float_t * *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR *iter_* = 0)

Constructor with non-equidistant binning, taking float bin edges.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 51 of file SFH1F.C.

References checkIterators().

6.90.2.3 SFH1F::SFH1F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, FillIteratorPoR *iter_* = 0)

Constructor with non-equidistant binning, taking double bin edges.

Parameters:

name TH1F name

title TH1F title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 65 of file SFH1F.C.

References checkIterators().

6.90.2.4 SFH1F::SFH1F (const char ∗ *name*, const char ∗ *title*, const Binning & *binning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, FillIteratorPoR *iter_* = 0)

Constructor taking a Binning object.

Parameters:

name TH1F name

title TH1F title

binning Defines the histogram binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 79 of file SFH1F.C.

References checkIterators().

6.90.2.5 SFH1F::SFH1F (const char ∗ *name*, const char ∗ *title*, const TH1F & *source*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, FillIteratorPoR *iter_* = 0)

"steals" binning from a TH1F -> contents are IGNORED

Parameters:

name New TH1F name

title New TH1F title

source Histogram to be copied

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 94 of file SFH1F.C.

6.90.3 Member Function Documentation

6.90.3.1 void SFH1F::Fill () [virtual]

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 113 of file SFH1F.C.

References cut, iter, FillIterator::next(), FillIterator::reset(), wfun, and xfun.

6.90.3.2 void SFH1F::checkIterators (const char * *name*) [protected]

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 173 of file SFH1F.C.

References checkOneIterator(), checkTwoIterators(), cut, wfun, and xfun.

Referenced by SFH1F().

6.90.3.3 template<class Fun1, class Fun2> void SFH1F::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) [protected]

Checks consistency of iterators of two function objects.

Function object 2

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

Definition at line 126 of file SFH1F.C.

Referenced by checkIterators().

6.90.3.4 template<class Fun> void SFH1F::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) [protected]

Function object.

Parameters:

name The histogram name

funname Name of function object

Definition at line 143 of file SFH1F.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

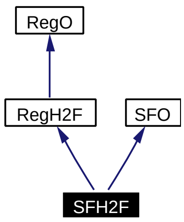
The documentation for this class was generated from the following files:

- **SFH1F.h**
- **SFH1F.C**

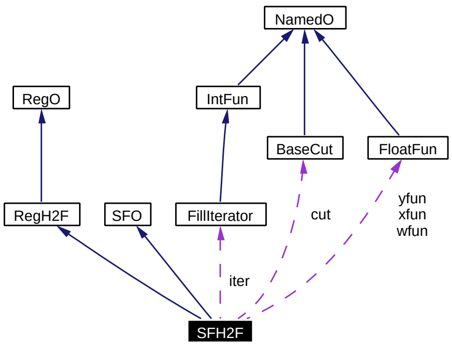
6.91 SFH2F Class Reference

`#include <SFH2F.h>`

Inheritance diagram for SFH2F:



Collaboration diagram for SFH2F:



6.91.1 Detailed Description

A self-filling 2-dimensional TH2F histogram.

A self-filling SFH2F object is a TH2F that registers itself in a **ROList** histogram list when it is created. It has a Fill(S) method that fills one or several entries into the histogram. It uses **FloatFun** and **BaseCut** objects to know what to fill, and possibly **FillIterator** objects in the case that several entries are to be made in one **Fill()** call.

Author: Jenny Böhme, Benno List

Date

2005/06/10 09:22:11

Author

blist

Changelog:

- 9.6.04 BL: Use **ROListPoR**, documentation ugraded, use const references to FitFunPoR, **BaseCutPoR**, and **FillIteratorPoR**, added constructor with Binning objects
- 10.6.04 BL: Derived from class **SFO** instead of SFH

- 15.7.04 BL: Added method **checkIterators()**
- 21.7.04 BL: Introduced checkTwoIterators, checkOneIterator
- 15.7.04 BL: Added methods getXFun etc

Definition at line 49 of file SFH2F.h.

Public Member Functions

- **SFH2F** (const char *name, const char *title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with equidistant x and y binnings.

- **SFH2F** (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant x and equidistant y binnings.

- **SFH2F** (const char *name, const char *title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with equidistant x and non-equidistant y binnings.

- **SFH2F** (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant x and y binnings taking doubles.

- **SFH2F** (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant x and y binnings taking floats.

- **SFH2F** (const char *name, const char *title, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor taking 2 Binning objects.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** * **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** ∗ **getYFun** () const

  Returns a pointer to the ordinate function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗name)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗name, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗name, const char ∗funname, Fun ∗fun)

  Function object.

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFH2F** (const **SFH2F** &source)
  Private copy constructor, left unimplemented: No copying!
- **SFH2F** & **operator=** (const **SFH2F** &source)
  Private assignment operator, left unimplemented: No assignment!

6.91.2 Constructor & Destructor Documentation

6.91.2.1 SFH2F::SFH2F (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with equidistant x and y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 36 of file SFH2F.C.

References checkIterators().

6.91.2.2 SFH2F::SFH2F (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant x and equidistant y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 52 of file SFH2F.C.

References checkIterators().

6.91.2.3 SFH2F::SFH2F (const char ∗ *name*, const char ∗ *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t ∗ *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with equidistant x and non-equidistant y binnings.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 68 of file SFH2F.C.

References checkIterators().

6.91.2.4 SFH2F::SFH2F (const char ∗ *name*, const char ∗ *title*, Int_t *nbinsx*, const Double_t ∗ *xbins*, Int_t *nbinsy*, const Double_t ∗ *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant x and y binnings taking doubles.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 84 of file SFH2F.C.

References checkIterators().

6.91.2.5 SFH2F::SFH2F (const char ∗ *name*, const char ∗ *title*, Int_t *nbinsx*, const Float_t ∗ *xbins*, Int_t *nbinsy*, const Float_t ∗ *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant x and y binnings taking floats.

Parameters:

name TH2F name

title TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 99 of file SFH2F.C.

References checkIterators().

6.91.2.6 SFH2F::SFH2F (const char * *name*, const char * *title*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor taking 2 Binning objects.

Parameters:

name TH2F name

title TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 114 of file SFH2F.C.

References checkIterators().

6.91.3 Member Function Documentation

6.91.3.1 void SFH2F::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the xfun object to get the abscissa, yfun to obtain the ordinate, and the wfun object to obtain the weight.

Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 133 of file SFH2F.C.

References cut, iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.91.3.2 void SFH2F::checkIterators (const char * *name*) `[protected]`

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 190 of file SFH2F.C.

References checkOneIterator(), checkTwoIterators(), cut, wfun, xfun, and yfun.

Referenced by SFH2F().

6.91.3.3 template<class Fun1, class Fun2> void SFH2F::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) `[protected]`

Checks consistency of iterators of two function objects.

Function object 2

Parameters:

- ***name*** The histogram name
- ***fun1name*** Name of function object 1
- ***fun1*** Function object 1
- ***fun2name*** Name of function object 2

Definition at line 146 of file SFH2F.C.

Referenced by checkIterators().

6.91.3.4 template<class Fun> void SFH2F::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) `[protected]`

Function object.

Parameters:

- ***name*** The histogram name
- ***funname*** Name of function object

Definition at line 160 of file SFH2F.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

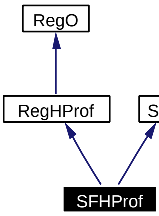
The documentation for this class was generated from the following files:

- **SFH2F.h**
- **SFH2F.C**

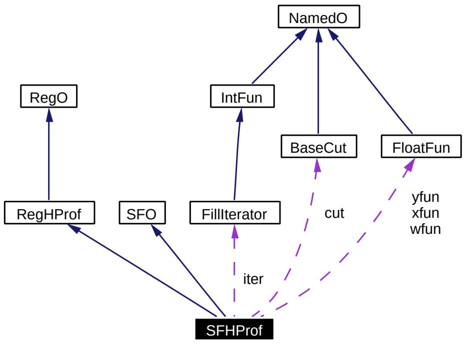
6.92 SFHProf Class Reference

`#include <SFHProf.h>`

Inheritance diagram for SFHProf:



Collaboration diagram for SFHProf:



6.92.1 Detailed Description

A self-filling 1-dimensional TProfile histogram.

A self-filling SFHProf object is a TProfile that registers itself in a **ROList** histogram list when it is created. It has a **Fill()** method that fills one or several entries into the histogram. It uses **FloatFun** and **BaseCut** objects to know what to fill, and possibly **FillIterator** objects in the case that several entries are to be made in one **Fill()** call.

Author: Jenny Böhme, Benno List

Changelog:

- 9.6.04 BL: Use HListPoR, **FloatFunPoR**, **BaseCutPoR**, **FillIteratorPoR**, documentation ugraded, added constructor with Binning objects
- 10.6.04 BL: Derived from class **SFO** instead of SFH
- 15.7.04 BL: Added method **checkIterators()**
- 15.7.04 BL: Added methods getXFun etc

Definition at line 47 of file SFHProf.h.

Public Member Functions

- **SFHProf** (const char ∗name, const char ∗title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with equidistant binning.

- **SFHProf** (const char ∗name, const char ∗title, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow, Axis_t yup, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with equidistant binning and limited y range.

- **SFHProf** (const char ∗name, const char ∗title, Int_t nbinsx, const Float_t ∗xbins, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant binning, taking float bin edges.

- **SFHProf** (const char ∗name, const char ∗title, Int_t nbinsx, const Double_t ∗xbins, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant binning, taking double bin edges.

- **SFHProf** (const char ∗name, const char ∗title, Int_t nbinsx, const Double_t ∗xbins, Axis_t ylow, Axis_t yup, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with non-equidistant binning and limited y range.

- **SFHProf** (const char ∗name, const char ∗title, const Binning &binning, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with Binning object.

- **SFHProf** (const char ∗name, const char ∗title, const Binning &binning, Axis_t ylow, Axis_t yup, Option_t ∗option, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_=0, const **FloatFunPoR** &wfun_=0, const **FillIteratorPoR** &iter_=0)

  Constructor with Binning object and limited y range.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** ∗ **getYFun** () const

  Returns a pointer to the ordinate function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗name)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗name, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗name, const char ∗funname, Fun ∗fun)

  Function object.

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFHProf** (const **SFHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFHProf** & **operator=** (const **SFHProf** &source)

  Private assignment operator, left unimplemented: No assignment!

6.92.2 Constructor & Destructor Documentation

6.92.2.1 SFHProf::SFHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with equidistant binning.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

*xfun_* Function object defining abscissa (what to plot on x)

*yfun_* Function object defining ordinate (what to plot on y)

*cut_* optional Function object defining a cut (0 means: no cut)

*wfun_* optional Function object defining weight (0 means: weight 1)

*iter_* optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 36 of file SFHProf.C.

References checkIterators().

6.92.2.2 SFHProf::SFHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow*, Axis_t *yup*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with equidistant binning and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

*xfun_* Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 56 of file SFHProf.C.

References checkIterators().

6.92.2.3 SFHProf::SFHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Float_t * *xbins*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant binning, taking float bin edges.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 77 of file SFHProf.C.

References checkIterators().

6.92.2.4 SFHProf::SFHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant binning, taking double bin edges.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 96 of file SFHProf.C.

References checkIterators().

6.92.2.5 SFHProf::SFHProf (const char * *name*, const char * *title*, Int_t *nbinsx*, const Double_t * *xbins*, Axis_t *ylow*, Axis_t *yup*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with non-equidistant binning and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 115 of file SFHProf.C.

References checkIterators().

6.92.2.6 SFHProf::SFHProf (const char * *name*, const char * *title*, const Binning & *binning*, Option_t * *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with Binning object.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

binning Defines the histogram binning

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 135 of file SFHProf.C.

References checkIterators().

6.92.2.7 SFHProf::SFHProf (const char ∗ *name*, const char ∗ *title*, const Binning & *binning*, Axis_t *ylow*, Axis_t *yup*, Option_t ∗ *option*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_* = 0, const FloatFunPoR & *wfun_* = 0, const FillIteratorPoR & *iter_* = 0)

Constructor with Binning object and limited y range.

Parameters:

name Histogran short name (as for TProfile)

title Histogram title (as for TProfile)

binning Defines the histogram binning

ylow Minimum y value (as for TProfile)

yup Maximum y value (as for TProfile)

option Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 155 of file SFHProf.C.

References checkIterators().

6.92.3 Member Function Documentation

6.92.3.1 void SFHProf::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the xfun object to get the abscissa, yfun to obtain the ordinate, and the wfun object to obtain the weight.

Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 179 of file SFHProf.C.

References cut, iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.92.3.2 void SFHProf::checkIterators (const char ∗ *name*) `[protected]`

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 236 of file SFHProf.C.

References checkOneIterator(), checkTwoIterators(), cut, wfun, xfun, and yfun.

Referenced by SFHProf().

6.92.3.3 template<class Fun1, class Fun2> void SFHProf::checkTwoIterators (const char ∗ *name*, const char ∗ *fun1name*, Fun1 ∗ *fun1*, const char ∗ *fun2name*, Fun2 ∗ *fun2*) `[protected]`

Checks consistency of iterators of two function objects.

Function object 2

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

Definition at line 192 of file SFHProf.C.

Referenced by checkIterators().

6.92.3.4 template<class Fun> void SFHProf::checkOneIterator (const char ∗ *name*, const char ∗ *funname*, Fun ∗ *fun*) `[protected]`

Function object.

Parameters:

name The histogram name

funname Name of function object

Definition at line 206 of file SFHProf.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

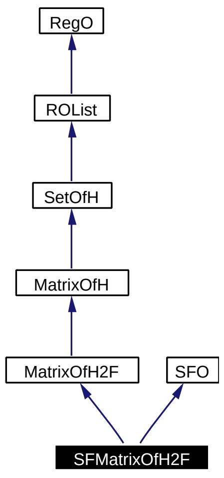
The documentation for this class was generated from the following files:

- **SFHProf.h**
- **SFHProf.C**

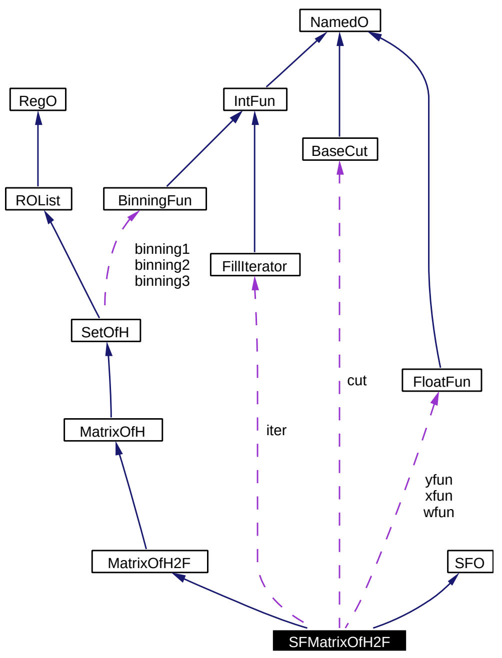
6.93 SFMatrixOfH2F Class Reference

`#include <SFMatrixOfH2F.h>`

Inheritance diagram for SFMatrixOfH2F:



Collaboration diagram for SFMatrixOfH2F:



6.93.1 Detailed Description

A 2-dimensional collection of 2-dimensional histograms that have the same axis.

A SFMatrixOfH2F contains a set of 1-dimensional TH2F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A SFMatrixOfH2F is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Jenny Böhme, Benno List

Date

2005/06/14 13:18:33

Author

blist

Changelog:

- 14.6.05 BL: First version

Definition at line 54 of file SFMatrixOfH2F.h.

Public Member Functions

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant x and y binnings.

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Double_t ∗xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t ∗ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Double_t ∗xbins, Int_t nbinsy, const Double_t ∗ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Float_t ∗xbins, Int_t nbinsy, const Float_t ∗ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_-, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

- **SFMatrixOfH2F** (const char ∗name_, const char ∗title_, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms, taking 2 Binning objects.

- virtual ∼**SFMatrixOfH2F** ()

  Virtual destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** ∗ **getYFun** () const

  Returns a pointer to the ordinate function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗**name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗**name**, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Checks consistency of iterator of function object with **FillIterator***.*

Private Member Functions

- **SFMatrixOfH2F** (const **SFMatrixOfH2F** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFMatrixOfH2F** & **operator=** (const **SFMatrixOfH2F** &source)

  Private assignment, left unimplemented: No assignment!

Private Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

6.93.2 Constructor & Destructor Documentation

6.93.2.1 SFMatrixOfH2F::SFMatrixOfH2F (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant x and y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 36 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.2.2 SFMatrixOfH2F::SFMatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 58 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.2.3 SFMatrixOfH2F::SFMatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 79 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.2.4 SFMatrixOfH2F::SFMatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 100 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.2.5 SFMatrixOfH2F::SFMatrixOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Int_t *nbinsy*, const Float_t * *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH1F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in x (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 120 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.2.6 SFMatrixOfH2F::SFMatrixOfH2F (const char * *name_*, const char * *title_*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms, taking 2 Binning objects.

Parameters:

name_ TH2F name

title_ TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 140 of file SFMatrixOfH2F.C.

References checkIterators().

6.93.3 Member Function Documentation

6.93.3.1 void SFMatrixOfH2F::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 163 of file SFMatrixOfH2F.C.

References cut, SetOfH::getBinning(), MatrixOfH::getBinNumber(), ROList::getEntries(), ROList::getEntry(), ROList::IndexType, BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.93.3.2 void SFMatrixOfH2F::checkIterators (const char * *name*) `[protected]`

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 301 of file SFMatrixOfH2F.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, xfun, and yfun.

Referenced by SFMatrixOfH2F().

6.93.3.3 template<class Fun1, class Fun2> void SFMatrixOfH2F::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) `[protected]`

Checks consistency of iterators of two function objects.

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

fun2 Function object 2

Definition at line 257 of file SFMatrixOfH2F.C.

Referenced by checkIterators().

6.93.3.4 template<class Fun> void SFMatrixOfH2F::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) `[protected]`

Checks consistency of iterator of function object with **FillIterator**.

Parameters:

name The histogram name

funname Name of function object

fun Function object

Definition at line 271 of file SFMatrixOfH2F.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

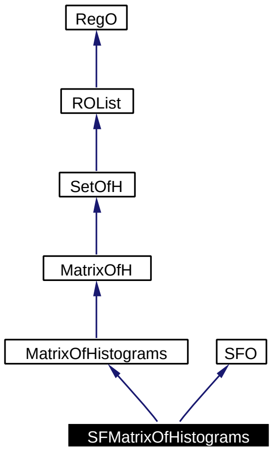
The documentation for this class was generated from the following files:

- **SFMatrixOfH2F.h**
- **SFMatrixOfH2F.C**

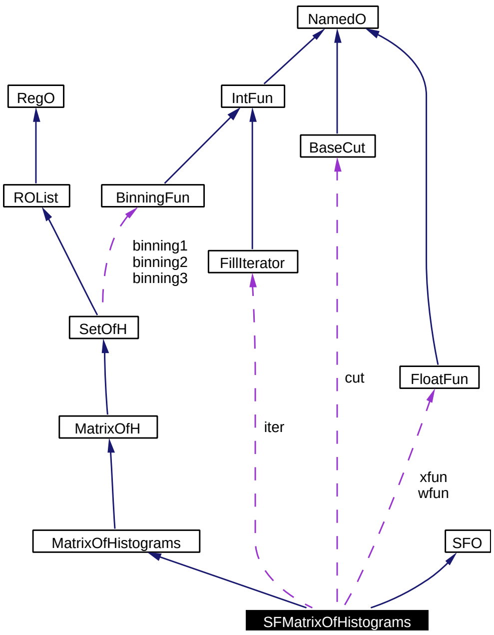
6.94 SFMatrixOfHistograms Class Reference

`#include <SFMatrixOfHistograms.h>`

Inheritance diagram for SFMatrixOfHistograms:



Collaboration diagram for SFMatrixOfHistograms:



6.94.1 Detailed Description

A self-filling 2-dimensional collection of 1-D histos with the same axis.

Like a **MatrixOfHistograms**, a SFMatrixOfHistograms contains a set of 1-dimensional TH1F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the two **BinningFun** objects. The same applies to the title.

A SFMatrixOfHistograms is a self-filling set of histograms, NOT a set of self-filling histograms (which is a **SFROList**). This means that instead of calling a **Fill()** of each histogram, the SFMatrixOfHistograms object uses the **BinningFun** objects to determine which of the histograms gets an entry, and then makes the entry in just that histogram.

A typical application of a SFMatrixOfHistograms is to generate a set of mass histograms depending on some kinematic quantities like t and W, to measure cross sections in different kinematic bins.

Using SFMatrixOfHistograms makes it unnecessary to define cut objects like "t between 0.05 and 0.1"

Author: Jenny Böhme, Benno List

Date
2005/07/08 14:49:14

Author
blist

Changelog:

- 10.6.04 BL: Additionally derived from **SFO**
- 22.6.04 BL: Added more documentation, use **ROListPoR**, **BinningFunPoR**, **FloatFunPoR**, **BaseCutPoR**, **FillIteratorPoR**, provide constructor with Binning object for axis binning
- 15.7.04 BL: Added method **checkIterators()**
- 15.7.04 BL: Added methods getXFun etc

Definition at line 74 of file SFMatrixOfHistograms.h.

Public Member Functions

- **SFMatrixOfHistograms** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_-, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of empty histograms.

- **SFMatrixOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant axis binning.

- **SFMatrixOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SFMatrixOfHistograms** (const char ∗name_, const char ∗title_, Int_t nbinsx, const Float_t ∗xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SFMatrixOfHistograms** (const char ∗name_, const char ∗title_, const Binning &**axisbinning**, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor taking a Binning object.

- virtual ∼**SFMatrixOfHistograms** ()

  Virtual destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗**name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗**name**, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Function object.

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFMatrixOfHistograms** (const **SFMatrixOfHistograms** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFMatrixOfHistograms** & **operator=** (const **SFMatrixOfHistograms** &source)

  Private assignment operator, left unimplemented: No assignment!

6.94.2 Constructor & Destructor Documentation

6.94.2.1 SFMatrixOfHistograms::SFMatrixOfHistograms (const char ∗ *name_*, const char ∗ *title_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of empty histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 40 of file SFMatrixOfHistograms.C.

References checkIterators().

6.94.2.2 SFMatrixOfHistograms::SFMatrixOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 55 of file SFMatrixOfHistograms.C.

References checkIterators().

6.94.2.3 SFMatrixOfHistograms::SFMatrixOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 73 of file SFMatrixOfHistograms.C.

References checkIterators().

6.94.2.4 SFMatrixOfHistograms::SFMatrixOfHistograms (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Float_t ∗ *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 90 of file SFMatrixOfHistograms.C.

References checkIterators().

6.94.2.5 SFMatrixOfHistograms::SFMatrixOfHistograms (const char ∗ *name_*, const char ∗ *title_*, const Binning & *axisbinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning Defines the histogram binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 107 of file SFMatrixOfHistograms.C.

References checkIterators().

6.94.3 Member Function Documentation

6.94.3.1 void SFMatrixOfHistograms::Fill () [virtual]

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** objects to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 125 of file SFMatrixOfHistograms.C.

References cut, SetOfH::getBinning(), MatrixOfH::getBinNumber(), ROList::getEntries(), ROList::getEntry(), ROList::IndexType, BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, and xfun.

6.94.3.2 void SFMatrixOfHistograms::checkIterators (const char ∗ *name*) [protected]

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 257 of file SFMatrixOfHistograms.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, and xfun.

Referenced by SFMatrixOfHistograms().

6.94.3.3 template<class Fun1, class Fun2> void SFMatrixOfHistograms::checkTwoIterators (const char ∗ *name*, const char ∗ *fun1name*, Fun1 ∗ *fun1*, const char ∗ *fun2name*, Fun2 ∗ *fun2*) [protected]

Checks consistency of iterators of two function objects.

Function object 2

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

Definition at line 213 of file SFMatrixOfHistograms.C.

Referenced by checkIterators().

6.94.3.4 template<class Fun> void SFMatrixOfHistograms::checkOneIterator (const char ∗ *name*, const char ∗ *funname*, Fun ∗ *fun*) [protected]

Function object.

Parameters:

name The histogram name

funname Name of function object

Definition at line 227 of file SFMatrixOfHistograms.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

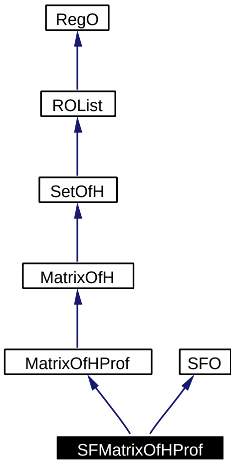
The documentation for this class was generated from the following files:

- **SFMatrixOfHistograms.h**
- **SFMatrixOfHistograms.C**

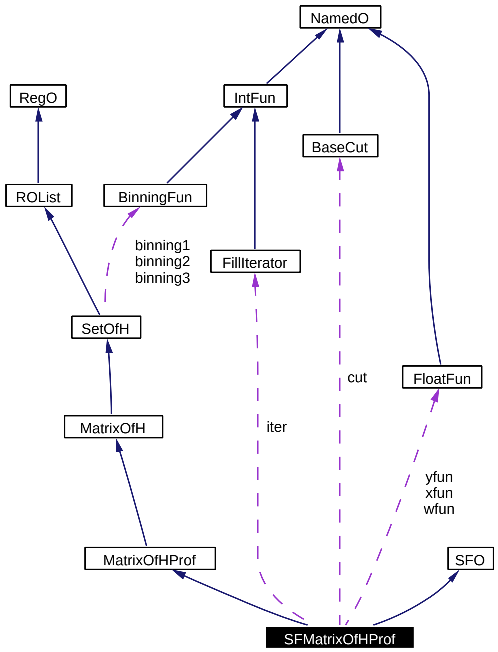
6.95 SFMatrixOfHProf Class Reference

`#include <SFMatrixOfHProf.h>`

Inheritance diagram for SFMatrixOfHProf:



Collaboration diagram for SFMatrixOfHProf:



6.95.1 Detailed Description

A self-filling, 2-dimensional collection of profile histograms that have the same axis.

A SFMatrixOfHProf contains a set of 1-dimensional THProfile histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using two **BinningFun** objects that at the same time define the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin names given by the **BinningFun** objects. The same applies to the title.

A SFMatrixOfHProf is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

A SFMatrixOfHProf is a self-filling set of histograms, NOT a set of self-filling histograms (which is a **SFROList**). This means that instead of calling a **Fill()** of each histogram, the SFMatrixOfHProf object uses the **BinningFun** objects to determine which of the histograms gets an entry, and then makes the entry in just that histogram.

Author: Jenny Böhme, Benno List

Date

2005/06/14 11:59:05

Author

blist

Changelog:

- 7.6.05 BL, JL: First version (derived from **MatrixOfHProf**)
- 15.7.04 BL: Added methods getXFun etc

Definition at line 60 of file SFMatrixOfHProf.h.

Public Member Functions

- **SFMatrixOfHProf** (const char *name_, const char *title_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for an empty set of histograms.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant axis binning.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant axis binning.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SFMatrixOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SFMatrixOfHProf** (const char *name_, const char *title_, const Binning &axisbinning_, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms taking a Binning object.

- **SFMatrixOfHProf** (const char *name_, const char *title_, const Binning &axisbinning_, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &rol, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning1_, const **BinningFunPoR** &binning2_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms taking a Binning object.

- virtual ~**SFMatrixOfHProf** ()

  Virtual destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** * **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** * **getYFun** () const

Returns a pointer to the ordinate function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Private Member Functions

- void **checkIterators** (const char ∗**name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗**name**, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Checks consistency of iterator of function object with **FillIterator**.

- **SFMatrixOfHProf** (const **SFMatrixOfHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFMatrixOfHProf** & **operator=** (const **SFMatrixOfHProf** &source)

  Private assignment, left unimplemented: No assignment!

Private Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

6.95.2 Constructor & Destructor Documentation

6.95.2.1 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for an empty set of histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 40 of file SFMatrixOfHProf.C.

6.95.2.2 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xlow Lower edge (as for TProfile)

xup Upper edge (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 55 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.3 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xlow Lower edge (as for TProfile)

xup Upper edge (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 75 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.4 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 97 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.5 SFMatrixOfHProf::SFMatrixOfHProf (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Double_t ∗ *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t ∗ *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 116 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.6 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 137 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.7 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TProfile)

xbins Bin edges (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 156 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.8 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 177 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.2.9 SFMatrixOfHProf::SFMatrixOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *rol*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning1_*, const BinningFunPoR & *binning2_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

rol **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning1_ The 1st **BinningFun** that assigns entries to histograms

binning2_ The 2nd **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 195 of file SFMatrixOfHProf.C.

References checkIterators().

6.95.3 Member Function Documentation

6.95.3.1 void SFMatrixOfHProf::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 220 of file SFMatrixOfHProf.C.

References cut, SetOfH::getBinning(), MatrixOfH::getBinNumber(), ROList::getEntries(), ROList::getEntry(), ROList::IndexType, BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.95.3.2 void SFMatrixOfHProf::checkIterators (const char * *name*) `[private]`

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 372 of file SFMatrixOfHProf.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, xfun, and yfun.

Referenced by SFMatrixOfHProf().

6.95.3.3 template<class Fun1, class Fun2> void SFMatrixOfHProf::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) `[private]`

Checks consistency of iterators of two function objects.

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

fun2 Function object 2

Definition at line 328 of file SFMatrixOfHProf.C.

Referenced by checkIterators().

6.95.3.4 template<class Fun> void SFMatrixOfHProf::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) `[private]`

Checks consistency of iterator of function object with **FillIterator**.

Parameters:

name The histogram name

funname Name of function object

fun Function object

Definition at line 342 of file SFMatrixOfHProf.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

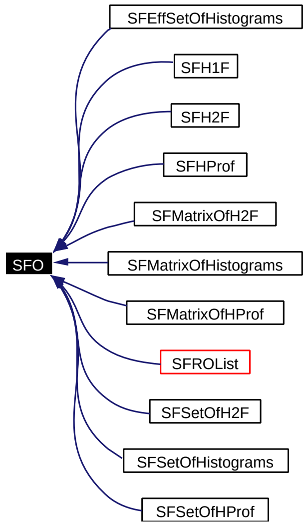
The documentation for this class was generated from the following files:

- **SFMatrixOfHProf.h**
- **SFMatrixOfHProf.C**

6.96 SFO Class Reference

`#include <SFO.h>`

Inheritance diagram for SFO:



6.96.1 Detailed Description

Abstract mix-in base class for self-filling objects.

SFO is the base class for all self-filling objects, whoose characteristic is that they have a **Fill()** method that does not require any arguments. This **Fill()** method is implemented in the derived classes using function objects.

The self-filling object can be a single histogram or a whole collection of histograms.

Author: Jenny Böhme, Benno List

Date

2004/06/15 11:05:46

Author

blist

Changelog:

- 10.6.04 BL: First version (adapted from class SFH)

Definition at line 30 of file SFO.h.

Public Member Functions

- **SFO** ()

  Default constructor, needed because other constructors are private.

- virtual ~**SFO** ()

    Virtual destructor to ensure that dynamic casts work.

- virtual void **Fill** ()=0

    The self-filling method.

Private Member Functions

- **SFO** (const **SFO** &rhs)

    Private copy constructor, left unimplemented: No copying!

- **SFO** & **operator=** (const **SFO** &rhs)

    Private assignment operator, left unimplemented: No assignment!

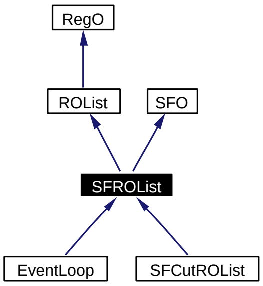
The documentation for this class was generated from the following file:

- **SFO.h**

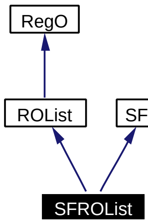
6.97 SFROList Class Reference

`#include <SFROList.h>`

Inheritance diagram for SFROList:



Collaboration diagram for SFROList:



6.97.1 Detailed Description

A list of histograms with a **Fill()** method.

Author: Benno List, Jenny Böhme

Date
2004/06/15 11:05:47

Author
blist

Changelog:

- 10.6.04 BL: First version, adapted from class SFHList

Definition at line 25 of file SFROList.h.

Public Member Functions

- **SFROList** ()
  Default constructor: creates an empty SFROList.

- **SFROList** (const **ROListPoR** &mother)
  Creates an empty **ROList** *and registers it with another list.*
- virtual void **Fill** ()
  Fills all self-filling histograms.

6.97.2 Constructor & Destructor Documentation

6.97.2.1 SFROList::SFROList (const ROListPoR & *mother*)

Creates an empty **ROList** and registers it with another list.

Parameters:
mother The "mother" list

Definition at line 22 of file SFROList.C.

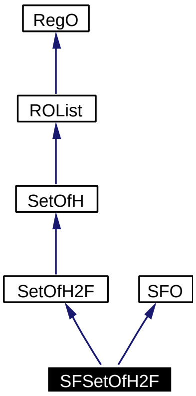
The documentation for this class was generated from the following files:

- **SFROList.h**
- **SFROList.C**

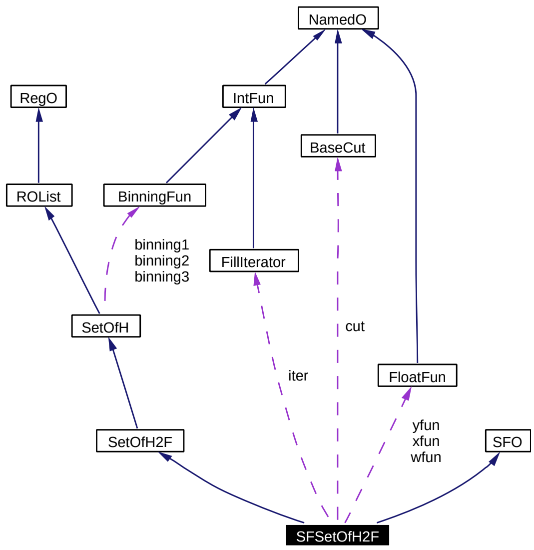
6.98 SFSetOfH2F Class Reference

`#include <SFSetOfH2F.h>`

Inheritance diagram for SFSetOfH2F:



Collaboration diagram for SFSetOfH2F:



6.98.1 Detailed Description

A collection of profile histograms that have the same axis.

A SFSetOfH2F contains a set of TH2F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SFSetOfH2F is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

Author: Benno List

Date

2005/06/10 09:22:11

Author

blist

Changelog:

- 10.6.5.05 BL: First version (from **SetOfHProf**)

Definition at line 51 of file SFSetOfH2F.h.

Public Member Functions

- **SFSetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant x and y binnings.

- **SFSetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Axis_t ylow, Axis_t yup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

- **SFSetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

- **SFSetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

- **SFSetOfH2F** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

- **SFSetOfH2F** (const char ∗name_, const char ∗title_, const Binning &xbinning, const Binning &ybinning, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms, taking 2 Binning objects.

- virtual ∼**SFSetOfH2F** ()

  Destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** ∗ **getYFun** () const

  Returns a pointer to the ordinate function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗**name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗**name**, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Checks consistency of iterator of function object with **FillIterator**.

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut * cut**
  optional Function object defining a cut (0 means: no cut)

- **FloatFun * wfun**
  optional Function object defining weight (0 means: weight 1)

- **FillIterator * iter**
  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFSetOfH2F** (const **SFSetOfH2F** &source)
  Private copy constructor, left unimplemented: No copying!

- **SFSetOfH2F** & **operator=** (const **SFSetOfH2F** &source)
  Private assignment operator, left unimplemented: No assignment!

6.98.2 Constructor & Destructor Documentation

6.98.2.1 SFSetOfH2F::SFSetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant x and y binnings.

Parameters:

name_ TH2F name
title_ TH2F title
nbinsx Number of x bins (as for TH2F)
xlow Lower x edge (as for TH2F)
xup Upper x edge (as for TH2F)
nbinsy Number of y bins (as for TH2F)
ylow Lower y edge (as for TH2F)
yup Upper y edge (as for TH2F)
hhl **ROList** object where "this" will be registered
xfun_ Function object defining abscissa (what to plot on x)
yfun_ Function object defining ordinate (what to plot on y)
cut_ optional Function object defining a cut (0 means: no cut)
wfun_ optional Function object defining weight (0 means: weight 1)
binning_ The **BinningFun** that assigns entries to histograms
iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 39 of file SFSetOfH2F.C.

References checkIterators().

6.98.2.2 SFSetOfH2F::SFSetOfH2F (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Double_t ∗ *xbins*, Int_t *nbinsy*, Axis_t *ylow*, Axis_t *yup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ylow Lower y edge (as for TH2F)

yup Upper y edge (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 60 of file SFSetOfH2F.C.

References checkIterators().

6.98.2.3 SFSetOfH2F::SFSetOfH2F (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Int_t *nbinsy*, const Double_t ∗ *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant x and non-equidistant y binnings.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xlow Lower x edge (as for TH2F)

xup Upper x edge (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 80 of file SFSetOfH2F.C.

References checkIterators().

6.98.2.4 SFSetOfH2F::SFSetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Int_t *nbinsy*, const Double_t * *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and y binnings taking doubles.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH2F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in y (as for TH2F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 100 of file SFSetOfH2F.C.

References checkIterators().

6.98.2.5 SFSetOfH2F::SFSetOfH2F (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Int_t *nbinsy*, const Float_t * *ybins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant x and y binnings taking floats.

Parameters:

name_ TH2F name

title_ TH2F title

nbinsx Number of x bins (as for TH2F)

xbins Bin edges in x (as for TH1F)

nbinsy Number of y bins (as for TH2F)

ybins Bin edges in x (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 119 of file SFSetOfH2F.C.

References checkIterators().

6.98.2.6 SFSetOfH2F::SFSetOfH2F (const char * *name_*, const char * *title_*, const Binning & *xbinning*, const Binning & *ybinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms, taking 2 Binning objects.

Parameters:

name_ TH2F name

title_ TH2F title

xbinning Defines the histogram x binning

ybinning Defines the histogram y binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 138 of file SFSetOfH2F.C.

References checkIterators().

6.98.3 Member Function Documentation

6.98.3.1 void SFSetOfH2F::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 159 of file SFSetOfH2F.C.

References cut, SetOfH::getBinning(), ROList::getEntry(), BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.98.3.2 void SFSetOfH2F::checkIterators (const char ∗ *name*) [protected]

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 240 of file SFSetOfH2F.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, xfun, and yfun.

Referenced by SFSetOfH2F().

6.98.3.3 template<class Fun1, class Fun2> void SFSetOfH2F::checkTwoIterators (const char ∗ *name*, const char ∗ *fun1name*, Fun1 ∗ *fun1*, const char ∗ *fun2name*, Fun2 ∗ *fun2*) [protected]

Checks consistency of iterators of two function objects.

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

fun2 Function object 2

Definition at line 196 of file SFSetOfH2F.C.

Referenced by checkIterators().

6.98.3.4 template<class Fun> void SFSetOfH2F::checkOneIterator (const char ∗ *name*, const char ∗ *funname*, Fun ∗ *fun*) [protected]

Checks consistency of iterator of function object with **FillIterator**.

Parameters:

name The histogram name

funname Name of function object

fun Function object

Definition at line 210 of file SFSetOfH2F.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

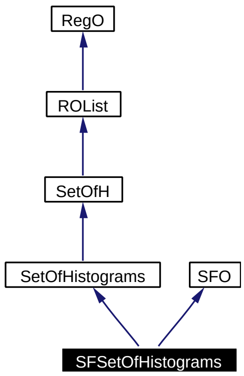
The documentation for this class was generated from the following files:

- **SFSetOfH2F.h**
- **SFSetOfH2F.C**

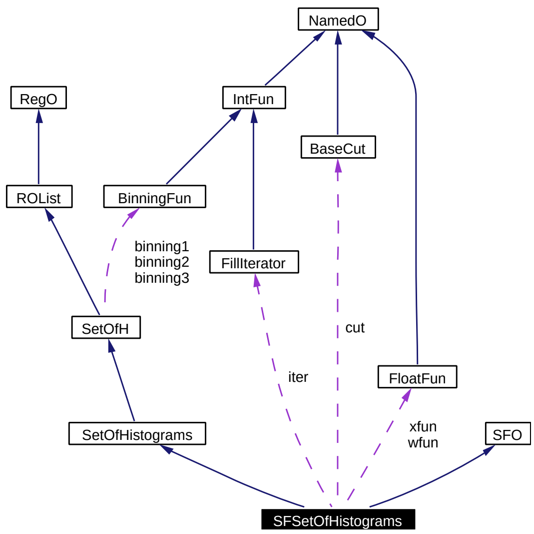
6.99 SFSetOfHistograms Class Reference

`#include <SFSetOfHistograms.h>`

Inheritance diagram for SFSetOfHistograms:



Collaboration diagram for SFSetOfHistograms:



6.99.1 Detailed Description

A self-filling collection of 1-dimensional histograms that have the same axis.

Like a **SetOfHistograms**, a SFSetOfHistograms contains a set of 1-dimensional TH1F histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SFSetOfHistograms is a self-filling set of histograms, NOT a set of self-filling histograms (which is a **SFROList**). This means that instead of calling a **Fill()** of each histogram, the SFSetOfHistograms object uses the **BinningFun** object to determine which of the histograms gets an entry, and then makes the entry in just that histogram.

A typical application of a SFSetOfHistograms is to generate a set of mass histograms depending on some kinematic quantity like t or W, to measure cross sections in different kinematic bins.

Using SFSetOfHistograms makes it unnecessary to define cut objects like "t between 0.05 and 0.1"

Author: Benno List, Jenny Böhme

Date

2005/05/31 13:36:00

Author

boehmej

Changelog:

- 10.6.04 BL: Additionally derived from **SFO**
- 15.7.04 BL: Added method **checkIterators()**
- 19.7.04 JB: Checked iterators of binning
- 21.7.04 BL: Introduced checkTwoIterators, checkOneIterator
- 31.5.05 JL, BL: Added accessors for function objects

Definition at line 61 of file SFSetOfHistograms.h.

Public Member Functions

- **SFSetOfHistograms** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of empty histograms.

- **SFSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant axis binning.

- **SFSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

- **SFSetOfHistograms** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

- **SFSetOfHistograms** (const char ∗name_, const char ∗title_, const Binning &**axisbinning**, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor taking a Binning object.

- virtual ∼**SFSetOfHistograms** ()

  Virtual destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** ∗ **getXFun** () const

  Returns a pointer to the abscissa function object.

- **BaseCut** ∗ **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** ∗ **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** ∗ **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ∗**name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ∗**name**, const char ∗fun1name, Fun1 ∗fun1, const char ∗fun2name, Fun2 ∗fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Checks consistency of iterator of function object with **FillIterator**.

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

optional Function object defining weight (0 means: weight 1)

- **FillIterator * iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFSetOfHistograms** (const **SFSetOfHistograms** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFSetOfHistograms** & **operator=** (const **SFSetOfHistograms** &source)

  Private assignment operator, left unimplemented: No assignment!

6.99.2 Constructor & Destructor Documentation

6.99.2.1 SFSetOfHistograms::SFSetOfHistograms (const char * *name_*, const char * *title_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of empty histograms.

Parameters:

*name_* Common histogram name

*title_* Common histogram title

hhl **ROList** object where "this" will be registered

*xfun_* Function object defining abscissa (what to plot)

*cut_* optional Function object defining a cut (0 means: no cut)

*wfun_* optional Function object defining weight (0 means: weight 1)

*binning_* The **BinningFun** that assigns entries to histograms

*iter_* optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 33 of file SFSetOfHistograms.C.

References checkIterators().

6.99.2.2 SFSetOfHistograms::SFSetOfHistograms (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant axis binning.

Parameters:

*name_* Common histogram name

*title_* Common histogram title

nbinsx Number of bins (as for TH1F)

xlow Lower edge (as for TH1F)

xup Upper edge (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 47 of file SFSetOfHistograms.C.

References checkIterators().

6.99.2.3 SFSetOfHistograms::SFSetOfHistograms (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Double_t ∗ *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking doubles.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 64 of file SFSetOfHistograms.C.

References checkIterators().

6.99.2.4 SFSetOfHistograms::SFSetOfHistograms (const char ∗ *name_*, const char ∗ *title_*, Int_t *nbinsx*, const Float_t ∗ *xbins*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant axis binning, taking floats.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of bins (as for TH1F)

xbins Bin edges (as for TH1F)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 80 of file SFSetOfHistograms.C.

References checkIterators().

6.99.2.5 SFSetOfHistograms::SFSetOfHistograms (const char * *name_*, const char * *title_*, const Binning & *axisbinning*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor taking a Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning Defines the histogram binning

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 96 of file SFSetOfHistograms.C.

References checkIterators().

6.99.3 Member Function Documentation

6.99.3.1 void SFSetOfHistograms::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 114 of file SFSetOfHistograms.C.

References cut, SetOfH::getBinning(), ROList::getEntry(), BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, and xfun.

6.99.3.2 void SFSetOfHistograms::checkIterators (const char * *name*) [protected]

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 216 of file SFSetOfHistograms.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, and xfun.

Referenced by SFSetOfHistograms().

6.99.3.3 template<class Fun1, class Fun2> void SFSetOfHistograms::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) [protected]

Checks consistency of iterators of two function objects.

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

fun2 Function object 2

Definition at line 172 of file SFSetOfHistograms.C.

Referenced by checkIterators().

6.99.3.4 template<class Fun> void SFSetOfHistograms::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) [protected]

Checks consistency of iterator of function object with **FillIterator**.

Parameters:

name The histogram name

funname Name of function object

fun Function object

Definition at line 186 of file SFSetOfHistograms.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

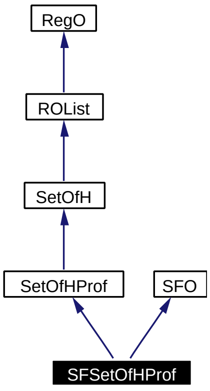
The documentation for this class was generated from the following files:

- **SFSetOfHistograms.h**
- **SFSetOfHistograms.C**

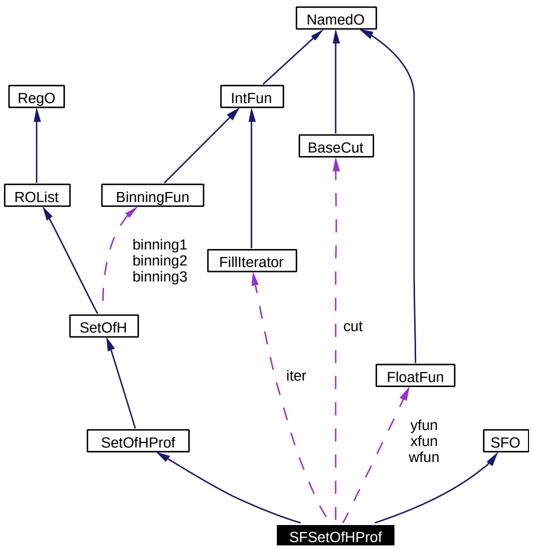
6.100 SFSetOfHProf Class Reference

`#include <SFSetOfHProf.h>`

Inheritance diagram for SFSetOfHProf:



Collaboration diagram for SFSetOfHProf:



6.100.1 Detailed Description

A self-filling collection of profile histograms that have the same axis.

A SFSetOfHProf contains a set of THProfile histograms; these histograms are (normally) all created by the constructor. The histograms share the same axis binning, and their names and titles are created using a **BinningFun** object that at the same time defines the number of histograms.

The histogram names will a concatenation of the name given by the user, and the bin name given by the **BinningFun**. The same applies to the title.

A SFSetOfHProf is a convenient tool to create a large set of histograms at the same time.

Functions exist to add, multiply, and divide sets of histograms. These operations are performed on pairs of corresponding histograms, if possible.

A SFSetOfHProf is a self-filling set of histograms, NOT a set of self-filling histograms (which is a **SFROList**). This means that instead of calling a **Fill()** of each histogram, the SFSetOfHProf object uses the **BinningFun** object to determine which of the histograms gets an entry, and then makes the entry in just that histogram.

Author: Benno List

Date
2005/06/07 17:03:24

Author
blist

Changelog:

- 13.5.05 BL: First version (from **SetOfHistograms**)
- 15.7.04 BL: Added methods getXFun etc

Definition at line 59 of file SFSetOfHProf.h.

Public Member Functions

- **SFSetOfHProf** (const char *name_, const char *title_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for an empty set histograms.

- **SFSetOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant binning.

- **SFSetOfHProf** (const char *name_, const char *title_, Int_t nbinsx, Axis_t xlow, Axis_t xup, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with equidistant binning and limited y range.

- **SFSetOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Float_t *xbins, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant binning, taking float bin edges.

- **SFSetOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

Constructor for a set of histograms with non-equidistant binning, taking double bin edges.

- **SFSetOfHProf** (const char *name_, const char *title_, Int_t nbinsx, const Double_t *xbins, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with non-equidistant binning and limited y range.

- **SFSetOfHProf** (const char *name_, const char *title_, const Binning &axisbinning_, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with Binning object.

- **SFSetOfHProf** (const char *name_, const char *title_, const Binning &axisbinning_, Axis_t ylow_, Axis_t yup_, Option_t *option_, const **ROListPoR** &hhl, const **FloatFunPoR** &xfun_, const **FloatFunPoR** &yfun_, const **BaseCutPoR** &cut_, const **FloatFunPoR** &wfun_, const **BinningFunPoR** &binning_, const **FillIteratorPoR** &iter_=0)

  Constructor for a set of histograms with Binning object and limited y range.

- virtual ~**SFSetOfHProf** ()

  Destructor.

- virtual void **Fill** ()

  The Self-filler method.

- **FloatFun** * **getXFun** () const

  Returns a pointer to the abscissa function object.

- **FloatFun** * **getYFun** () const

  Returns a pointer to the ordinate function object.

- **BaseCut** * **getCut** () const

  Returns a pointer to the cut function object.

- **FloatFun** * **getWFun** () const

  Returns a pointer to the weight function object.

- **FillIterator** * **getIter** () const

  Returns a pointer to the iterator object.

Protected Member Functions

- void **checkIterators** (const char ***name**)

  Checks consistency of iterators.

- template<class Fun1, class Fun2> void **checkTwoIterators** (const char ***name**, const char *fun1name, Fun1 *fun1, const char *fun2name, Fun2 *fun2)

  Checks consistency of iterators of two function objects.

- template<class Fun> void **checkOneIterator** (const char ∗**name**, const char ∗funname, Fun ∗fun)

  Checks consistency of iterator of function object with **FillIterator***.*

Protected Attributes

- **FloatFun** ∗ **xfun**

  Function object defining abscissa (what to plot on x).

- **FloatFun** ∗ **yfun**

  Function object defining ordinate (what to plot on y).

- **BaseCut** ∗ **cut**

  optional Function object defining a cut (0 means: no cut)

- **FloatFun** ∗ **wfun**

  optional Function object defining weight (0 means: weight 1)

- **FillIterator** ∗ **iter**

  optional Iterator object for more than 1 entry per **Fill()** *call*

Private Member Functions

- **SFSetOfHProf** (const **SFSetOfHProf** &source)

  Private copy constructor, left unimplemented: No copying!

- **SFSetOfHProf** & **operator=** (const **SFSetOfHProf** &source)

  Private assignment operator, left unimplemented: No assignment!

6.100.2 Constructor & Destructor Documentation

6.100.2.1 SFSetOfHProf::SFSetOfHProf (const char ∗ *name_*, const char ∗ *title_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for an empty set histograms.

Parameters:

name_ Common histogram name

title_ Common histogram title

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 38 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.2 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant binning.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 53 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.3 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, Axis_t *xlow*, Axis_t *xup*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with equidistant binning and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xlow Lower x edge (as for TProfile)

xup Upper x edge (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 72 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.4 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Float_t * *xbins*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant binning, taking float bin edges.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 93 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.5 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant binning, taking double bin edges.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 111 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.6 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, Int_t *nbinsx*, const Double_t * *xbins*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with non-equidistant binning and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

nbinsx Number of x bins (as for TProfile)

xbins Bin edges in x (as for TProfile)

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 129 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.7 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with Binning object.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 149 of file SFSetOfHProf.C.

References checkIterators().

6.100.2.8 SFSetOfHProf::SFSetOfHProf (const char * *name_*, const char * *title_*, const Binning & *axisbinning_*, Axis_t *ylow_*, Axis_t *yup_*, Option_t * *option_*, const ROListPoR & *hhl*, const FloatFunPoR & *xfun_*, const FloatFunPoR & *yfun_*, const BaseCutPoR & *cut_*, const FloatFunPoR & *wfun_*, const BinningFunPoR & *binning_*, const FillIteratorPoR & *iter_* = 0)

Constructor for a set of histograms with Binning object and limited y range.

Parameters:

name_ Common histogram name

title_ Common histogram title

axisbinning_ Defines the histogram binning

ylow_ Minimum y value (as for TProfile)

yup_ Maximum y value (as for TProfile)

option_ Option string (as for TProfile)

hhl **ROList** object where "this" will be registered

xfun_ Function object defining abscissa (what to plot on x)

yfun_ Function object defining ordinate (what to plot on y)

cut_ optional Function object defining a cut (0 means: no cut)

wfun_ optional Function object defining weight (0 means: weight 1)

binning_ The **BinningFun** that assigns entries to histograms

iter_ optional Iterator object for more than 1 entry per **Fill()** call

Definition at line 166 of file SFSetOfHProf.C.

References checkIterators().

6.100.3 Member Function Documentation

6.100.3.1 void SFSetOfHProf::Fill () `[virtual]`

The Self-filler method.

Fill() evaluates the cut object to see whether an event fulfills the cut defined by the cut object, then evaluates the **BinningFun** object to see into which histogram to make an entry, then evaluates the xfun object to get the abscissa and the wfun object to obtain the weight. Optionally, the iter object serves as iterator.

Implements **SFO** (p. 407).

Definition at line 190 of file SFSetOfHProf.C.

References cut, SetOfH::getBinning(), ROList::getEntry(), BinningFun::isExclusive(), iter, FillIterator::next(), FillIterator::reset(), wfun, xfun, and yfun.

6.100.3.2 void SFSetOfHProf::checkIterators (const char * *name*) `[protected]`

Checks consistency of iterators.

Parameters:

name The histogram name

Definition at line 272 of file SFSetOfHProf.C.

References checkOneIterator(), checkTwoIterators(), cut, SetOfH::getBinning(), wfun, xfun, and yfun.

Referenced by SFSetOfHProf().

6.100.3.3 template<class Fun1, class Fun2> void SFSetOfHProf::checkTwoIterators (const char * *name*, const char * *fun1name*, Fun1 * *fun1*, const char * *fun2name*, Fun2 * *fun2*) `[protected]`

Checks consistency of iterators of two function objects.

Parameters:

name The histogram name

fun1name Name of function object 1

fun1 Function object 1

fun2name Name of function object 2

fun2 Function object 2

Definition at line 228 of file SFSetOfHProf.C.

Referenced by checkIterators().

6.100.3.4 template<class Fun> void SFSetOfHProf::checkOneIterator (const char * *name*, const char * *funname*, Fun * *fun*) `[protected]`

Checks consistency of iterator of function object with **FillIterator**.

Parameters:

name The histogram name

funname Name of function object

fun Function object

Definition at line 242 of file SFSetOfHProf.C.

References NamedO::getName(), and iter.

Referenced by checkIterators().

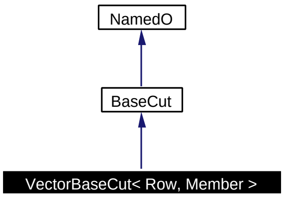
The documentation for this class was generated from the following files:

- **SFSetOfHProf.h**
- **SFSetOfHProf.C**

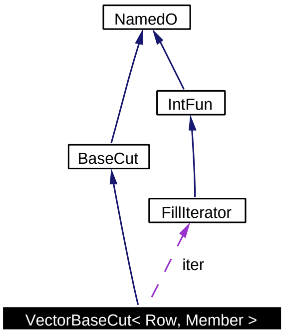
6.101 VectorBaseCut< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorBaseCut< Row, Member >:



Collaboration diagram for VectorBaseCut< Row, Member >:



6.101.1 Detailed Description

template<class Row, class Member = bool> class VectorBaseCut< Row, Member >

Returns a data member of a vector's row in a **BaseCut**.

Definition at line 323 of file VectorIterator.h.

Public Member Functions

- **VectorBaseCut** (const std::vector< Row > &theVector_, Member Row::*p_mem_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- bool **operator()** () const

  Returns the value of the data member, converted to bool.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorBaseCut** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member Row::* **p_mem**
  Member pointer to data member.
- const **FillIterator** & **iter**
  The iterator.

6.101.2 Constructor & Destructor Documentation

6.101.2.1 template<class Row, class Member = bool> VectorBaseCut< Row, Member >::VectorBaseCut (const std::vector< Row > & *theVector_*, Member Row::* *p_mem_*, const FillIterator & *iter_*, const char * *name_* = "?") `[inline]`

Constructor.

Parameters:
*theVector_* The vector
*p_mem_* Member pointer
*iter_* The iterator
*name_* The variable name

Definition at line 326 of file VectorIterator.h.

References NamedO::getName(), VectorBaseCut< Row, Member >::iter, VectorBaseCut< Row, Member >::p_mem, and VectorBaseCut< Row, Member >::theVector.

6.101.3 Member Data Documentation

6.101.3.1 template<class Row, class Member = bool> Member Row::* VectorBaseCut< Row, Member >::p_mem `[protected]`

Member pointer to data member.

p_mem is a pointer to a data member of T (indicated by "Row::*"), which has to be declared as Member member_name; (indicated by "Member"). Member must be convertible to bool.

Definition at line 356 of file VectorIterator.h.

Referenced by VectorBaseCut< Row, Member >::operator()(), and VectorBaseCut< Row, Member >::VectorBaseCut().

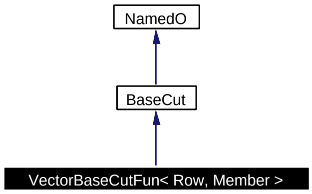
The documentation for this class was generated from the following file:

- **VectorIterator.h**

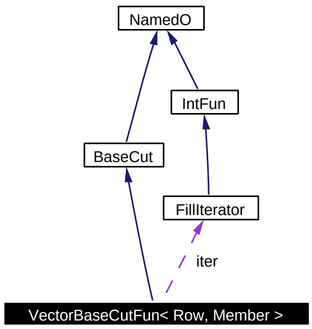
6.102 VectorBaseCutFun< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorBaseCutFun< Row, Member >:



Collaboration diagram for VectorBaseCutFun< Row, Member >:



6.102.1 Detailed Description

template<class Row, class Member = float> class VectorBaseCutFun< Row, Member >

Returns the result of a member method of a vector's row in a baseCut.

Definition at line 371 of file VectorIterator.h.

Public Member Functions

- **VectorBaseCutFun** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)() const, const **FillIterator** &iter_, const char *name_="?")

    Constructor.

- bool **operator()** () const

    Returns the value of the member method, converted to bool.

- virtual const **FillIterator** * **getIterator** () const

    Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorBaseCutFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member(Row::∗ **p_mem**)() const
  Member pointer to data member.
- const **FillIterator** & **iter**
  The iterator.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 template<class Row, class Member = float> VectorBaseCutFun< Row, Member >::VectorBaseCutFun (const std::vector< Row > & *theVector_*, Member(Row::∗ *p_mem_*)() const, const FillIterator & *iter_*, const char ∗ *name_* = "?") `[inline]`

Constructor.

Parameters:

theVector_ The vector

p_mem_ Member pointer

iter_ The iterator

name_ The variable name

Definition at line 374 of file VectorIterator.h.

References NamedO::getName(), VectorBaseCutFun< Row, Member >::iter, VectorBaseCutFun< Row, Member >::p_mem, and VectorBaseCutFun< Row, Member >::theVector.

6.102.3 Member Data Documentation

6.102.3.1 template<class Row, class Member = float> Member(Row::∗ VectorBaseCutFun< Row, Member >::p_mem)() const `[protected]`

Member pointer to data member.

p_mem is a pointer to a member function of T (indicated by "Row::∗"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to bool.

Referenced by VectorBaseCutFun< Row, Member >::operator()(), and VectorBaseCutFun< Row, Member >::VectorBaseCutFun().

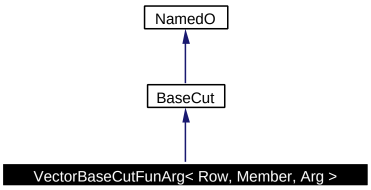
The documentation for this class was generated from the following file:

- **VectorIterator.h**

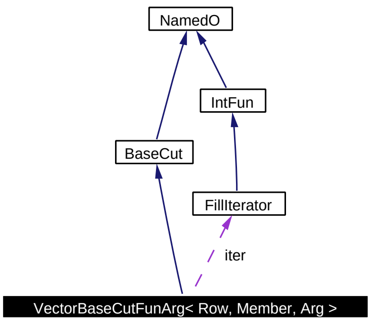
6.103 VectorBaseCutFunArg< Row, Member, Arg > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorBaseCutFunArg< Row, Member, Arg >:



Collaboration diagram for VectorBaseCutFunArg< Row, Member, Arg >:



6.103.1 Detailed Description

template<class Row, class Member = float, class Arg = bool> class VectorBaseCutFunArg< Row, Member, Arg >

Returns the result of a member method of a vector's row that takes an argument in a baseCut.

Definition at line 419 of file VectorIterator.h.

Public Member Functions

- **VectorBaseCutFunArg** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)(Arg) const, Arg arg_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- bool **operator()** () const

  Returns the value of the member method, converted to bool.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorBaseCutFunArg** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member(Row::∗ **p_mem**)(Arg) const
  Member pointer to data member.
- Arg **arg**
  The argument.
- const **FillIterator** & **iter**
  The iterator.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 template<class Row, class Member = float, class Arg = bool> VectorBaseCutFunArg< Row, Member, Arg >::VectorBaseCutFunArg (const std::vector< Row > & *theVector_*, Member(Row::∗ *p_mem_*)(Arg) const, Arg *arg_*, const FillIterator & *iter_*, const char ∗ *name_* = "?") `[inline]`

Constructor.

Parameters:

theVector_ The vector

p_mem_ Member pointer

arg_ The argument

iter_ The iterator

name_ The variable name

Definition at line 422 of file VectorIterator.h.

References VectorBaseCutFunArg< Row, Member, Arg >::arg, NamedO::getName(), VectorBaseCutFunArg< Row, Member, Arg >::iter, VectorBaseCutFunArg< Row, Member, Arg >::p_mem, and VectorBaseCutFunArg< Row, Member, Arg >::theVector.

6.103.3 Member Data Documentation

6.103.3.1 template<class Row, class Member = float, class Arg = bool> Arg VectorBaseCutFunArg< Row, Member, Arg >::arg `[protected]`

The argument.

p_mem is a pointer to a member function of T (indicated by "Row::*"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to bool.

Definition at line 455 of file VectorIterator.h.

Referenced by VectorBaseCutFunArg< Row, Member, Arg >::operator()(), and VectorBaseCutFunArg< Row, Member, Arg >::VectorBaseCutFunArg().

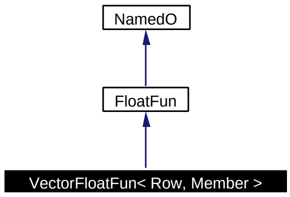
The documentation for this class was generated from the following file:

- **VectorIterator.h**

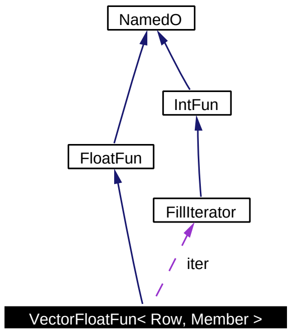
6.104 VectorFloatFun< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorFloatFun< Row, Member >:



Collaboration diagram for VectorFloatFun< Row, Member >:



6.104.1 Detailed Description

template<class Row, class Member = float> class VectorFloatFun< Row, Member >

Returns a data member of a vector's row in a **FloatFun**.

Definition at line 29 of file VectorIterator.h.

Public Member Functions

- **VectorFloatFun** (const std::vector< Row > &theVector_, Member Row::*p_mem_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- Float_FF **operator()** () const

  Returns the value of the data member, converted to float.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ∼**VectorFloatFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member Row::∗ **p_mem**
  Member pointer to data member.
- const **FillIterator** & **iter**
  The iterator.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 template<class Row, class Member = float> VectorFloatFun< Row, Member >::VectorFloatFun (const std::vector< Row > & *theVector_*, Member Row::∗ *p_mem_*, const FillIterator & *iter_*, const char ∗ *name_* = "?") `[inline]`

Constructor.

Parameters:
*theVector_* The vector
*p_mem_* Member pointer
*iter_* The iterator
*name_* The variable name

Definition at line 32 of file VectorIterator.h.

References NamedO::getName(), VectorFloatFun< Row, Member >::iter, VectorFloatFun< Row, Member >::p_mem, and VectorFloatFun< Row, Member >::theVector.

6.104.3 Member Data Documentation

6.104.3.1 template<class Row, class Member = float> Member Row::∗ VectorFloatFun< Row, Member >::p_mem `[protected]`

Member pointer to data member.

p_mem is a pointer to a data member of T (indicated by "Row::∗"), which has to be declared as Member member_name; (indicated by "Member"). Member must be convertible to float.

Definition at line 62 of file VectorIterator.h.

Referenced by VectorFloatFun< Row, Member >::operator()(), and VectorFloatFun< Row, Member >::VectorFloatFun().

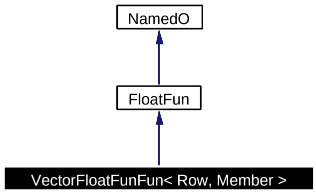
The documentation for this class was generated from the following file:

- **VectorIterator.h**

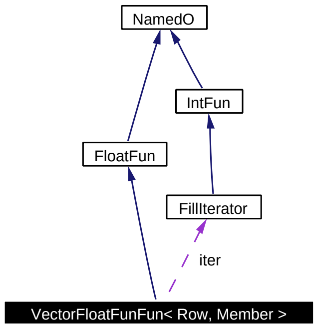
6.105 VectorFloatFunFun< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorFloatFunFun< Row, Member >:



Collaboration diagram for VectorFloatFunFun< Row, Member >:



6.105.1 Detailed Description

template<class Row, class Member = float> class VectorFloatFunFun< Row, Member >

Returns the result of a member method of a vector's row in a **FloatFun**.

Definition at line 77 of file VectorIterator.h.

Public Member Functions

- **VectorFloatFunFun** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)() const, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- Float_FF **operator()** () const

  Returns the value of the member method, converted to float.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorFloatFunFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member(Row::∗ **p_mem**)() const
  Member pointer to member function.
- const **FillIterator** & **iter**
  The iterator.

6.105.2 Constructor & Destructor Documentation

6.105.2.1 template<class Row, class Member = float> VectorFloatFunFun< Row, Member >::VectorFloatFunFun (const std::vector< Row > & *theVector_*, Member(Row::∗ *p_mem_*)() const, const FillIterator & *iter_*, const char ∗ *name_* = "?") [inline]

Constructor.

Parameters:
theVector_ The vector
p_mem_ Member pointer
iter_ The iterator
name_ The variable name

Definition at line 80 of file VectorIterator.h.

References NamedO::getName(), VectorFloatFunFun< Row, Member >::iter, VectorFloatFunFun< Row, Member >::p_mem, and VectorFloatFunFun< Row, Member >::theVector.

6.105.3 Member Data Documentation

6.105.3.1 template<class Row, class Member = float> Member(Row::∗ VectorFloatFunFun< Row, Member >::p_mem)() const [protected]

Member pointer to member function.

p_mem is a pointer to a member function of T (indicated by "Row::∗"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to float.

Referenced by VectorFloatFunFun< Row, Member >::operator()(), and VectorFloatFunFun< Row, Member >::VectorFloatFunFun().

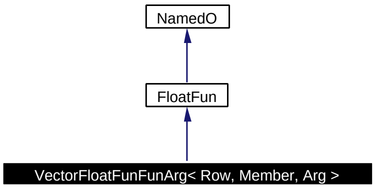
The documentation for this class was generated from the following file:

- **VectorIterator.h**

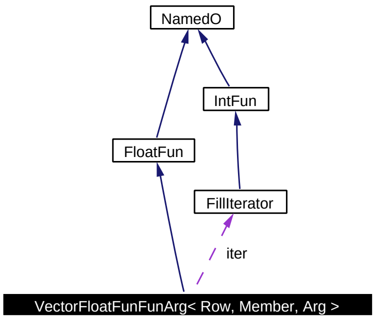
6.106 VectorFloatFunFunArg< Row, Member, Arg > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorFloatFunFunArg< Row, Member, Arg >:



Collaboration diagram for VectorFloatFunFunArg< Row, Member, Arg >:



6.106.1 Detailed Description

template<class Row, class Member = float, class Arg = float> class VectorFloatFunFunArg< Row, Member, Arg >

Returns the result of a member method of a vector's row that takes an argument in a **FloatFun**.

Definition at line 125 of file VectorIterator.h.

Public Member Functions

- **VectorFloatFunFunArg** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)(Arg) const, Arg arg_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- Float_FF **operator()** () const

  Returns the value of the member method, converted to float.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorFloatFunFunArg** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member(Row::* **p_mem**)(Arg) const
  Member pointer to member function.
- Arg **arg**
  The argument.
- const **FillIterator** & **iter**
  The iterator.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 template<class Row, class Member = float, class Arg = float> VectorFloatFunFunArg< Row, Member, Arg >::VectorFloatFunFunArg (const std::vector< Row > & *theVector_*, Member(Row::* *p_mem_*)(Arg) const, Arg *arg_*, const FillIterator & *iter_*, const char * *name_* = "?") [inline]

Constructor.

Parameters:

theVector_ The vector

p_mem_ Member pointer

arg_ The argument

iter_ The iterator

name_ The variable name

Definition at line 128 of file VectorIterator.h.

References VectorFloatFunFunArg< Row, Member, Arg >::arg, NamedO::getName(), VectorFloatFunFunArg< Row, Member, Arg >::iter, VectorFloatFunFunArg< Row, Member, Arg >::p_mem, and VectorFloatFunFunArg< Row, Member, Arg >::theVector.

6.106.3 Member Data Documentation

6.106.3.1 template<class Row, class Member = float, class Arg = float> Arg VectorFloatFunFunArg< Row, Member, Arg >::arg [protected]

The argument.

p_mem is a pointer to a member function of T (indicated by "Row::*"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to float.

Definition at line 161 of file VectorIterator.h.

Referenced by VectorFloatFunFunArg< Row, Member, Arg >::operator()(), and VectorFloatFunFunArg< Row, Member, Arg >::VectorFloatFunFunArg().

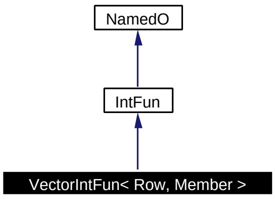
The documentation for this class was generated from the following file:

- **VectorIterator.h**

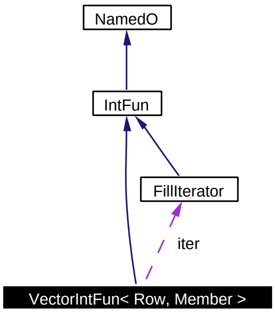
6.107 VectorIntFun< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorIntFun< Row, Member >:



Collaboration diagram for VectorIntFun< Row, Member >:



6.107.1 Detailed Description

template<class Row, class Member = int> class VectorIntFun< Row, Member >

Returns a data member of a vector's row in a **IntFun**.

Definition at line 176 of file VectorIterator.h.

Public Member Functions

- **VectorIntFun** (const std::vector< Row > &theVector_, Member Row::*p_mem_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- int **operator()** () const

  Returns the value of the data member, converted to int.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorIntFun** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member Row::* **p_mem**
  Member pointer to data member.
- const **FillIterator** & **iter**
  The iterator.

6.107.2 Constructor & Destructor Documentation

6.107.2.1 template<class Row, class Member = int> VectorIntFun< Row, Member >::VectorIntFun (const std::vector< Row > & *theVector_*, Member Row::* *p_mem_*, const FillIterator & *iter_*, const char * *name_* = "?") [inline]

Constructor.

Parameters:
*theVector_* The vector
*p_mem_* Member pointer
*iter_* The iterator
*name_* The variable name

Definition at line 179 of file VectorIterator.h.

References NamedO::getName(), VectorIntFun< Row, Member >::iter, VectorIntFun< Row, Member >::p_mem, and VectorIntFun< Row, Member >::theVector.

6.107.3 Member Data Documentation

6.107.3.1 template<class Row, class Member = int> Member Row::* VectorIntFun< Row, Member >::p_mem [protected]

Member pointer to data member.

p_mem is a pointer to a data member of T (indicated by "Row::*"), which has to be declared as Member member_name; (indicated by "Member"). Member must be convertible to int.

Definition at line 209 of file VectorIterator.h.

Referenced by VectorIntFun< Row, Member >::operator()(), and VectorIntFun< Row, Member >::VectorIntFun().

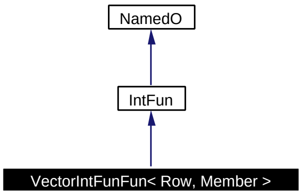
The documentation for this class was generated from the following file:

- **VectorIterator.h**

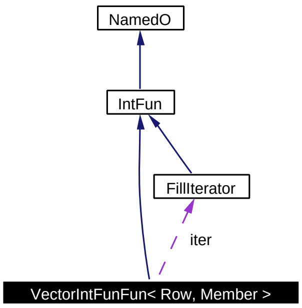
6.108 VectorIntFunFun< Row, Member > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorIntFunFun< Row, Member >:



Collaboration diagram for VectorIntFunFun< Row, Member >:



6.108.1 Detailed Description

template<class Row, class Member = int> class VectorIntFunFun< Row, Member >

Returns the result of a member method of a vector's row in a **IntFun**.

Definition at line 224 of file VectorIterator.h.

Public Member Functions

- **VectorIntFunFun** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)() const, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- int **operator()** () const

  Returns the value of the member method, converted to int.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ∼**VectorIntFunFun** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**

  The vector.

- Member(Row::∗ **p_mem**)() const

  Member pointer to data member.

- const **FillIterator** & **iter**

  The iterator.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 template<class Row, class Member = int> VectorIntFunFun< Row, Member >::VectorIntFunFun (const std::vector< Row > & *theVector_*, Member(Row::∗ *p_mem_*)() const, const FillIterator & *iter_*, const char ∗ *name_* = "?") `[inline]`

Constructor.

Parameters:

theVector_ The vector

p_mem_ Member pointer

iter_ The iterator

name_ The variable name

Definition at line 227 of file VectorIterator.h.

References NamedO::getName(), VectorIntFunFun< Row, Member >::iter, VectorIntFunFun< Row, Member >::p_mem, and VectorIntFunFun< Row, Member >::theVector.

6.108.3 Member Data Documentation

6.108.3.1 template<class Row, class Member = int> Member(Row::∗ VectorIntFunFun< Row, Member >::p_mem)() const `[protected]`

Member pointer to data member.

p_mem is a pointer to a member function of T (indicated by "Row::∗"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to int.

Referenced by VectorIntFunFun< Row, Member >::operator()(), and VectorIntFunFun< Row, Member >::VectorIntFunFun().

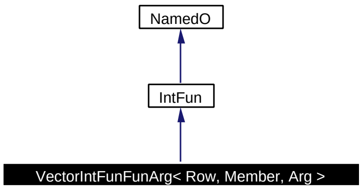
The documentation for this class was generated from the following file:

- **VectorIterator.h**

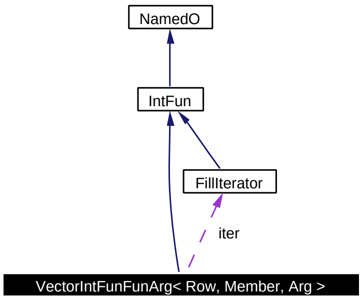
6.109 VectorIntFunFunArg< Row, Member, Arg > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorIntFunFunArg< Row, Member, Arg >:



Collaboration diagram for VectorIntFunFunArg< Row, Member, Arg >:



6.109.1 Detailed Description

template<class Row, class Member = int, class Arg = int> class VectorIntFunFunArg< Row, Member, Arg >

Returns the result of a member method of a vector's row that takes an argument in a **IntFun**.

Definition at line 272 of file VectorIterator.h.

Public Member Functions

- **VectorIntFunFunArg** (const std::vector< Row > &theVector_, Member(Row::*p_mem_)(Arg) const, Arg arg_, const **FillIterator** &iter_, const char *name_="?")

  Constructor.

- int **operator()** () const

  Returns the value of the member method, converted to int.

- virtual const **FillIterator** * **getIterator** () const

  Returns pointer to iterator.

Protected Member Functions

- virtual ~**VectorIntFunFunArg** ()
  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**
  The vector.
- Member(Row::* **p_mem**)(Arg) const
  Member pointer to data member.
- Arg **arg**
  The argument.
- const **FillIterator** & **iter**
  The iterator.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 template<class Row, class Member = int, class Arg = int> VectorIntFunFunArg< Row, Member, Arg >::VectorIntFunFunArg (const std::vector< Row > & *theVector_*, Member(Row::* *p_mem_*)(Arg) const, Arg *arg_*, const FillIterator & *iter_*, const char * *name_* = "?") [inline]

Constructor.

Parameters:

theVector_ The vector

p_mem_ Member pointer

arg_ The argument

iter_ The iterator

name_ The variable name

Definition at line 275 of file VectorIterator.h.

References VectorIntFunFunArg< Row, Member, Arg >::arg, NamedO::getName(), VectorIntFunFunArg< Row, Member, Arg >::iter, VectorIntFunFunArg< Row, Member, Arg >::p_mem, and VectorIntFunFunArg< Row, Member, Arg >::theVector.

6.109.3 Member Data Documentation

6.109.3.1 template<class Row, class Member = int, class Arg = int> Arg VectorIntFunFunArg< Row, Member, Arg >::arg [protected]

The argument.

p_mem is a pointer to a member function of T (indicated by "Row::*"), which has to be declared as Member function_name() const; (indicated by "Member(...)() const"). Member must be convertible to int.

Definition at line 308 of file VectorIterator.h.

Referenced by VectorIntFunFunArg< Row, Member, Arg >::operator()(), and VectorIntFunFunArg< Row, Member, Arg >::VectorIntFunFunArg().

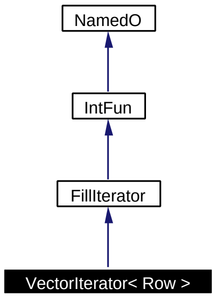
The documentation for this class was generated from the following file:

- **VectorIterator.h**

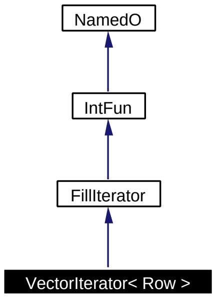
6.110 VectorIterator< Row > Class Template Reference

`#include <VectorIterator.h>`

Inheritance diagram for VectorIterator< Row >:



Collaboration diagram for VectorIterator< Row >:



6.110.1 Detailed Description

template<class Row> class VectorIterator< Row >

An iterator over a vector with row type Row.

If we have a vector with a row type Row, e.g. vector<FooBar>, then a **VectorFloatFun** allows to iterate over all elements of the vector and return the value of one data member.

Let's assume we have a class FooBar defined like this:

```
class FooBar {
  public:
    float foo;
    int bar;
    float foobar() const { return foo*bar; }
};
```

and our DataLoop (derived from **EventLoop**, of course) has a data member foobarvector:

```
class DataLoop: public EventLoop {
  public:
    DataLoop();
```

```
    ~DataLoop();
    void loop();
  private:
    vector<FooBar> foobarvector;
};
```

We can for instance fill foobarvector in DataLoop:loop():

```
void DataLoop::loop() {
  // fill foobarvector here
  foobarvector[xxx] = yyy;
  // Now fill the self-filling histos:
  EventLoop::loop();
}
```

Now, to access the contents of foobarvector, we can do things like this:

```
void DataLoop::DataLoop() {
  VectorIterator<FooBar>& foobar_it =
      *new VectorIterator<FooBar> (foobarvector);
  FloatFun& foo_fun = foobar_it.Float (&FooBar::foo);
  FloatFun& bar_fun = foobar_it.Float (&FooBar::bar);
  FloatFun& foobar_fun = foobar_it.Float (&FooBar::foobar);
}
```

These **FloatFun** objects will return the value of data members (foo or bar), or even the result of an access method like foobar(). This is achieved by creating objects of type **VectorFloatFun** (for access to data members) or **VectorFloatFunFun** (for access to member methods).

The notation &FooBar::foo is probably unfamiliar. Such a thing is called a member pointer. Essentially it is a kind of offset that describes where within an object a data member is stored. For more information, see any good C++ text book.

Author: Benno List, Jenny Böhme

Date

2006/09/12 09:24:48

Author

zimmerto

Definition at line 533 of file VectorIterator.h.

Public Member Functions

- **VectorIterator** (const std::vector< Row > &theVector_, const char *name_="?")

  Constructor.

- virtual int **operator()** () const

  Returns iterator value, starting at 0.

- virtual bool **next** ()

  Increments iterator; returns false if iterator cannot be incremented.

- virtual bool **reset** ()

Resets iterator; returns false if iterator value range is empty.

- virtual bool **isValid** () const

  Returns whether current value of iterator is valid.

- template<class Member> **VectorFloatFun**< Row, Member > & **Float** (Member Row::*p_mem, const char *name_=”?”) const

  Returns a **FloatFun** *that returns a data member of T.*

- template<class Member> **VectorFloatFunFun**< Row, Member > & **Float** (Member(Row::*p_mem)() const, const char *name_=”?”) const

  Returns a **FloatFun** *that returns the result of a member method of T.*

- template<class Member, class Arg> **VectorFloatFunFunArg**< Row, Member, Arg > & **Float** (Member(Row::*p_mem)(Arg) const, Arg arg_, const char *name_=”?”) const

  Returns a **FloatFun** *that returns the result of a member method of T that takes an argument.*

- template<class Member> **VectorIntFun**< Row, Member > & **Int** (Member Row::*p_mem, const char *name_=”?”) const

  Returns a **IntFun** *that returns a data member of T.*

- template<class Member> **VectorIntFunFun**< Row, Member > & **Int** (Member(Row::*p_mem)() const, const char *name_=”?”) const

  Returns a **IntFun** *that returns the result of a member method of T.*

- template<class Member, class Arg> **VectorIntFunFunArg**< Row, Member, Arg > & **Int** (Member(Row::*p_mem)(Arg) const, Arg arg_, const char *name_=”?”) const

  Returns a **IntFun** *that returns the result of a member method of T that takes an argument.*

- template<class Member> **VectorBaseCut**< Row, Member > & **Bool** (Member Row::*p_mem, const char *name_=”?”) const

  Returns a **BaseCut** *that returns a data member of T.*

- template<class Member> **VectorBaseCutFun**< Row, Member > & **Bool** (Member(Row::*p_mem)() const, const char *name_=”?”) const

  Returns a **BaseCut** *that returns the result of a member method of T.*

- template<class Member, class Arg> **VectorBaseCutFunArg**< Row, Member, Arg > & **Bool** (Member(Row::*p_mem)(Arg) const, Arg arg_, const char *name_=”?”) const

  Returns a **BaseCut** *that returns the result of a member method of T with an argument.*

Protected Member Functions

- virtual ~**VectorIterator** ()

  Protected destructor ensures creation on the heap.

Protected Attributes

- const std::vector< Row > & **theVector**

  The vector.

- unsigned int **index**

  The iterator index.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 template<class Row> VectorIterator< Row >::VectorIterator (const std::vector< Row > & *theVector_*, const char * *name_* = "?") [inline, explicit]

Constructor.

Parameters:

theVector_ the vector

name_ The variable name

Definition at line 536 of file VectorIterator.h.

References NamedO::getName(), VectorIterator< Row >::index, and VectorIterator< Row >::theVector.

6.110.3 Member Function Documentation

6.110.3.1 template<class Row> template<class Member> VectorFloatFun<Row, Member>& VectorIterator< Row >::Float (Member Row::* *p_mem*, const char * *name_* = "?") const [inline]

Returns a **FloatFun** that returns a data member of T.

class Member (Type of Member) must be a type that can be converted to a float, such as float, double, or int.

Definition at line 575 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.2 template<class Row> template<class Member> VectorFloatFunFun<Row, Member>& VectorIterator< Row >::Float (Member(Row::* *p_mem*)() const, const char * *name_* = "?") const [inline]

Returns a **FloatFun** that returns the result of a member method of T.

class Member (Type of Member) must be a type that can be converted to a float, such as float, double, or int. The access method itself must be declared as Member foobar() const; (the const is important!).

Definition at line 590 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.3 template<class Row> template<class Member, class Arg> VectorFloatFunFunArg<Row, Member, Arg>& VectorIterator< Row >::Float (Member(Row::* *p_mem*)(Arg) const, Arg *arg_*, const char * *name_* = "?") const [inline]

Returns a **FloatFun** that returns the result of a member method of T that takes an argument.

class Member (Type of Member) must be a type that can be converted to a float, such as float, double, or int. The access method itself must be declared as Member foobar(Arg) const; (the const is important!), where A is the type of the argument. A must have a copy constructor!

Definition at line 606 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.4 template<class Row> template<class Member> VectorIntFun<Row, Member>& VectorIterator< Row >::Int (Member Row::* *p_mem*, const char * *name_* = "?") const [inline]

Returns a **IntFun** that returns a data member of T.

class Member (Type of Member) must be a type that can be converted to a int, such as float, double, or int.

Definition at line 619 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.5 template<class Row> template<class Member> VectorIntFunFun<Row, Member>& VectorIterator< Row >::Int (Member(Row::* *p_mem*)() const, const char * *name_* = "?") const [inline]

Returns a **IntFun** that returns the result of a member method of T.

class Member (Type of Member) must be a type that can be converted to a int, such as float, double, or int. The access method itself must be declared as Member foobar() const; (the const is important!).

Definition at line 634 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.6 template<class Row> template<class Member, class Arg> VectorIntFunFunArg<Row, Member, Arg>& VectorIterator< Row >::Int (Member(Row::* *p_mem*)(Arg) const, Arg *arg_*, const char * *name_* = "?") const [inline]

Returns a **IntFun** that returns the result of a member method of T that takes an argument.

class Member (Type of Member) must be a type that can be converted to a int, such as float, double, or int. The access method itself must be declared as Member foobar(Arg) const; (the const is important!), where A is the type of the argument. A must have a copy constructor!

Definition at line 650 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.7 template<class Row> template<class Member> VectorBaseCut<Row, Member>& VectorIterator< Row >::Bool (Member Row::* *p_mem*, const char * *name_* = "?") const [inline]

Returns a **BaseCut** that returns a data member of T.

class Member (Type of Member) must be a type that can be converted to a bool, such as bool, float, double, or int.

Definition at line 663 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.8 template<class Row> template<class Member> VectorBaseCutFun<Row, Member>& VectorIterator< Row >::Bool (Member(Row::* *p_mem*)() const, const char * *name_* = "?") const [inline]

Returns a **BaseCut** that returns the result of a member method of T.

class Member (Type of Member) must be a type that can be converted to a bool, such as bool, float, double, or int. The access method itself must be declared as Member foobar() const; (the const is important!).

Definition at line 678 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

6.110.3.9 template<class Row> template<class Member, class Arg> VectorBaseCutFunArg<Row, Member, Arg>& VectorIterator< Row >::Bool (Member(Row::* *p_mem*)(Arg) const, Arg *arg_*, const char * *name_* = "?") const [inline]

Returns a **BaseCut** that returns the result of a member method of T with an argument.

class Member (Type of Member) must be a type that can be converted to a bool, such as bool, float, double, or int. The access method itself must be declared as Member foobar(Arg) const; (the const is important!)., where A is the type of the argument. A must have a copy constructor!

Definition at line 693 of file VectorIterator.h.

References VectorIterator< Row >::theVector.

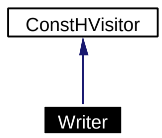
The documentation for this class was generated from the following file:

- **VectorIterator.h**

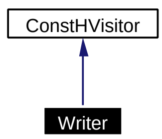
6.111 Writer Class Reference

`#include <HVisitors.h>`

Inheritance diagram for Writer:



Collaboration diagram for Writer:



6.111.1 Detailed Description

An **HVisitor** that writes all TObject objects into an (already open) root file.

Definition at line 29 of file HVisitors.h.

Public Member Functions

- virtual void **visit** (**RegO** &h) const
  The visit function.

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

- **HVisitors.h**

Chapter 7

SFH File Documentation

7.1 BaseCut.C File Reference

7.1.1 Detailed Description

Implements stuff related to class **BaseCut**.

Changelog:

- 22.7.04 BL: Taken out of **BaseCut.h**

CVS Log messages:

- **Log**
  BaseCut.C,v
- Revision 1.7 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **BaseCut.C**.

`#include "BaseCut.h"`

`#include "BaseCutHelpers.h"`

`#include "FillIterator.h"`

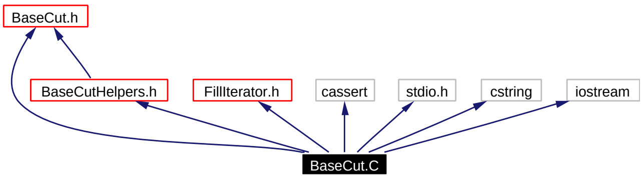
`#include <cassert>`

`#include <stdio.h>`

`#include <cstring>`

`#include <iostream>`

Include dependency graph for BaseCut.C:



Functions

- **BaseCut** & **cached** (const **ROListPoR** &rol, const **BaseCutPoR** &cut, const **FillIteratorPoR** &iter)
- **BaseCut** & **operator &&** (const **BaseCut** &lhs_, const **BaseCut** &rhs_)
- **BaseCut** & **operator**|| (const **BaseCut** &lhs_, const **BaseCut** &rhs_)
- **BaseCut** & **operator!** (const **BaseCut** &rhs_)
- **FloatFun** & **Float** (const **BaseCut** &arg)
- **IntFun** & **Int** (const **BaseCut** &arg)

7.2 BaseCut.h File Reference

7.2.1 Detailed Description

Declares logical operators and helper classes for class **BaseCut**.

Changelog:

- 30.6.04 BL: Added cuts that are defined by comparison operators between floatfun objects
- 15.7.04 BL Added conversions to **FloatFun** and **IntFun**
- 22.7.04 BL: Added method getName()
- 22.7.04 BL: Introduced **BaseCutBase.h** and **BaseCut.C**
- 31.5.05 BL: New class **ConstBaseCut**
- 2.6.05 BL: Changed all classes to new naming scheme (name in **NamedO** base class)
- 4.6.05 BL: Moved many classes to **BaseCutHelpers.h**, decoupled **BaseCut.h** from many other header files

CVS Log messages:

- **Log**
  BaseCut.h,v
- Revision 1.18 2005/11/22 09:00:16 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

Definition in file **BaseCut.h**.

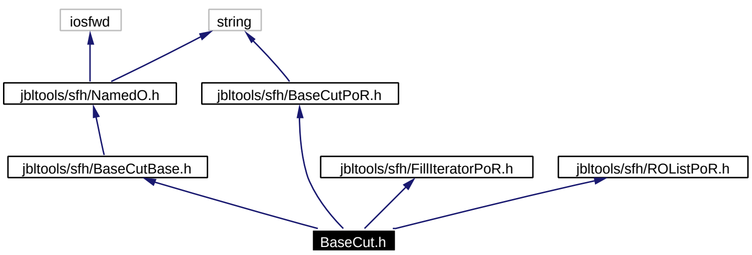
`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

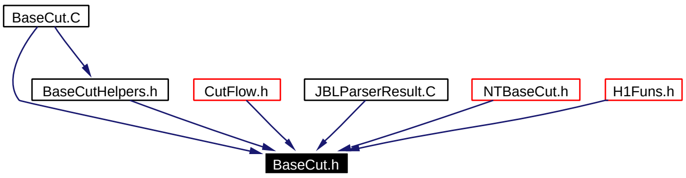
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/ROListPoR.h"`

Include dependency graph for BaseCut.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ConstBaseCut**

  A **IntFun** *that returns a constant value.*

Variables

- <<<<<<< **BaseCut**.hclassFloatBaseCut:publicFloatFun{public:FloatBaseCut(constBaseCut &theBaseCut_);virtualfloatoperator()() const{returntheBaseCut();}virtualconstFillIterator ∗getIterator() const{returntheBaseCut.getIterator();}virtualconstchar ∗getName() const{returnname;}protected:virtual∼FloatBaseCut(){delete[]name;name=0;}constBaseCut &theBaseCut;char ∗name;private:FloatBaseCut(constFloatBaseCut &rhs);FloatBaseCut &operator=(constFloatBaseCut &rhs);};classIntBaseCut:publicIntFun{public:IntBaseCut(constBaseCut &theBaseCut_);virtualintoperator()() const{returntheBaseCut();}virtualconstFillIterator ∗getIterator() const{returntheBaseCut.getIterator();}virtualconstchar ∗getName() const{returnname;}protected:virtual∼IntBaseCut(){delete[]name;name=0;}constBaseCut &theBaseCut;char ∗name;private:IntBaseCut(constIntBaseCut &rhs);IntBaseCut &operator=(constIntBaseCut &rhs);};classTwoCutCompound:publicBaseCut{public:TwoCutCompound(constBaseCut &lhs_, constBaseCut &rhs_, constchar ∗format="TwoCutCompound(%s,%s)");virtualconstFillIterator ∗getIterator() const;virtualconstchar ∗getName() const{returnname;}protected:virtual∼TwoCutCompound(){delete[]name;name=0;}constBaseCut &lhs;constBaseCut &rhs;char ∗name;private:TwoCutCompound(constTwoCutCompound &rhs);TwoCutCompound &operator=(constTwoCutCompound &rhs);};classAndOfTwoCuts:publicTwoCutCompound{public:AndOfTwoCuts(constBaseCut &lhs_, constBaseCut &rhs_):TwoCutCompound(lhs_, rhs_,"(%s)&&(%s)"){};virtualbooloperator()() const{returnlhs()&&rhs();};protected:virtual∼AndOfTwoCuts(){};};classOrOfTwoCuts:publicTwoCutCompound{public:OrOfTwoCuts(constBaseCut &lhs_, constBaseCut &rhs_):TwoCutCompound(lhs_, rhs_,"(%s)||(%s)"){};virtualbooloperator()() const{returnlhs()||rhs();}protected:virtual∼OrOfTwoCuts(){};};classOneCutCompound:publicBaseCut{public:OneCutCompound(constBaseCut &arg_, constchar ∗format="OneCutCompound(%s)");virtualconstFillIterator ∗getIterator() const{returnarg.getIterator();}virtualconstchar ∗getName() const{returnname;}protected:virtual∼OneCutCompound(){delete[]name;name=0;}constBaseCut &arg;char ∗name;private:OneCutCompound(constOneCutCompound &rhs);OneCutCompound &operator=(constOneCutCompound &rhs);};classNotOfACut:publicOneCutCompound{public:NotOfACut(constBaseCut &arg_):OneCutCompound(arg_,"!(%s)"){}virtualbooloperator()() const{return!arg();}protected:virtual∼NotOfACut(){};};classCachedBaseCut:publicBaseCut, publicCachedO{public:CachedBaseCut(constROListPoR &rol, constBaseCut &arg_);virtualbooloperator()() const;virtualvoidinvalidateCache(){cacheValid=false;}virtualconstchar ∗getName()

const{returnname;}protected:virtual∼CachedBaseCut(){delete[ ]name;name=0;}constBaseCut &arg;mutableboolcacheValid;mutableboolcachedValue;char ∗name;private:CachedBaseCut(constCachedBaseCut &rhs);CachedBaseCut &operator=(constCachedBaseCut &rhs);};classCachedBaseCutIt:publicBaseCut, publicCachedO{public:CachedBaseCutIt(constROListPoR &rol, constBaseCut &arg_, constFillIterator &iter_);virtualbooloperator()() const;virtualconstFillIterator ∗getIterator() const{return &iter;}virtualvoidinvalidateCache(){cacheValid.resize(0);cachedValues.resize(0);}virtualconstchar ∗getName() const{returnCachedBaseCutIt::name;}protected:voidgrowCache(unsignedints) const;virtual∼CachedBaseCutIt(){delete[ ]name;name=0;}constBaseCut &arg;constFillIterator &iter;mutablestd::vector< bool >cacheValid;mutablestd::vector< bool >cachedValues;char ∗name;private:CachedBaseCutIt(constCachedBaseCutIt &rhs);CachedBaseCutIt &operator=(constCachedBaseCutIt &rhs);};=======>>>>>>> **BaseCut** & **operator&&** (const **BaseCut** &lhs_, const **BaseCut** &rhs_)

"And" of two **BaseCut** *objects*

7.3 BaseCutBase.h File Reference

7.3.1 Detailed Description

Declares class **BaseCut**.

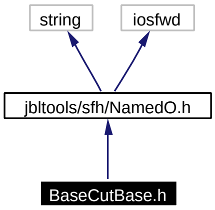
Changelog:

- 22.7.04 BL: Taken out of **BaseCut.h**

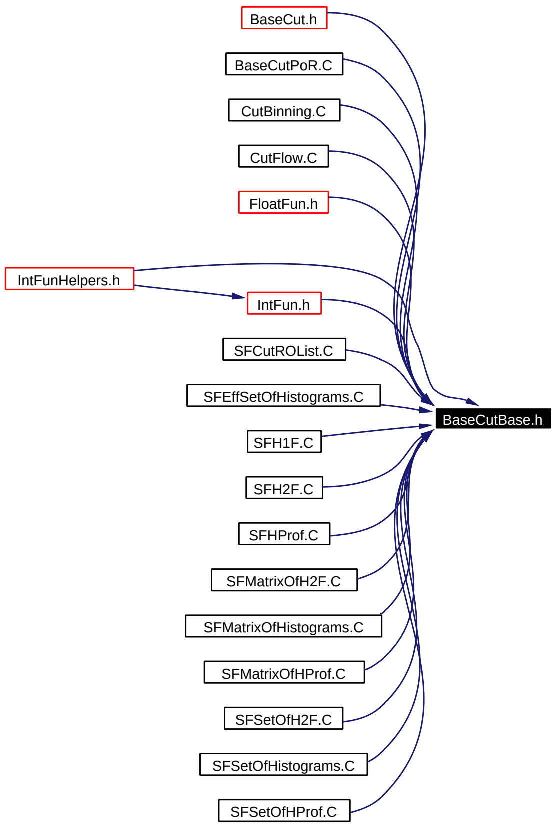
Definition in file **BaseCutBase.h**.

`#include "jbltools/sfh/NamedO.h"`

Include dependency graph for BaseCutBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **BaseCut**

  Abstract base class for function objects doing cuts.

7.4 BaseCutHelpers.h File Reference

7.4.1 Detailed Description

Declares various helper classes for operators between **BaseCut** objects.

Changelog:

- 4.6.05 BL: First version (taken out of **BaseCut.h**)
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

Definition in file **BaseCutHelpers.h**.

`#include "jbltools/sfh/BaseCut.h"`

`#include "jbltools/sfh/IntFunBase.h"`

`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/ROListPoR.h"`

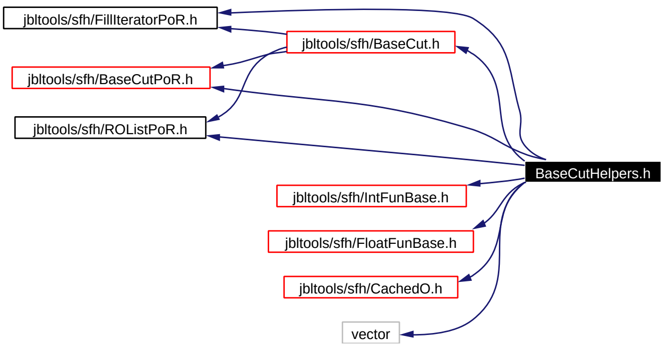
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

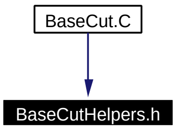
`#include "jbltools/sfh/CachedO.h"`

`#include <vector>`

Include dependency graph for BaseCutHelpers.h:



This graph shows which files directly or indirectly include this file:



7.5 BaseCutPoR.C File Reference

7.5.1 Detailed Description

Implements class **BaseCutPoR**.

Changelog:

- 29.5.05 BL: First version

CVS Log messages:

- **Log**
  BaseCutPoR.C,v
- Revision 1.4 2005/07/08 15:01:34 blist
- Added identity string to all .C files

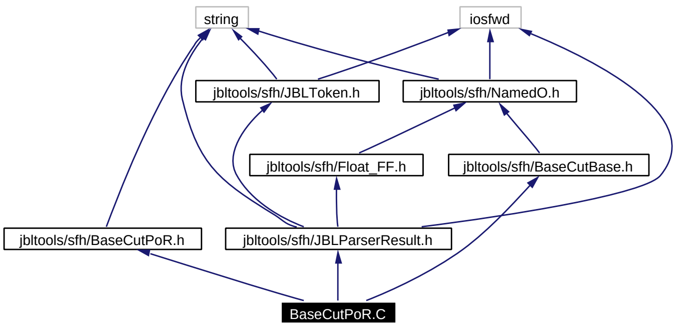
Definition in file **BaseCutPoR.C**.

`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/JBLParserResult.h"`

Include dependency graph for BaseCutPoR.C:



7.6 BaseCutPoR.h File Reference

7.6.1 Detailed Description

Declares classes **BaseCutPoR** and **BaseCutPoRConst**.

Author: Benno List, Jenny Böhme

Date

2005/06/29 14:40:18

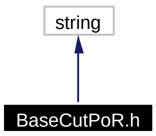
Author

blist

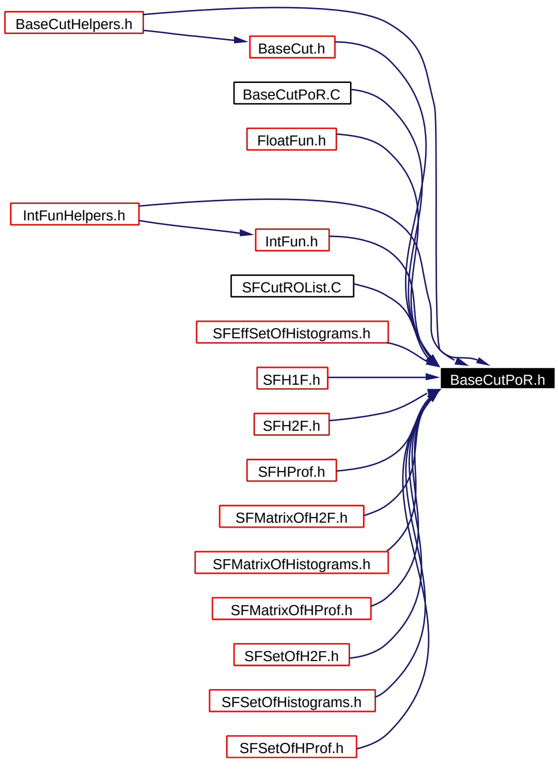
Definition in file **BaseCutPoR.h**.

`#include <string>`

Include dependency graph for BaseCutPoR.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **BaseCutPoR**

  BaseCut *Pointer or Reference.*

- class **BaseCutPoRConst**

  const **BaseCut** *Pointer or Reference*

7.7 Binning.C File Reference

7.7.1 Detailed Description

Implements class Binning.

Author: Benno List, Jenny Böhme

Date

2005/09/29 08:56:40

Author

boehmej

Changelog:

- 4.6.04 BL: First version
- 22.6.04 BL: Use doubles
- 23.8.05 BL: Added constructor from TAxis object, use Axis_t

CVS Log messages:

- **Log**
  Binning.C,v
- Revision 1.9 2005/09/29 08:56:40 boehmej
- bugfix in **Binning.C**

- Revision 1.8 2005/08/23 12:05:43 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

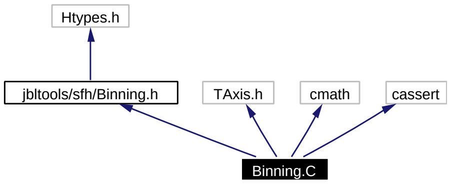
- Revision 1.7 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**

- Revision 1.6 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **Binning.C**.

```
#include "jbltools/sfh/Binning.h"
#include <TAxis.h>
#include <cmath>
#include <cassert>
```

Include dependency graph for Binning.C:


Htypes.h
jbltools/sfh/Binning.h
TAxis.h
cmath
cassert
Binning.C

7.8 Binning.h File Reference

7.8.1 Detailed Description

Declares class Binning.

CVS Log messages:

- **Log**
  Binning.h,v

- Revision 1.7 2005/08/23 12:05:43 blist

- Data members of Binning are now private, bin edges are of type Axis_t

- New constructor for Binning from TAxis

- Self-Filling histograms etc. now use Float_FF instead of float (or double)

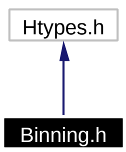
- Revision 1.6 2005/07/13 11:35:50 blist

- Alphanumeric labels possible for Binning, implemented for **CutBinning**

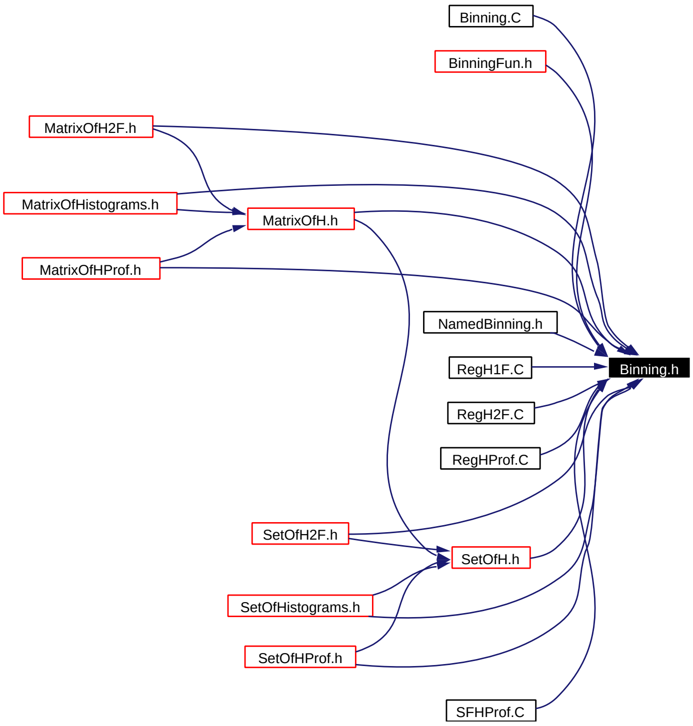
Definition in file **Binning.h**.

`#include <Htypes.h>`

Include dependency graph for Binning.h:



This graph shows which files directly or indirectly include this file:


Binning.C
BinningFun.h
MatrixOfH2F.h
MatrixOfHistograms.h
MatrixOfH.h
MatrixOfHProf.h
NamedBinning.h
RegH1F.C
Binning.h
RegH2F.C
RegHProf.C
SetOfH2F.h
SetOfH.h
SetOfHistograms.h
SetOfHProf.h
SFHProf.C

7.9 BinningFun.h File Reference

7.9.1 Detailed Description

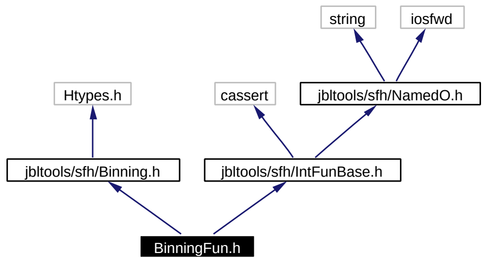
Declares class **BinningFun**.

Definition in file **BinningFun.h**.

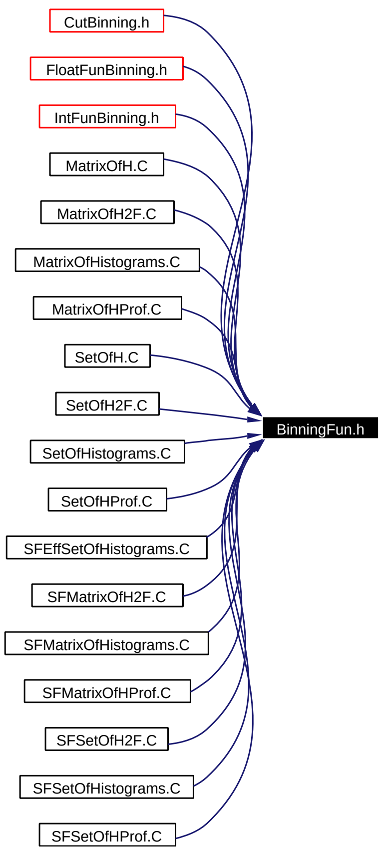
`#include "jbltools/sfh/Binning.h"`

`#include "jbltools/sfh/IntFunBase.h"`

Include dependency graph for BinningFun.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **BinningFun**

  Abstract base class for function objects that sort events into bins.

7.10 BinningFunPoR.h File Reference

7.10.1 Detailed Description

Declares classes **BinningFunPoR** and **BinningFunPoRConst**.

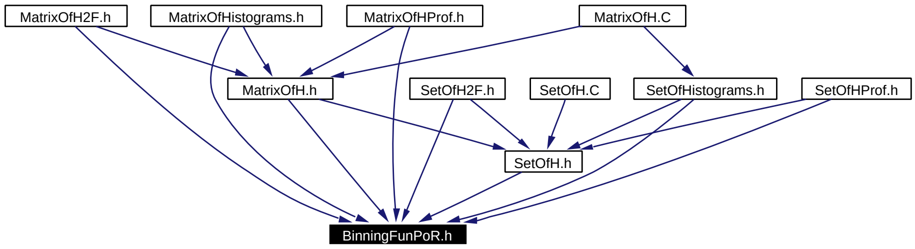
Author: Benno List, Jenny Böhme

Date
2004/06/29 13:03:38

Author
blist

Definition in file **BinningFunPoR.h**.

This graph shows which files directly or indirectly include this file:



Compounds

- class **BinningFunPoR**

  BinningFun *Pointer or Reference.*

- class **BinningFunPoRConst**

  const **BinningFun** *Pointer or Reference*

7.11 CachedO.C File Reference

7.11.1 Detailed Description

Implements class **CachedO**.

Author: Benno List, Jenny Böhme

Date
2005/07/08 15:01:34

Author
blist

Changelog:

- 21.6.06 BL First version

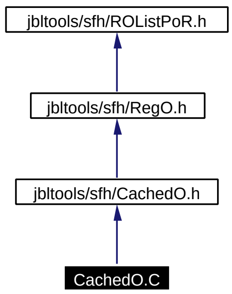
CVS Log messages:

- **Log**
  CachedO.C,v
- Revision 1.2 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **CachedO.C**.

```
#include "jbltools/sfh/CachedO.h"
```

Include dependency graph for CachedO.C:



7.12 CachedO.h File Reference

7.12.1 Detailed Description

Declares class **CachedO**.

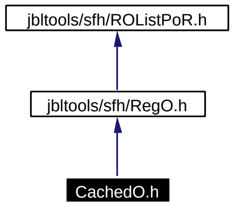
Changelog:

- 21.6.06 BL First version

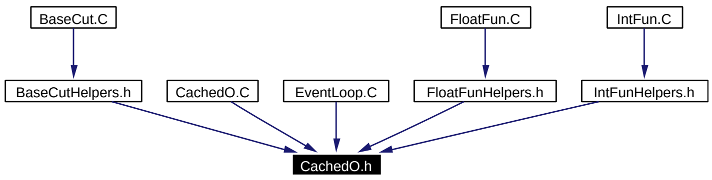
Definition in file **CachedO.h**.

`#include "jbltools/sfh/RegO.h"`

Include dependency graph for CachedO.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **CachedO**

  Mix-in class for cached, registered objects.

7.13 CutBinning.C File Reference

7.13.1 Detailed Description

Implements class **CutBinning**.

Changelog:

- 10.6.05 BL: First version
- 13.7.05 BL: Added alphanumeric Bin Labels

CVS Log messages:

- **Log**
  CutBinning.C,v
- Revision 1.4 2005/07/20 10:51:32 blist
- Added constructors with pointers to non-const **BaseCut** objects for **CutBinning**
- Revision 1.3 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**
- Revision 1.2 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **CutBinning.C**.

`#include "jbltools/sfh/CutBinning.h"`

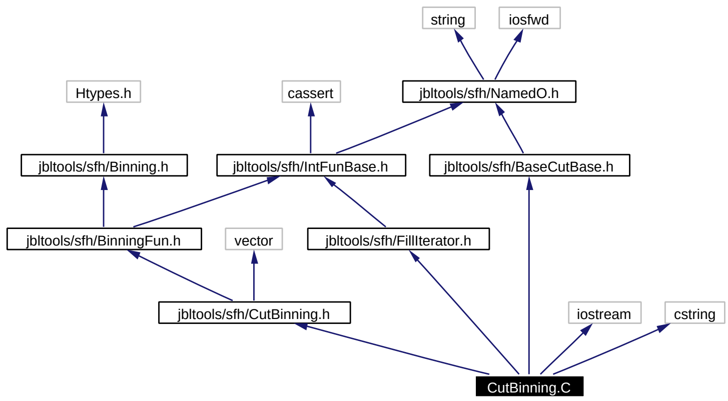
`#include "jbltools/sfh/FillIterator.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include <iostream>`

`#include <cstring>`

Include dependency graph for CutBinning.C:



7.14 CutBinning.h File Reference

7.14.1 Detailed Description

Declares class **CutBinning**.

Changelog:

- 10.6.05 BL: First version
- 13.7.05 BL: Added alphanumeric bin labels

CVS Log messages:

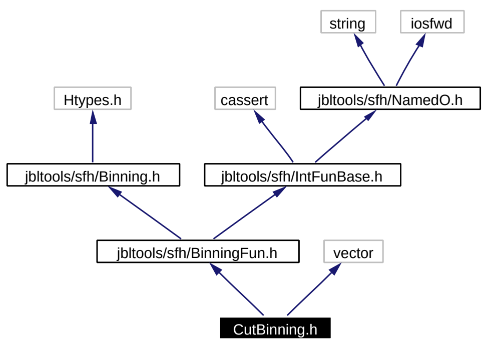
- **Log**
  CutBinning.h,v
- Revision 1.4 2005/07/20 10:51:32 blist
- Added constructors with pointers to non-const **BaseCut** objects for **CutBinning**
- Revision 1.3 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**

Definition in file **CutBinning.h**.

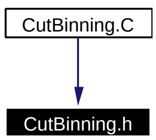
`#include "jbltools/sfh/BinningFun.h"`

`#include <vector>`

Include dependency graph for CutBinning.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **CutBinning**

  Class that generates a non-exclusive binning from an array of cuts.

7.15 CutFlow.C File Reference

7.15.1 Detailed Description

Implements class **CutFlow**.

Changelog:

- 13.6.05 BL: First version

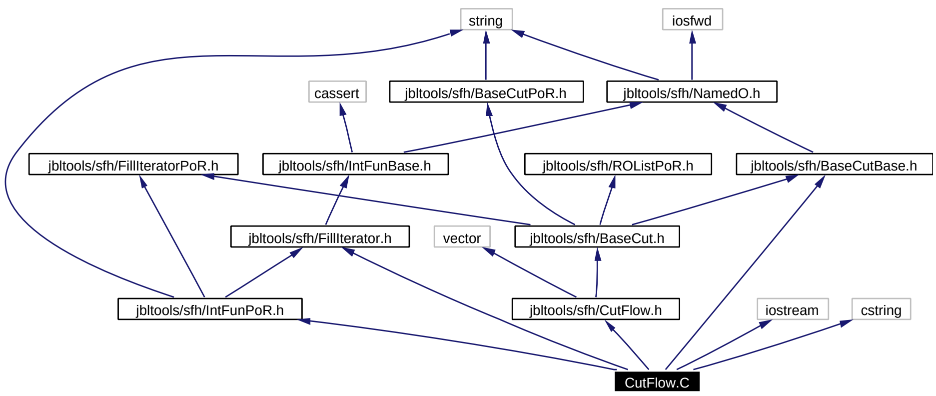
CVS Log messages:

- **Log**
  CutFlow.C,v
- Revision 1.2 2005/07/20 10:56:41 blist
- Added constructors for pointers to non-const **BaseCut** to **CutFlow**
- Revision 1.1 2005/07/13 10:51:12 blist
- New class **CutFlow**

Definition in file **CutFlow.C**.

```
#include "jbltools/sfh/CutFlow.h"
#include "jbltools/sfh/FillIterator.h"
#include "jbltools/sfh/IntFunPoR.h"
#include "jbltools/sfh/BaseCutBase.h"
#include <iostream>
#include <cstring>
```

Include dependency graph for CutFlow.C:



7.16 CutFlow.h File Reference

7.16.1 Detailed Description

Declares class **CutFlow**.

Changelog:

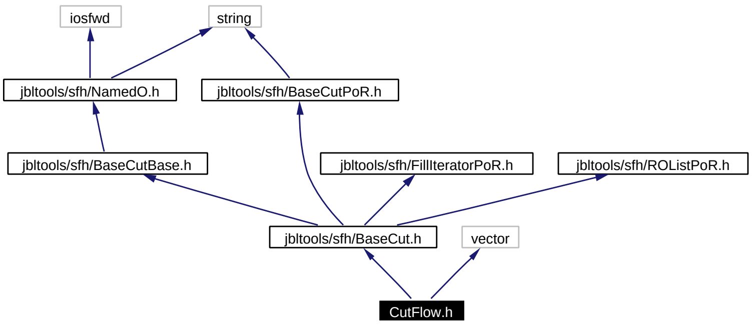
- 13.6.05 BL: First version

Definition in file **CutFlow.h**.

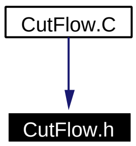
`#include "jbltools/sfh/BaseCut.h"`

`#include <vector>`

Include dependency graph for CutFlow.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **CutFlow**

  Class that returns the value of a cut depending on an iterator.

7.17 EventLoop.C File Reference

7.17.1 Detailed Description

Implements class **EventLoop**.

Author: Jenny Böhme, Benno List

Date
2005/09/29 08:56:40

Author
boehmej

Changelog:

- 4.6.04 BL: SFH not purely virtual anymore, so we can cross-cast from RegH∗ to SFH∗, allowing us to decouple **EventLoop::loop()** from the concrete SFH subclasses.
- 8.6.04 BL: Derive **EventLoop** from SFHList, not HList
- 21.6.04 BL: Invalidate all cached objects
- 29.6.04 BL: Add default implementation for output

CVS Log messages:

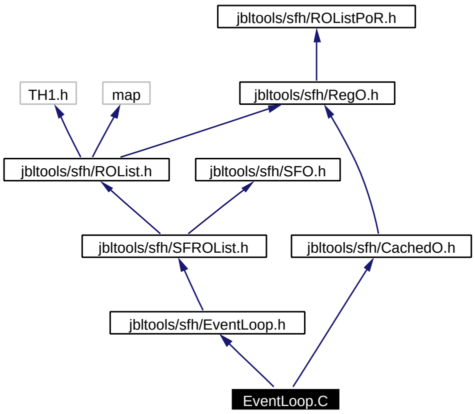
- **Log**
  EventLoop.C,v
- Revision 1.9 2005/09/29 08:56:40 boehmej
- bugfix in **Binning.C**
- Revision 1.8 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **EventLoop.C**.

`#include "jbltools/sfh/EventLoop.h"`

`#include "jbltools/sfh/CachedO.h"`

Include dependency graph for EventLoop.C:


jbltools/sfh/ROListPoR.h
TH1.h
map
jbltools/sfh/RegO.h
jbltools/sfh/ROList.h
jbltools/sfh/SFO.h
jbltools/sfh/SFROList.h
jbltools/sfh/CachedO.h
jbltools/sfh/EventLoop.h
EventLoop.C

7.18 EventLoop.h File Reference

7.18.1 Detailed Description

Declares class **EventLoop**.

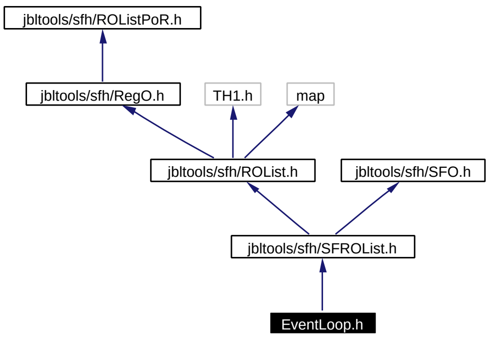
Changelog:

- 8.6.04 BL: Derive **EventLoop** from SFHList, not HList
- 10.6.04 BL: Derive **EventLoop** from **SFROList**, not SFHList
- 21.6.04 BL: Add list of cached objects
- 29.6.04 BL: Add documentation, add default implementation for output

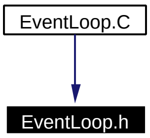
Definition in file **EventLoop.h**.

`#include "jbltools/sfh/SFROList.h"`

Include dependency graph for EventLoop.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **EventLoop**

  Base class for user event loops.

7.19 FillIterator.h File Reference

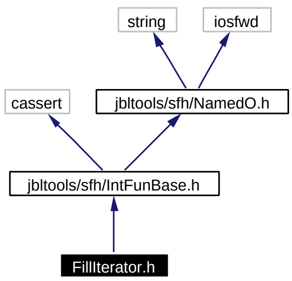
7.19.1 Detailed Description

Declares **FillIterator**.

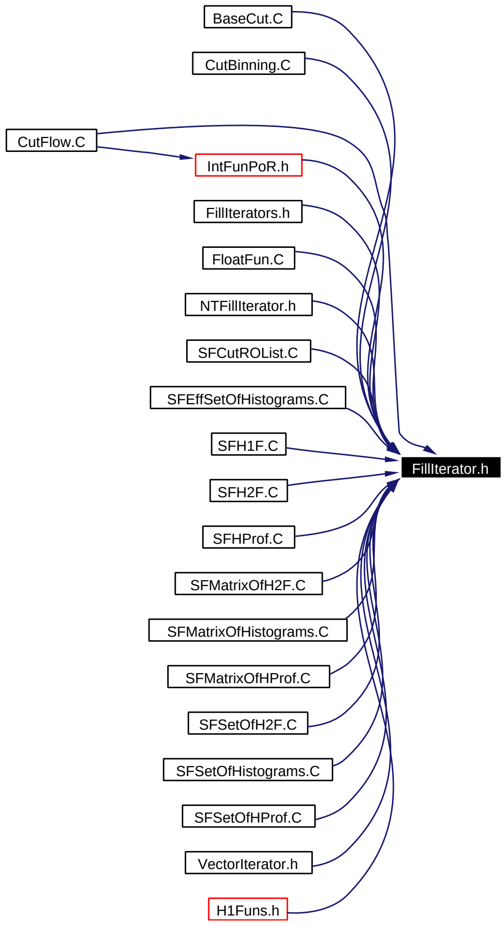
Definition in file **FillIterator.h**.

`#include "jbltools/sfh/IntFunBase.h"`

Include dependency graph for FillIterator.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **FillIterator**

  Abstract base class for iterators used to fill self-filling histograms.

7.20 FillIteratorPoR.h File Reference

7.20.1 Detailed Description

Declares classes **FillIteratorPoR** and **FillIteratorPoRConst**.

Author: Benno List, Jenny Böhme

Date

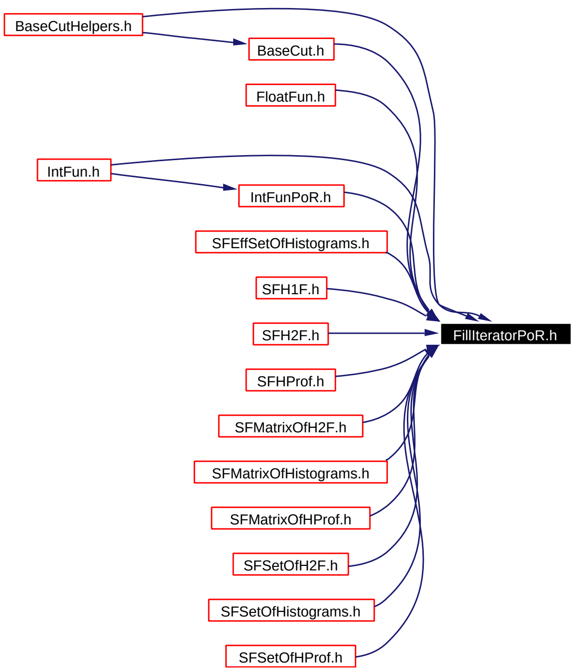
2004/06/29 13:03:38

Author

blist

Definition in file **FillIteratorPoR.h**.

This graph shows which files directly or indirectly include this file:



Compounds

- class **FillIteratorPoR**

  FillIterator *Pointer or Reference.*

- class **FillIteratorPoRConst**

  const **FillIterator** *Pointer or Reference*

7.21 FillIterators.h File Reference

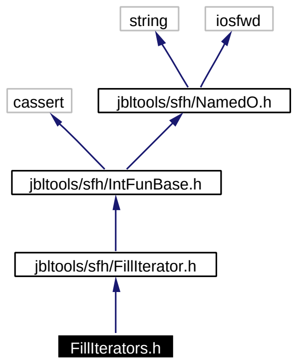
7.21.1 Detailed Description

Declares some **FillIterator** subclasses.

Definition in file **FillIterators.h**.

`#include "jbltools/sfh/FillIterator.h"`

Include dependency graph for FillIterators.h:



Compounds

- class **RangeIterator**

  FillIterator *that iterates over a constant range of values.*

7.22 Fitter.h File Reference

7.22.1 Detailed Description

Declares sclass **Fitter**.

Changelog:

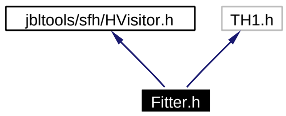
- 20.8.04 BL, JB: Add doxygen documentation

Definition in file **Fitter.h**.

`#include "jbltools/sfh/HVisitor.h"`

`#include <TH1.h>`

Include dependency graph for Fitter.h:



Compounds

- class **Fitter**

  An **HVisitor** *that fits TH1 histograms with a TF1 function.*

7.23 Float_FF.h File Reference

7.23.1 Detailed Description

Declares type Float_FF.

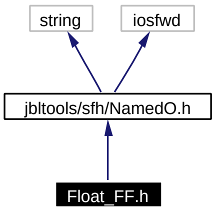
Changelog:

- 21.7.05 BL: First version

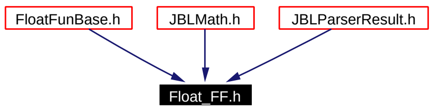
Definition in file **Float_FF.h**.

`#include "jbltools/sfh/NamedO.h"`

Include dependency graph for Float_FF.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef float **Float_FF**

7.24 FloatFun.C File Reference

7.24.1 Detailed Description

Implements stuff connected with **FloatFun** subclasses.

Changelog:

- 21.7.04 BL: First version
- 3.6.05 BL: Major overhaul
- 8.6.05 BL: Added **pow()**
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

CVS Log messages:

- **Log**
  FloatFun.C,v
- Revision 1.22 2005/11/22 09:00:17 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C

- Revision 1.21 2005/11/21 22:31:28 blist
- Random numbers are now OK

- Revision 1.20 2005/11/21 21:43:40 blist
- Reasonable random numbers, and test

- Revision 1.19 2005/11/21 19:58:58 blist
- New classes **GaussRandomFun**, **Poisson1Fun**, **Poisson2Fun**

- Revision 1.18 2005/11/21 18:30:35 blist
- New class **RandomFun**

- Revision 1.17 2005/07/21 11:20:12 blist
- Use Float_FF instead of float as result type of **FloatFun**

- Revision 1.16 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **FloatFun.C**.

```
#include "jbltools/sfh/FloatFun.h"
#include "jbltools/sfh/FloatFunHelpers.h"
#include "jbltools/sfh/IntFun.h"
#include "jbltools/sfh/FillIterator.h"
```

`#include <cmath>`

`#include <values.h>`

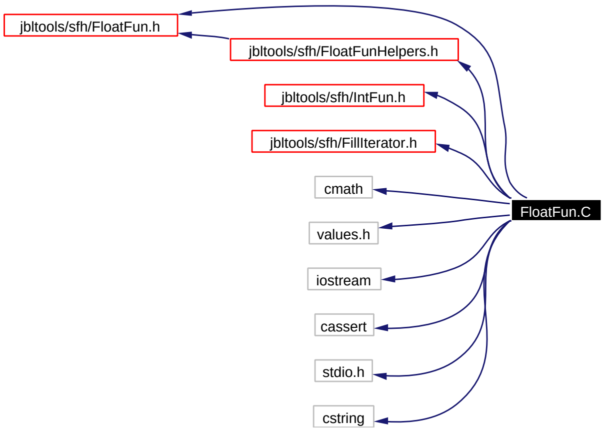
`#include <iostream>`

`#include <cassert>`

`#include <stdio.h>`

`#include <cstring>`

Include dependency graph for FloatFun.C:



Functions

- **FloatFun** & **abs** (const **FloatFun** &arg_)
- **FloatFun** & **acos** (const **FloatFun** &arg_)
- **FloatFun** & **asin** (const **FloatFun** &arg_)
- **FloatFun** & **atan** (const **FloatFun** &arg_)
- **FloatFun** & **ceil** (const **FloatFun** &arg_)
- **FloatFun** & **cos** (const **FloatFun** &arg_)
- **FloatFun** & **cosh** (const **FloatFun** &arg_)
- **FloatFun** & **exp** (const **FloatFun** &arg_)
- **FloatFun** & **fabs** (const **FloatFun** &arg_)
- **FloatFun** & **floor** (const **FloatFun** &arg_)
- **FloatFun** & **log** (const **FloatFun** &arg_)
- **FloatFun** & **log10** (const **FloatFun** &arg_)
- **FloatFun** & **sin** (const **FloatFun** &arg_)
- **FloatFun** & **sinh** (const **FloatFun** &arg_)
- **FloatFun** & **sqrt** (const **FloatFun** &arg_)
- **FloatFun** & **tan** (const **FloatFun** &arg_)
- **FloatFun** & **tanh** (const **FloatFun** &arg_)
- **FloatFun** & **square** (const **FloatFun** &arg_)
- **FloatFun** & **rad** (const **FloatFun** &arg_)
- **FloatFun** & **deg** (const **FloatFun** &arg_)
- **FloatFun** & **atan2** (const **FloatFun** &lhs, const **FloatFun** &rhs)
- **FloatFun** & **pow** (const **FloatFun** &arg1, const **FloatFun** &arg2)

- **FloatFun** & **cached** (const **ROListPoR** &rol, const **FloatFunPoR** &fun, const **FillIteratorPoR** &iter)
- **FloatFun** & **operator+** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator+** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFun** & **operator+** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator-** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator-** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFun** & **operator-** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator-** (const **FloatFun** &arg_)
- const **FloatFun** & **operator+** (const **FloatFun** &arg_)
- **FloatFun** & **operator+** (**FloatFun** &arg_)
- **FloatFun** & **operator** * (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator** * (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator** * (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFun** & **operator/** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator/** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFun** & **operator/** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator<** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator==** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator!=** (const **FloatFun** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<=** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>=** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator==** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator!=** (Float_FF lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator<=** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator>** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator>=** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator==** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator!=** (const **FloatFun** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator<** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator>=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator==** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator!=** (const **FloatFunComp** &lhs_, const **FloatFun** &rhs_)
- **FloatFunComp** & **operator<** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator<=** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator>** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator>=** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator==** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFunComp** & **operator!=** (const **FloatFunComp** &lhs_, Float_FF rhs_)
- **FloatFun** & **max** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)

- **FloatFun** & **min** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)
- **FloatFun** & **sum** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)
- **FloatFun** & **average** (const **FloatFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)

7.25 FloatFun.h File Reference

7.25.1 Detailed Description

Defines arithmetic operators between and functions of **FloatFun** objects.

Changelog:

- 29.6.04 BL: Added classes OneFunCompound, ScalarProductFloatFun, SqrtFloatFun, SinFloatFun, CosFloatFun
- 30.6.04 BL: Declaration of **FloatFun** now in separate file **FloatFunBase.h** (the name **FloatFun.h** for this header file was kept for compatibility reasons). Renamed SqrFloatFun->SquareFloatFun, added Atan2Fun
- 8.7.04 BL: Added division
- 3.5.05 BL: Added exp and log, bug in operator> fixed
- 31.5.05 BL: New mechanism for name generation in OneFunCompound and TwoFunCompound
- 2.6.05 BL: Removed various helper classes for sin, cos etc. (replaced by template stuff in **FloatFun.C**)
- 3.6.05 BL: Introduced new header **FloatFunHelpers.h**, which is only needed inFloatFun.C, adapted all classes in **FloatFun.h** to new name scheme
- 4.6.05 BL: Decoupled **BaseCut.h** from many other header files
- 8.6.05 BL: Added **pow()**
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**
- 21.11.05 BL: Added classes **RandomFun**, **GaussRandomFun**
- 22.11.05 BL: Moved **RandomFun**, **GaussRandomFun** to **RandomFuns.h**

CVS Log messages:

- **Log**
  FloatFun.h,v
- Revision 1.33 2005/11/22 09:00:17 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

Definition in file **FloatFun.h**.

`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/JBLMath.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

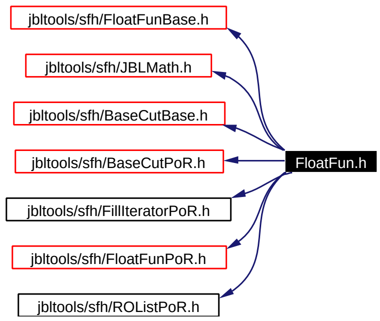
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

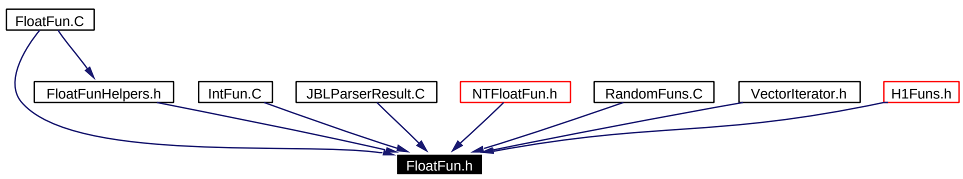
`#include "jbltools/sfh/FloatFunPoR.h"`

`#include "jbltools/sfh/ROListPoR.h"`

Include dependency graph for FloatFun.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ConstFun**

  A **FloatFun** *that returns a constant value.*

- class **FloatFunComp**

  Abstract base class for comparisons between **FloatFun** *objects.*

Functions

- const **FloatFun** & **operator+** (const **FloatFun** &arg_)

Variables

- <<<<<<< **FloatFun**.hclassFloatFloatFunLessEqual:publicFloatFloat-FunComp{public:FloatFloatFunLessEqual(floatlhs_, constFloatFun &rhs_-):FloatFloatFunComp(lhs_, rhs_,”%8f<=(%s)”){}virtualbooloperator()() const{returnlhs<=rhs();}protected:virtual∼FloatFloatFunLessEqual(){}};class-FloatFloatFunGreater:publicFloatFloatFunComp{public:FloatFloatFun-Greater(floatlhs_, constFloatFun &rhs_):FloatFloatFunComp(lhs_, rhs_,”%8f >(%s)”){}virtualbooloperator()() const{returnlhs >rhs();}protected:virtual∼Float-FloatFunGreater(){}};classFloatFloatFunGreaterEqual:publicFloatFloatFun-Comp{public:FloatFloatFunGreaterEqual(floatlhs_, constFloatFun &rhs_):Float-FloatFunComp(lhs_, rhs_,”%8f >=(%s)”){}virtualbooloperator()() const{returnlhs

>=rhs();}protected:virtual~FloatFloatFunGreaterEqual(){}};classFloatFloatFun-
Equal:publicFloatFloatFunComp{public:FloatFloatFunEqual(floatlhs_, constFloat-
Fun &rhs_):FloatFloatFunComp(lhs_, rhs_,"%8f==(%s)"){}virtualbooloperator()()
const{returnlhs==rhs();}protected:virtual~FloatFloatFunEqual(){}};classFloatFloat-
FunNotEqual:publicFloatFloatFunComp{public:FloatFloatFunNotEqual(floatlhs_, const-
FloatFun &rhs_):FloatFloatFunComp(lhs_, rhs_,"%8f!=(%s)"){}virtualbooloperator()()
const{returnlhs!=rhs();}protected:virtual~FloatFloatFunNotEqual(){}};classFloatFun-
FloatComp:publicFloatFunComp{public:FloatFunFloatComp(constFloatFun &lhs_-
, floatrhs_, constchar *format="FloatFunFloatComp(%s, float)");virtualfloatlhs-
Value() const{returnlhs();}virtualfloatrhsValue() const{returnrhs;}virtualconstFill-
Iterator *getIterator() const{returnlhs.getIterator();}virtualconstchar *getName()
const{returnname;}protected:virtual~FloatFunFloatComp(){delete[]name;name=0;}const-
FloatFun &lhs;floatrhs;char *name;private:FloatFunFloatComp(constFloatFunFloatComp
&source);FloatFunFloatComp &operator=(constFloatFunFloatComp &source);};class-
FloatFunFloatLess:publicFloatFunFloatComp{public:FloatFunFloatLess(constFloatFun
&lhs_, floatrhs_):FloatFunFloatComp(lhs_, rhs_,"(%s)<%8f"){}virtualbooloperator()()
const{returnlhs()< rhs;}protected:virtual~FloatFunFloatLess(){}};classFloatFunFloat-
LessEqual:publicFloatFunFloatComp{public:FloatFunFloatLessEqual(constFloatFun
&lhs_, floatrhs_):FloatFunFloatComp(lhs_, rhs_,"(%s)<=%8f"){}virtualbooloperator()()
const{returnlhs()<=rhs;}protected:virtual~FloatFunFloatLessEqual(){}};class-
FloatFunFloatGreater:publicFloatFunFloatComp{public:FloatFunFloat-
Greater(constFloatFun &lhs_, floatrhs_):FloatFunFloatComp(lhs_, rhs_-
,"(%s)>%8f"){}virtualbooloperator()() const{returnlhs()>rhs;}protected:virtual~Float-
FunFloatGreater(){}};classFloatFunFloatGreaterEqual:publicFloatFunFloat-
Comp{public:FloatFunFloatGreaterEqual(constFloatFun &lhs_, floatrhs_-
):FloatFunFloatComp(lhs_, rhs_,"(%s)>=%8f"){}virtualbooloperator()()
const{returnlhs()>=rhs;}protected:virtual~FloatFunFloatGreaterEqual(){}};classFloat-
FunFloatEqual:publicFloatFunFloatComp{public:FloatFunFloatEqual(constFloatFun
&lhs_, floatrhs_):FloatFunFloatComp(lhs_, rhs_,"(%s)==%8f"){}virtualbooloperator()()
const{returnlhs()==rhs;}protected:virtual~FloatFunFloatEqual(){}};classFloatFun-
FloatNotEqual:publicFloatFunFloatComp{public:FloatFunFloatNotEqual(constFloatFun
&lhs_, floatrhs_):FloatFunFloatComp(lhs_, rhs_,"(%s)!=%8f"){}virtualbooloperator()()
const{returnlhs()!=rhs;}protected:virtual~FloatFunFloatNotEqual(){}};classCollective-
FloatFun:publicFloatFun{public:CollectiveFloatFun(constFloatFunPoR &fun_=0,
constBaseCutPoR &cut_=0, constFillIteratorPoR &intiter_=0, constFillIteratorPoR
&extiter_=0, constchar *format="(CollectiveFloatFun(%s,%s,%s,%s)");virtualconst-
FillIterator *getIterator() const{returnextiter;}virtualconstchar
*getName() const{returnname;}protected:virtual~CollectiveFloat-
Fun(){delete[]name;name=0;}voidcheckIterators();voidmakeName(constchar
*format);constFloatFun *fun;constBaseCut *cut;**FillIterator** *intiter;constFill-
Iterator *extiter;char *name;private:CollectiveFloatFun(constCollectiveFloatFun
&source);CollectiveFloatFun &operator=(constCollectiveFloatFun &source);};class-
MaxFloatFun:publicCollectiveFloatFun{public:MaxFloatFun(constFloatFunPoR
&fun_=0, constBaseCutPoR &cut_=0, constFillIteratorPoR &intiter_=0, const-
FillIteratorPoR &extiter_=0):CollectiveFloatFun(fun_, cut_, intiter_, extiter_-
,"max(%s,%s,%s,%s)"){}virtualfloatoperator()() const;protected:virtual~MaxFloat-
Fun(){}};classMinFloatFun:publicCollectiveFloatFun{public:MinFloatFun(constFloat-
FunPoR &fun_=0, constBaseCutPoR &cut_=0, constFillIteratorPoR &intiter_=0,
constFillIteratorPoR &extiter_=0):CollectiveFloatFun(fun_, cut_, intiter_, extiter_-
,"min(%s,%s,%s,%s)"){}virtualfloatoperator()() const;protected:virtual~MinFloat-
Fun(){}};classSumFloatFun:publicCollectiveFloatFun{public:SumFloatFun(constFloat-
FunPoR &fun_=0, constBaseCutPoR &cut_=0, constFillIteratorPoR &intiter_=0,
constFillIteratorPoR &extiter_=0):CollectiveFloatFun(fun_, cut_, intiter_, extiter_-
,"sum(%s,%s,%s,%s)"){}virtualfloatoperator()() const;protected:virtual~SumFloat-

Fun(){}};classAverageFloatFun:publicCollectiveFloatFun{public:AverageFloatFun(constFloatFunPoR &fun_=0, constBaseCutPoR &cut_=0, constFillIteratorPoR &intiter_=0, constFillIteratorPoR &extiter_=0):CollectiveFloatFun(fun_, cut_, intiter_, extiter_,”average(%s,%s,%s,%s)”){}virtualfloatoperator()() const;protected:virtual∼AverageFloatFun(){}};classCachedFloatFun:publicFloatFun, publicCachedO{public:CachedFloatFun(constROListPoR &rol, constFloatFun &arg_);virtualfloatoperator()() const;virtualvoidinvalidateCache(){cacheValid=false;}virtualconstchar ∗getName() const{returnCachedFloatFun::name;}protected:virtual∼CachedFloatFun(){delete[ ]name;name=0;}constFloatFun &arg;mutableboolcacheValid;mutablefloatcachedValue;char ∗name;private:CachedFloatFun(constCachedFloatFun &rhs);CachedFloatFun &operator=(constCachedFloatFun &rhs);};classCachedFloatFunIt:publicFloatFun, publicCachedO{public:CachedFloatFunIt(constROListPoR &rol, constFloatFun &arg_, constFillIterator &iter_);virtualfloatoperator()() const;virtualconstFillIterator ∗getIterator() const{return &iter;}virtualvoidinvalidateCache(){cacheValid.resize(0);cachedValues.resize(0);}virtualconstchar ∗getName() const{returnname;}protected:voidgrowCache(unsignedints) const;virtual∼CachedFloatFunIt(){delete[ ]name;name=0;}constFloatFun &arg;constFillIterator &iter;mutablestd::vector< bool >cacheValid;mutablestd::vector< float >cachedValues;char ∗name;private:CachedFloatFunIt(constCachedFloatFunIt &rhs);CachedFloatFunIt &operator=(constCachedFloatFunIt &rhs);};inlineFloatFun &operator+(constFloatFun &lhs_, constFloatFun &rhs_){return ∗newSumOfTwoFuns(lhs_, rhs_);}inlineFloatFun &operator+(constFloatFun &lhs_, floatrhs_){return ∗newScalarSumFloatFun(rhs_, lhs_);}inlineFloatFun &operator+(floatlhs_, constFloatFun &rhs_){return ∗newScalarSumFloatFun(lhs_, rhs_);}inlineFloatFun &operator-(constFloatFun &lhs_, constFloatFun &rhs_){return ∗newDiffOfTwoFuns(lhs_, rhs_);}inlineFloatFun &operator-(constFloatFun &lhs_, floatrhs_){return ∗newScalarSumFloatFun(-rhs_, lhs_, 1, NamedO::str(’(’)+lhs_.getName()+”)-”+NamedO::str(rhs_));}inlineFloatFun &operator-(floatlhs_, constFloatFun &rhs_){return ∗newScalarSumFloatFun(lhs_, rhs_,-1, NamedO::str(lhs_)+”-(”+rhs_.getName()+’)’);}inlineFloatFun &operator-(constFloatFun &arg_){return ∗newScalarProductFloatFun(”-”,-1, arg_);}inlineFloatFun &operator ∗(constFloatFun &lhs_, constFloatFun &rhs_){if(&lhs_==&rhs_) return ∗newSquareFloatFun(lhs_);return ∗newProdOfTwoFuns(lhs_, rhs_);}inlineFloatFun &operator ∗(floatlhs_, constFloatFun &rhs_){return ∗newScalarProductFloatFun(lhs_, rhs_);}inlineFloatFun &operator ∗(constFloatFun &lhs_, floatrhs_){return ∗newScalarProductFloatFun(rhs_, lhs_);}=======**FloatFun** &sqrt(constFloatFun &arg_);>>>> **FloatFun** & **tan** (const **FloatFun** &arg_)

Tangent of a **FloatFun***.*

7.26 FloatFunBase.h File Reference

7.26.1 Detailed Description

Declares class **FloatFun**.

Changelog:

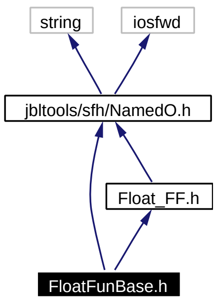
- 29.6.04 BL: Taken out of **FloatFun.h**

Definition in file **FloatFunBase.h**.

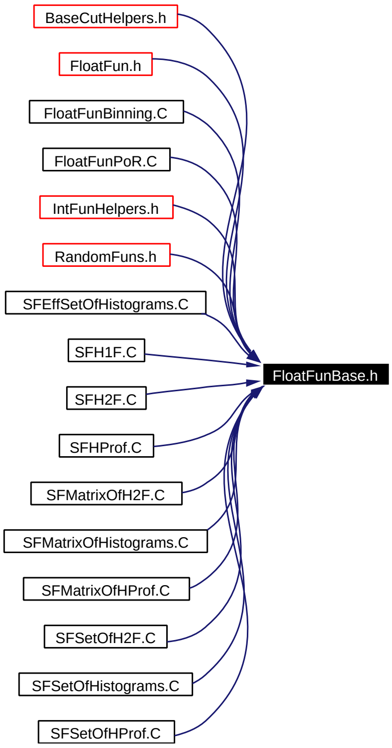
`#include "jbltools/sfh/NamedO.h"`

`#include "Float_FF.h"`

Include dependency graph for FloatFunBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **FloatFun**

  Base class for objects that return a float number for each event.

7.27 FloatFunBinning.C File Reference

7.27.1 Detailed Description

Implements **FloatFunBinning**.

Author: Jenny Boehme

Date

2006/06/30 11:05:00

Author

blist

Changelog:

- 4.6.04 BL: Derived BiningFun from Binning, so **FloatFunBinning** inherits getNBins() and getBin().
- 22.6.04 BL: Use snprintf instead of sprintf
- 15.7.04 BL: Added method getIterator()
- 23.8.05 BL: Don't access binedges anymore, Use double for xmin and xmax
- 30.6.06 BL: Bug fix: initvarname wrote 0 behind last allocated char

CVS Log messages:

- **Log**
  FloatFunBinning.C,v
- Revision 1.15 2006/06/30 11:05:00 blist
- Bug fix in **FloatFunBinning.C** **IntFunBinning.C** (segfault!)
- **SetOfHistograms** has new getSummary() method

- Revision 1.14 2005/08/23 12:05:43 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.13 2005/07/08 15:01:34 blist
- Added identity string to all .C files

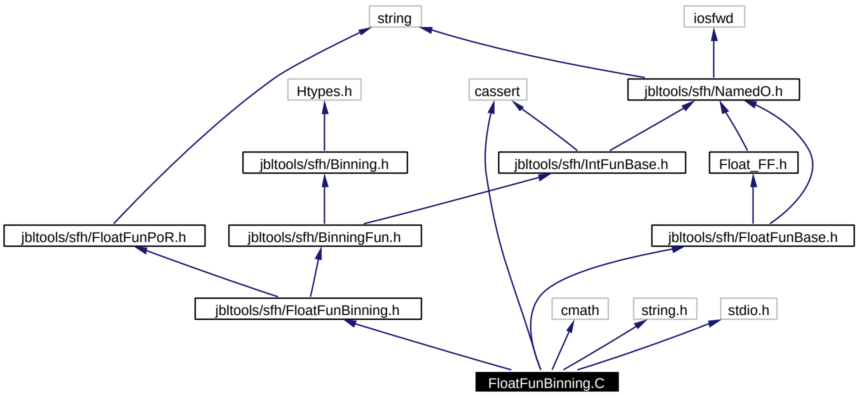
Definition in file **FloatFunBinning.C**.

```
#include "jbltools/sfh/FloatFunBinning.h"
#include "jbltools/sfh/FloatFunBase.h"
#include <cassert>
#include <cmath>
```

#include <string.h>

#include <stdio.h>

Include dependency graph for FloatFunBinning.C:



7.28 FloatFunBinning.h File Reference

7.28.1 Detailed Description

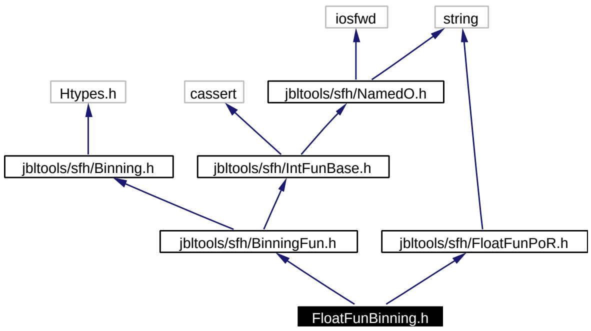
Declares **FloatFunBinning**.

Definition in file **FloatFunBinning.h**.

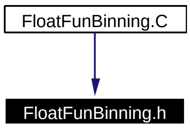
`#include "jbltools/sfh/BinningFun.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

Include dependency graph for FloatFunBinning.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **FloatFunBinning**

  A class to get a concrete **BinningFun** *subclass from a* **FloatFun**.

7.29 FloatFunHelpers.h File Reference

7.29.1 Detailed Description

Declares various helper classes for operators between **FloatFun** objects.

Changelog:

- 3.6.05 BL: First version (taken out of **FloatFun.h**)
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

Definition in file **FloatFunHelpers.h**.

`#include "jbltools/sfh/FloatFun.h"`

`#include "jbltools/sfh/JBLMath.h"`

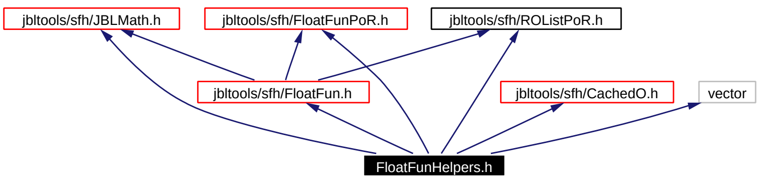
`#include "jbltools/sfh/ROListPoR.h"`

`#include "jbltools/sfh/CachedO.h"`

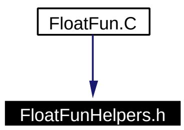
`#include "jbltools/sfh/FloatFunPoR.h"`

`#include <vector>`

Include dependency graph for FloatFunHelpers.h:



This graph shows which files directly or indirectly include this file:



7.30 FloatFunPoR.C File Reference

7.30.1 Detailed Description

Implements class **FloatFunPoR**.

Changelog:

- 29.5.05 BL: First version

CVS Log messages:

- **Log**
  FloatFunPoR.C,v
- Revision 1.4 2005/07/08 15:01:34 blist
- Added identity string to all .C files

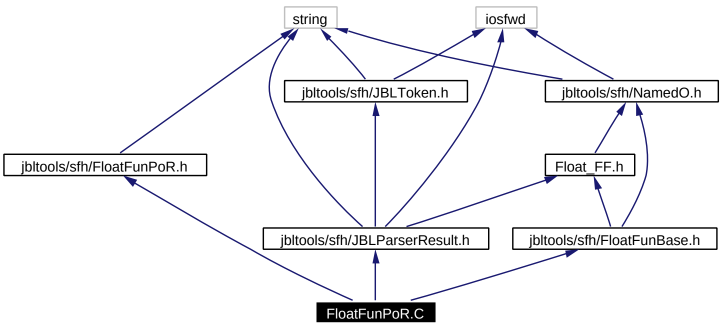
Definition in file **FloatFunPoR.C**.

`#include "jbltools/sfh/FloatFunPoR.h"`

`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/JBLParserResult.h"`

Include dependency graph for FloatFunPoR.C:



7.31 FloatFunPoR.h File Reference

7.31.1 Detailed Description

Declares classes **FloatFunPoR** and **FloatFunPoRConst**.

Author: Benno List, Jenny Böhme

Date

2005/06/29 14:40:18

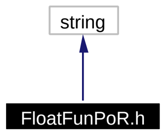
Author

blist

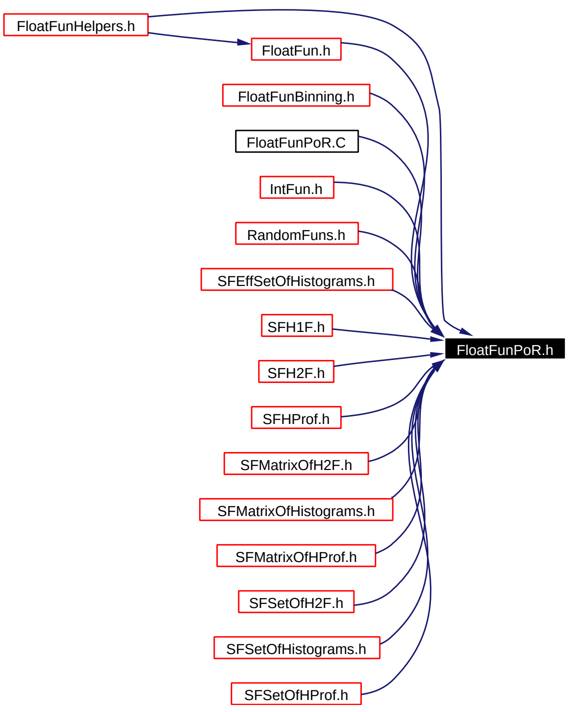
Definition in file **FloatFunPoR.h**.

`#include <string>`

Include dependency graph for FloatFunPoR.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **FloatFunPoR**

  FloatFun *Pointer or Reference.*

- class **FloatFunPoRConst**

  Constant **FloatFun** *Pointer or Reference.*

7.32 H1BankFuns.h File Reference

7.32.1 Detailed Description

Declares H1 specific **FillIterator** and **FloatFun** classes for Bank accss.

Author: Benno List, Jenny Böhme

Date
2005/07/21 11:20:12

Author
blist

Changelog:

- 8.7.04 BL First version
- 4.6.05 BL: Adapted some classes to new scheme that stores object names in **NamedO** (untested yet!)
- 5.6.05 BL: Bugs removed
- 8.6.05 BL: Bug fix
- 21.7.05 BL: Use Float_FF instead of Float_FF as result type of **FloatFun**

Definition in file **H1BankFuns.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

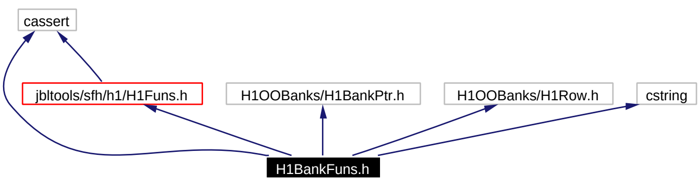
`#include "H1OOBanks/H1BankPtr.h"`

`#include "H1OOBanks/H1Row.h"`

`#include <cassert>`

`#include <cstring>`

Include dependency graph for H1BankFuns.h:



Compounds

- class **JBLH1SomeBankFloatFun**

  Returns a Float_FF value from an float-valued column of an H1 bank row.

- class **JBLH1BankFloatFunInt**

  Returns a Float_FF value from an int-valued column of an H1 bank row.

- class **JBLH1BankIntFun**

  Returns a int value from an int-valued column of an H1 bank row.

- class **JBLH1SomeBankIterator**

  A **FillIterator** *for an H1 bank.*

- class **JBLH1SomeBank**

  A class that allows generic access to an H1 bank.

Typedefs

- typedef **JBLH1SomeBankIterator**< H1BankPtr, H1Row > **JBLH1BankIterator**
- typedef **JBLH1SomeBankIterator**< H1DmisBankPtr, H1DmisRow > **JBLH1DmisBankIterator**
- typedef **JBLH1SomeBank**< H1BankPtr, H1Row > **JBLH1Bank**
- typedef **JBLH1SomeBank**< H1DmisBankPtr, H1DmisRow > **JBLH1DmisBank**

7.33 H1DQHatVariables.h File Reference

7.33.1 Detailed Description

Declares objects for H1 Data Quality HAT variable access.

Important: Generate the contents using script hatvar.pl!

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:25:00

Author

blist

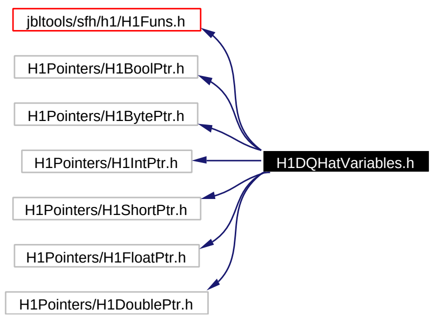
Changelog:

- 7.7.04 BL First version

Definition in file **H1DQHatVariables.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Pointers/H1BoolPtr.h"
#include "H1Pointers/H1BytePtr.h"
#include "H1Pointers/H1IntPtr.h"
#include "H1Pointers/H1ShortPtr.h"
#include "H1Pointers/H1FloatPtr.h"
#include "H1Pointers/H1DoublePtr.h"
```

Include dependency graph for H1DQHatVariables.h:



7.34 H1Funs.h File Reference

7.34.1 Detailed Description

Declares H1 specific **FillIterator** and **FloatFun** classes.

Author: Benno List, Jenny Boehme

Date
2005/08/23 14:38:08

Author
hallerma

Changelog:

- 6.7.04 BL First version
- 15.7.04 BL: Added method getIterator()
- 15.7.04 BL: Made JBLH1ClonesArrayIterator and JBLH1PartFun special cases of templates
- 22.12.04 BL: Added "isValid" to JBLH1SomeArrayIterator<>
- 4.6.05 BL: Adapted some classes to new scheme that stores object names in **NamedO** (untested yet!)
- 5.6.05 BL: Bugs removed
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

Definition in file **H1Funs.h**.

`#include "jbltools/sfh/FillIterator.h"`

`#include "jbltools/sfh/FloatFun.h"`

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/BaseCut.h"`

`#include "H1Pointers/H1ClonesArrayPtr.h"`

`#include "H1Mods/H1Part.h"`

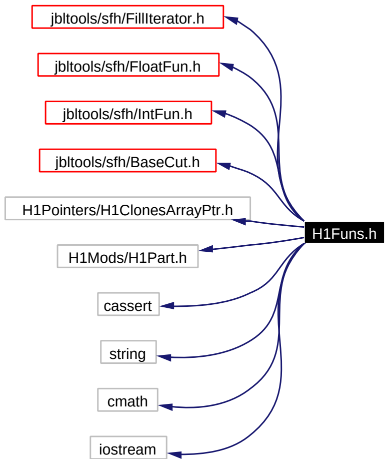
`#include <cassert>`

`#include <string>`

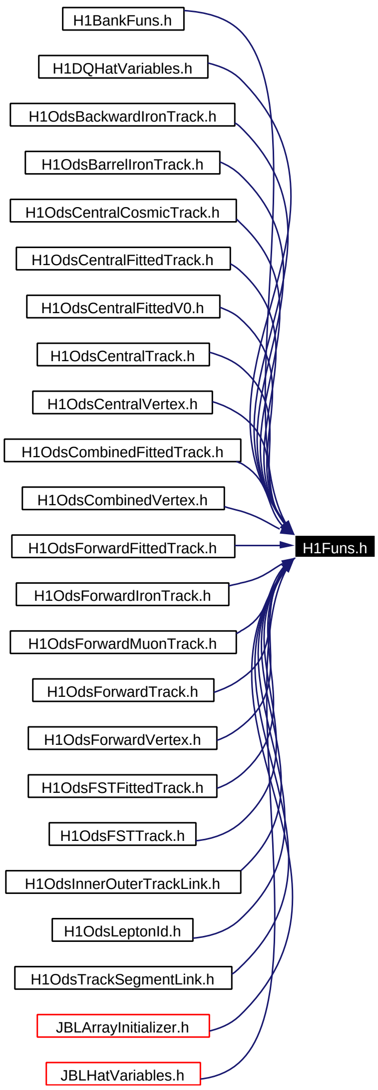
`#include <cmath>`

`#include <iostream>`

Include dependency graph for H1Funs.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **JBLH1FloatFunIter**

  A **FloatFun** *using an iterator that returns a variable from an H1Tree.*

- class **JBLH1FloatFun**

  A **FloatFun** *that returns a variable from an H1Tree.*

- class **JBLH1IntFunIter**

  A **IntFun** *using an iterator that returns a variable from an H1Tree.*

- class **JBLH1IntFun**

  A **IntFun** *that returns a variable from an H1Tree.*

- class **JBLH1SomeArrayIterator**

  A **FillIterator** *for an H1ClonesArray.*

- class **JBLH1ClonesArrayEntries**

  An **IntFun** *that returns the number of entries in an H1ClonesArray.*

Typedefs

- typedef **JBLH1SomeArrayIterator**< H1ClonesArrayPtr > **JBLH1ClonesArray-Iterator**

7.35 H1HatVariables.h File Reference

7.35.1 Detailed Description

Declares objects for H1 HAT variable access.

This file declares only static references; the static references are valid only within the file scope.

The references point to objects that are created by JBLHatVariables::init automatically.

Important: Generate the contents using script hatvar.pl!

Author: Benno List, Jenny Boehme

Date

2006/09/07 11:01:36

Author

zimmerto

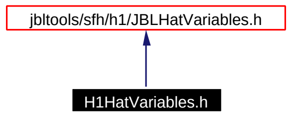
Changelog:

- 7.7.04 BL First version
- 8.6.05 BL New initialization method
- 10.5.06: BL, TZ: Add H1OO 2.8 version

Definition in file **H1HatVariables.h**.

`#include "jbltools/sfh/h1/JBLHatVariables.h"`

Include dependency graph for H1HatVariables.h:



7.36 H1ModsConvPhotons.h File Reference

7.36.1 Detailed Description

Declares objects for H1 mODS Converted photon access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:31

Author
blist

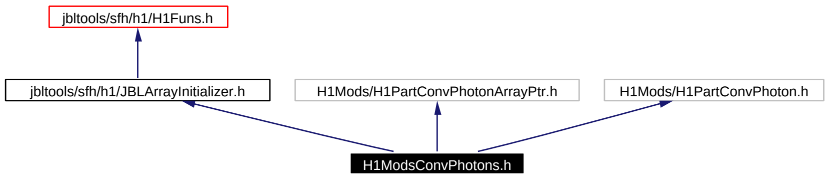
Changelog:

- 15.7.04 BL First version
- 6.6.05 BL: Make variables global instead of static to ensure only one instance in the program
- 8.6.05 BL Updated to new initialization scheme

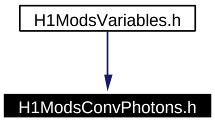
Definition in file **H1ModsConvPhotons.h**.

```
#include "jbltools/sfh/h1/JBLArrayInitializer.h"
#include "H1Mods/H1PartConvPhotonArrayPtr.h"
#include "H1Mods/H1PartConvPhoton.h"
```

Include dependency graph for H1ModsConvPhotons.h:



This graph shows which files directly or indirectly include this file:



7.37 H1ModsDstarK03Pis.h File Reference

7.37.1 Detailed Description

Declares objects for H1 mODS D∗->K0 3pi access.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:38:31

Author

blist

Changelog:

- 15.7.04 BL First version
- 6.6.05 BL: Make variables global instead of static to ensure only one instance in the program
- 8.6.05 BL Updated to new initialization scheme

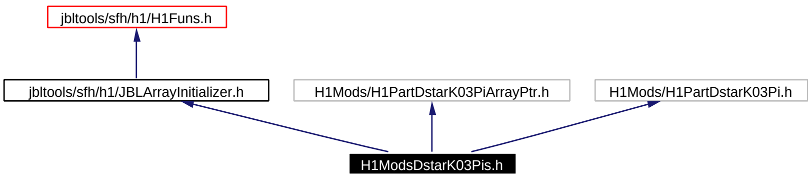
Definition in file **H1ModsDstarK03Pis.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

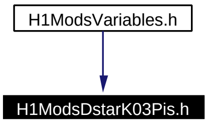
`#include "H1Mods/H1PartDstarK03PiArrayPtr.h"`

`#include "H1Mods/H1PartDstarK03Pi.h"`

Include dependency graph for H1ModsDstarK03Pis.h:



This graph shows which files directly or indirectly include this file:



7.38 H1ModsDstars.h File Reference

7.38.1 Detailed Description

Declares objects for H1 mODS D∗ -> K pi access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:02:49

Author
blist

Changelog:

- 15.7.04 BL First version
- 6.6.05 BL: Make variables global instead of static to ensure only one instance in the program
- 8.6.05 BL Updated to new initialization scheme

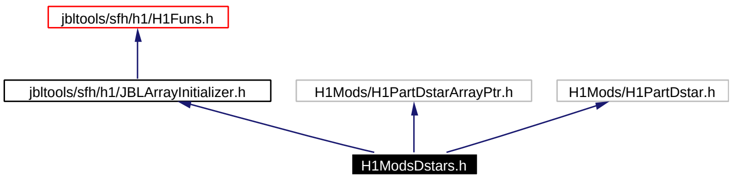
Definition in file **H1ModsDstars.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

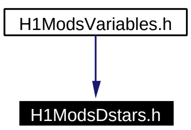
`#include "H1Mods/H1PartDstarArrayPtr.h"`

`#include "H1Mods/H1PartDstar.h"`

Include dependency graph for H1ModsDstars.h:



This graph shows which files directly or indirectly include this file:



7.39 H1ModsEms.h File Reference

7.39.1 Detailed Description

Declares objects for H1 mODS Em clusters access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:31

Author
blist

Changelog:

- 15.7.04 BL First version
- 6.6.05 BL: Make variables global instead of static to ensure only one instance in the program
- 8.6.05 BL Updated to new initialization scheme

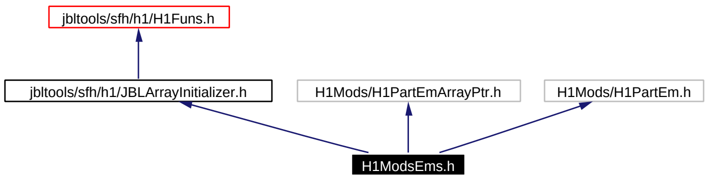
Definition in file **H1ModsEms.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

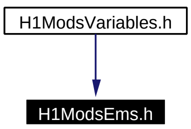
`#include "H1Mods/H1PartEmArrayPtr.h"`

`#include "H1Mods/H1PartEm.h"`

Include dependency graph for H1ModsEms.h:



This graph shows which files directly or indirectly include this file:



7.40 H1ModsFNCs.h File Reference

7.40.1 Detailed Description

Declares objects for H1 mODS FNC access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:31

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

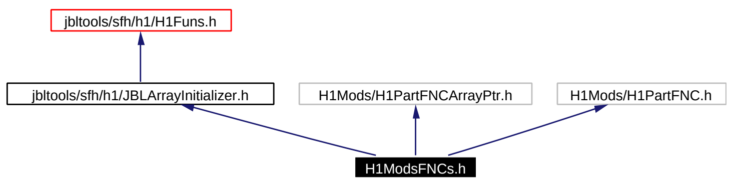
Definition in file **H1ModsFNCs.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

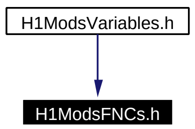
`#include "H1Mods/H1PartFNCArrayPtr.h"`

`#include "H1Mods/H1PartFNC.h"`

Include dependency graph for H1ModsFNCs.h:



This graph shows which files directly or indirectly include this file:



7.41 H1ModsFPSs.h File Reference

7.41.1 Detailed Description

Declares objects for H1 mODS FPS access.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:57:19

Author

blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

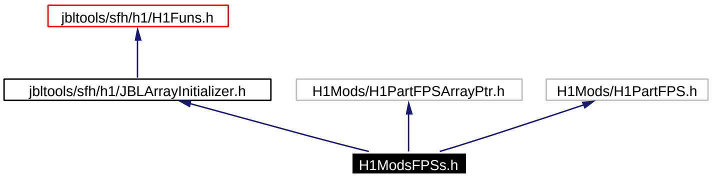
Definition in file **H1ModsFPSs.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

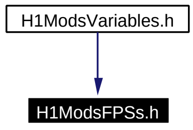
`#include "H1Mods/H1PartFPSArrayPtr.h"`

`#include "H1Mods/H1PartFPS.h"`

Include dependency graph for H1ModsFPSs.h:



This graph shows which files directly or indirectly include this file:



7.42 H1ModsJets.h File Reference

7.42.1 Detailed Description

Declares objects for H1 mODS Jet access.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:38:32

Author

blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

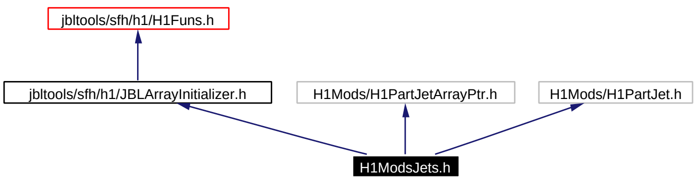
Definition in file **H1ModsJets.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

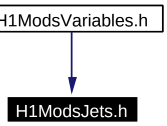
`#include "H1Mods/H1PartJetArrayPtr.h"`

`#include "H1Mods/H1PartJet.h"`

Include dependency graph for H1ModsJets.h:



This graph shows which files directly or indirectly include this file:



7.43 H1ModsJPsis.h File Reference

7.43.1 Detailed Description

Declares objects for H1 mODS J/psi access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

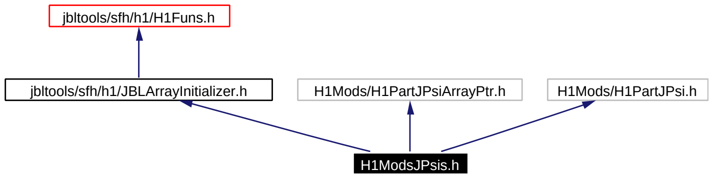
Definition in file **H1ModsJPsis.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

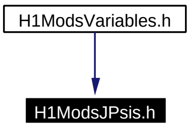
`#include "H1Mods/H1PartJPsiArrayPtr.h"`

`#include "H1Mods/H1PartJPsi.h"`

Include dependency graph for H1ModsJPsis.h:



This graph shows which files directly or indirectly include this file:



7.44 H1ModsK0s.h File Reference

7.44.1 Detailed Description

Declares objects for H1 mODS K0 access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

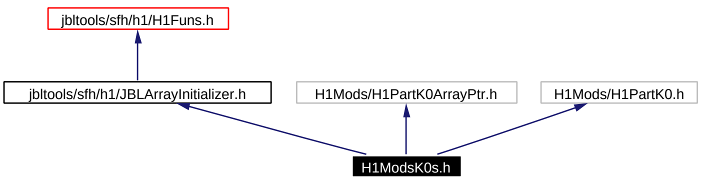
Definition in file **H1ModsK0s.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

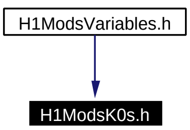
`#include "H1Mods/H1PartK0ArrayPtr.h"`

`#include "H1Mods/H1PartK0.h"`

Include dependency graph for H1ModsK0s.h:



This graph shows which files directly or indirectly include this file:



7.45 H1ModsLambdas.h File Reference

7.45.1 Detailed Description

Declares objects for H1 mODS Lambda access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

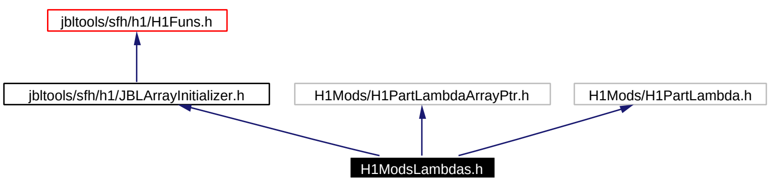
Definition in file **H1ModsLambdas.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

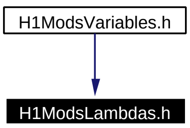
`#include "H1Mods/H1PartLambdaArrayPtr.h"`

`#include "H1Mods/H1PartLambda.h"`

Include dependency graph for H1ModsLambdas.h:



This graph shows which files directly or indirectly include this file:



7.46 H1ModsLightVMDiffs.h File Reference

7.46.1 Detailed Description

Declares objects for H1 mODS Light diffractive VM access.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:38:32

Author

blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

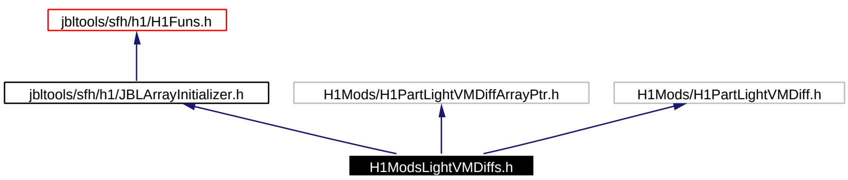
Definition in file **H1ModsLightVMDiffs.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

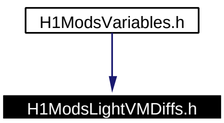
`#include "H1Mods/H1PartLightVMDiffArrayPtr.h"`

`#include "H1Mods/H1PartLightVMDiff.h"`

Include dependency graph for H1ModsLightVMDiffs.h:



This graph shows which files directly or indirectly include this file:



7.47 H1ModsMCParticles.h File Reference

7.47.1 Detailed Description

Declares objects for H1 mODS MC particle access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

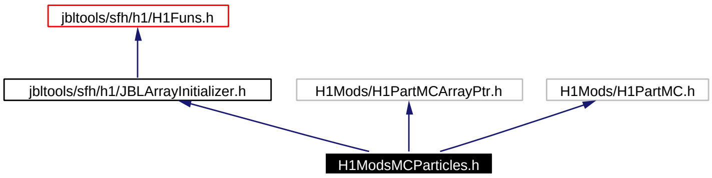
Definition in file **H1ModsMCParticles.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

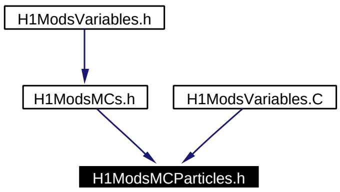
`#include "H1Mods/H1PartMCArrayPtr.h"`

`#include "H1Mods/H1PartMC.h"`

Include dependency graph for H1ModsMCParticles.h:



This graph shows which files directly or indirectly include this file:



7.48 H1ModsMCs.h File Reference

7.48.1 Detailed Description

Declares objects for H1 mODS MC particle access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

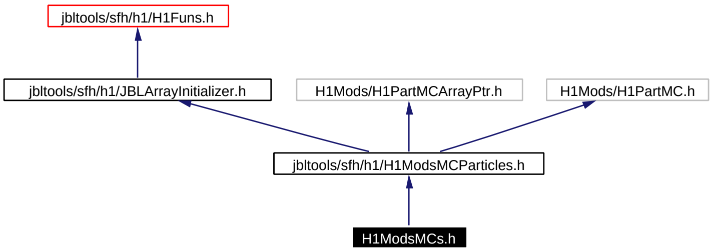
Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

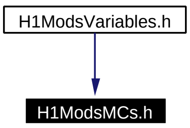
Definition in file **H1ModsMCs.h**.

`#include "jbltools/sfh/h1/H1ModsMCParticles.h"`

Include dependency graph for H1ModsMCs.h:



This graph shows which files directly or indirectly include this file:



7.49 H1ModsMuons.h File Reference

7.49.1 Detailed Description

Declares objects for H1 mODS Muon access.

Author: Benno List, Jenny Boehme

Date

2005/08/23 14:04:47

Author

hallerma

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

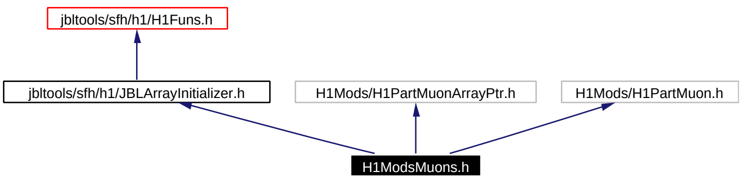
Definition in file **H1ModsMuons.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

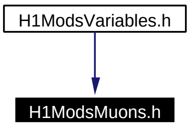
`#include "H1Mods/H1PartMuonArrayPtr.h"`

`#include "H1Mods/H1PartMuon.h"`

Include dependency graph for H1ModsMuons.h:



This graph shows which files directly or indirectly include this file:



7.50 H1ModsPartCands.h File Reference

7.50.1 Detailed Description

Declares objects for H1 mODS Particle Candidates access.

Author: Benno List, Jenny Boehme

Date
2005/06/08 07:38:32

Author
blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

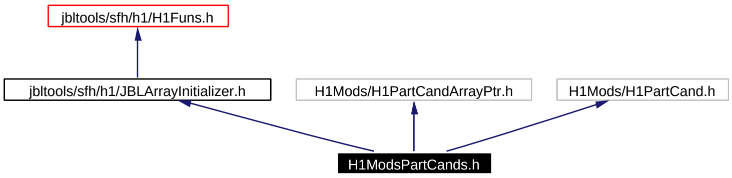
Definition in file **H1ModsPartCands.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

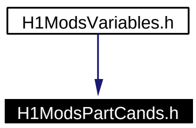
`#include "H1Mods/H1PartCandArrayPtr.h"`

`#include "H1Mods/H1PartCand.h"`

Include dependency graph for H1ModsPartCands.h:



This graph shows which files directly or indirectly include this file:



7.51 H1ModsSelTracks.h File Reference

7.51.1 Detailed Description

Declares objects for H1 mODS selected Tracks access.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:38:32

Author

blist

Changelog:

- 15.7.04 BL First version
- 8.6.05 BL Updated to new initialization scheme

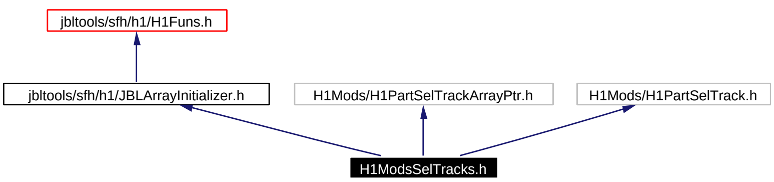
Definition in file **H1ModsSelTracks.h**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

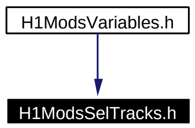
`#include "H1Mods/H1PartSelTrackArrayPtr.h"`

`#include "H1Mods/H1PartSelTrack.h"`

Include dependency graph for H1ModsSelTracks.h:



This graph shows which files directly or indirectly include this file:



7.52 H1ModsVariables.C File Reference

7.52.1 Detailed Description

Defines static members for all initializer classes needed for ODS access.

Author: Benno List, Jenny Boehme

Date

2005/08/23 14:17:06

Author

hallerma

Changelog:

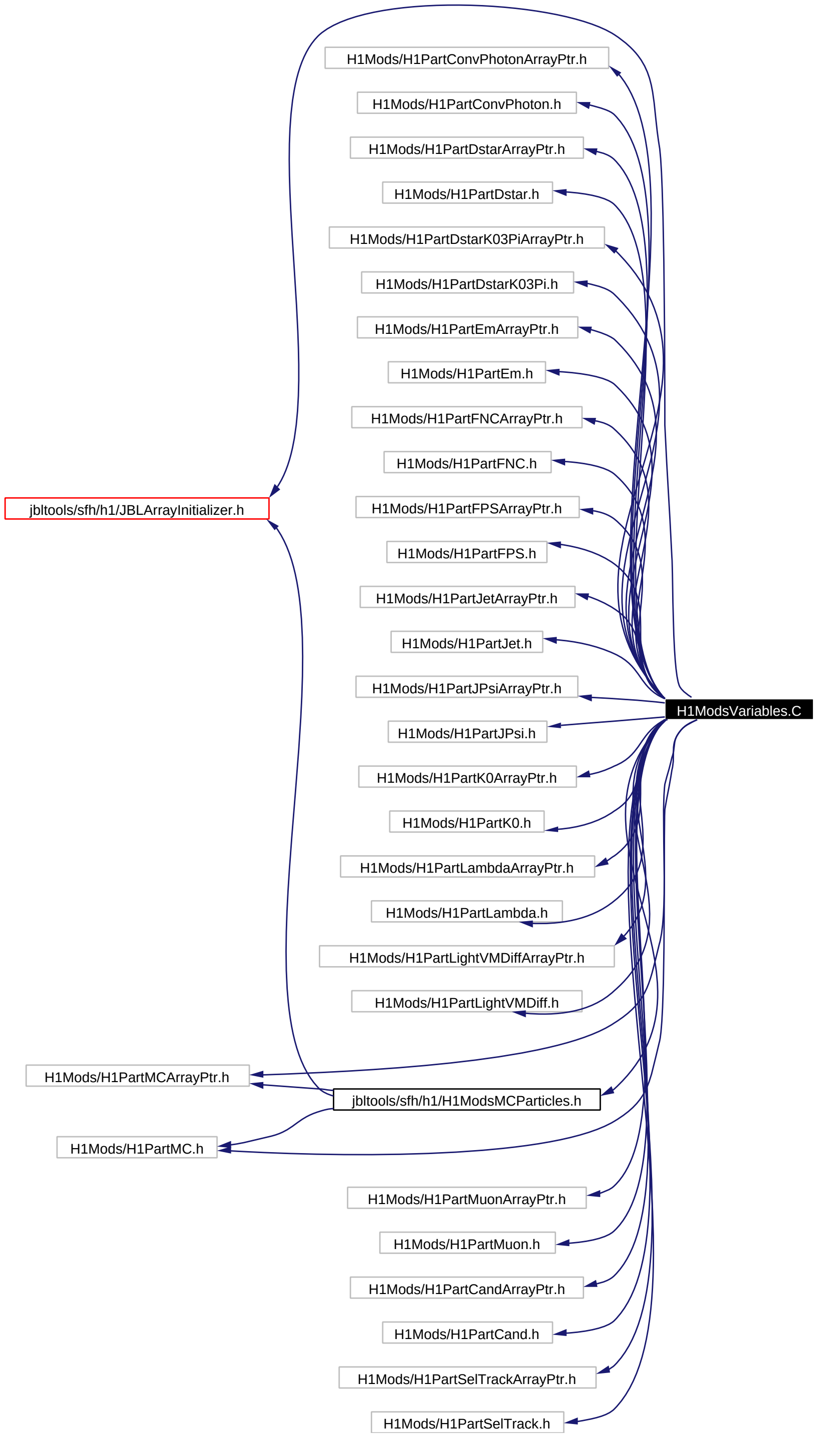
- 8.6.05 BL Updated to new initialization scheme

Definition in file **H1ModsVariables.C**.

```
#include "jbltools/sfh/h1/JBLArrayInitializer.h"
#include "H1Mods/H1PartConvPhotonArrayPtr.h"
#include "H1Mods/H1PartConvPhoton.h"
#include "H1Mods/H1PartDstarArrayPtr.h"
#include "H1Mods/H1PartDstar.h"
#include "H1Mods/H1PartDstarK03PiArrayPtr.h"
#include "H1Mods/H1PartDstarK03Pi.h"
#include "H1Mods/H1PartEmArrayPtr.h"
#include "H1Mods/H1PartEm.h"
#include "H1Mods/H1PartFNCArrayPtr.h"
#include "H1Mods/H1PartFNC.h"
#include "H1Mods/H1PartFPSArrayPtr.h"
#include "H1Mods/H1PartFPS.h"
#include "H1Mods/H1PartJetArrayPtr.h"
#include "H1Mods/H1PartJet.h"
#include "H1Mods/H1PartJPsiArrayPtr.h"
#include "H1Mods/H1PartJPsi.h"
#include "H1Mods/H1PartK0ArrayPtr.h"
#include "H1Mods/H1PartK0.h"
#include "H1Mods/H1PartLambdaArrayPtr.h"
#include "H1Mods/H1PartLambda.h"
#include "H1Mods/H1PartLightVMDiffArrayPtr.h"
#include "H1Mods/H1PartLightVMDiff.h"
```

```
#include "H1Mods/H1PartMCArrayPtr.h"
#include "H1Mods/H1PartMC.h"
#include "jbltools/sfh/h1/H1ModsMCParticles.h"
#include "H1Mods/H1PartMuonArrayPtr.h"
#include "H1Mods/H1PartMuon.h"
#include "H1Mods/H1PartCandArrayPtr.h"
#include "H1Mods/H1PartCand.h"
#include "H1Mods/H1PartSelTrackArrayPtr.h"
#include "H1Mods/H1PartSelTrack.h"
```

Include dependency graph for H1ModsVariables.C:


H1Mods/H1PartConvPhotonArrayPtr.h
H1Mods/H1PartConvPhoton.h
H1Mods/H1PartDstarArrayPtr.h
H1Mods/H1PartDstar.h
H1Mods/H1PartDstarK03PiArrayPtr.h
H1Mods/H1PartDstarK03Pi.h
H1Mods/H1PartEmArrayPtr.h
H1Mods/H1PartEm.h
H1Mods/H1PartFNCArrayPtr.h
H1Mods/H1PartFNC.h
jbltools/sfh/h1/JBLArrayInitializer.h
H1Mods/H1PartFPSArrayPtr.h
H1Mods/H1PartFPS.h
H1Mods/H1PartJetArrayPtr.h
H1Mods/H1PartJet.h
H1Mods/H1PartJPsiArrayPtr.h
H1ModsVariables.C
H1Mods/H1PartJPsi.h
H1Mods/H1PartK0ArrayPtr.h
H1Mods/H1PartK0.h
H1Mods/H1PartLambdaArrayPtr.h
H1Mods/H1PartLambda.h
H1Mods/H1PartLightVMDiffArrayPtr.h
H1Mods/H1PartLightVMDiff.h
H1Mods/H1PartMCArrayPtr.h
jbltools/sfh/h1/H1ModsMCParticles.h
H1Mods/H1PartMC.h
H1Mods/H1PartMuonArrayPtr.h
H1Mods/H1PartMuon.h
H1Mods/H1PartCandArrayPtr.h
H1Mods/H1PartCand.h
H1Mods/H1PartSelTrackArrayPtr.h
H1Mods/H1PartSelTrack.h

7.53 H1ModsVariables.h File Reference

7.53.1 Detailed Description

Declares objects for H1 mODS variable access.

Author: Benno List, Jenny Boehme

Date

2005/06/06 15:02:03

Author

blist

Changelog:

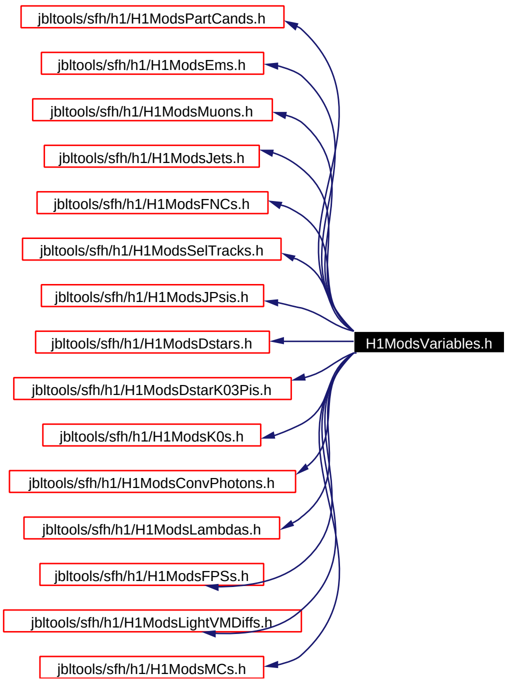
- 7.7.04 BL First version

- 16.7.04 BL Split into different .h files

Definition in file **H1ModsVariables.h**.

```
#include "jbltools/sfh/h1/H1ModsPartCands.h"
#include "jbltools/sfh/h1/H1ModsEms.h"
#include "jbltools/sfh/h1/H1ModsMuons.h"
#include "jbltools/sfh/h1/H1ModsJets.h"
#include "jbltools/sfh/h1/H1ModsFNCs.h"
#include "jbltools/sfh/h1/H1ModsSelTracks.h"
#include "jbltools/sfh/h1/H1ModsJPsis.h"
#include "jbltools/sfh/h1/H1ModsDstars.h"
#include "jbltools/sfh/h1/H1ModsDstarK03Pis.h"
#include "jbltools/sfh/h1/H1ModsK0s.h"
#include "jbltools/sfh/h1/H1ModsConvPhotons.h"
#include "jbltools/sfh/h1/H1ModsLambdas.h"
#include "jbltools/sfh/h1/H1ModsFPSs.h"
#include "jbltools/sfh/h1/H1ModsLightVMDiffs.h"
#include "jbltools/sfh/h1/H1ModsMCs.h"
```

Include dependency graph for H1ModsVariables.h:


jbltools/sfh/h1/H1ModsPartCands.h
jbltools/sfh/h1/H1ModsEms.h
jbltools/sfh/h1/H1ModsMuons.h
jbltools/sfh/h1/H1ModsJets.h
jbltools/sfh/h1/H1ModsFNCs.h
jbltools/sfh/h1/H1ModsSelTracks.h
jbltools/sfh/h1/H1ModsJPsis.h
jbltools/sfh/h1/H1ModsDstars.h
H1ModsVariables.h
jbltools/sfh/h1/H1ModsDstarK03Pis.h
jbltools/sfh/h1/H1ModsK0s.h
jbltools/sfh/h1/H1ModsConvPhotons.h
jbltools/sfh/h1/H1ModsLambdas.h
jbltools/sfh/h1/H1ModsFPSs.h
jbltools/sfh/h1/H1ModsLightVMDiffs.h
jbltools/sfh/h1/H1ModsMCs.h

7.54 H1OdsBackwardIronTrack.h File Reference

7.54.1 Detailed Description

Declares objects for H1 ODS Backward Iron Track access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:20

Author

blist

Changelog:

- 15.7.04 BL First version

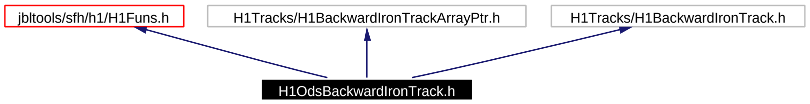
Definition in file **H1OdsBackwardIronTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1BackwardIronTrackArrayPtr.h"`

`#include "H1Tracks/H1BackwardIronTrack.h"`

Include dependency graph for H1OdsBackwardIronTrack.h:



7.55 H1OdsBarrelIronTrack.h File Reference

7.55.1 Detailed Description

Declares objects for H1 ODS Barrel Iron Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:21

Author
blist

Changelog:

- 15.7.04 BL First version

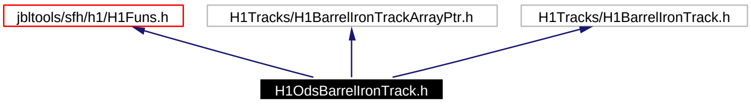
Definition in file **H1OdsBarrelIronTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1BarrelIronTrackArrayPtr.h"`

`#include "H1Tracks/H1BarrelIronTrack.h"`

Include dependency graph for H1OdsBarrelIronTrack.h:



7.56 H1OdsCentralCosmicTrack.h File Reference

7.56.1 Detailed Description

Declares objects for H1 ODS Central Cosmic Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:22

Author
blist

Changelog:

- 15.7.04 BL First version

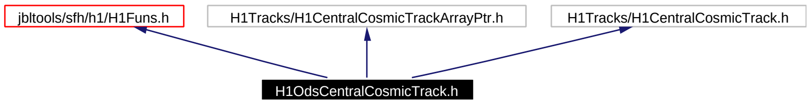
Definition in file **H1OdsCentralCosmicTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CentralCosmicTrackArrayPtr.h"`

`#include "H1Tracks/H1CentralCosmicTrack.h"`

Include dependency graph for H1OdsCentralCosmicTrack.h:



7.57 H1OdsCentralFittedTrack.h File Reference

7.57.1 Detailed Description

Declares objects for H1 ODS Central Fitted Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:22

Author
blist

Changelog:

- 15.7.04 BL First version

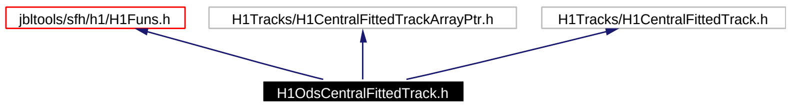
Definition in file **H1OdsCentralFittedTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CentralFittedTrackArrayPtr.h"`

`#include "H1Tracks/H1CentralFittedTrack.h"`

Include dependency graph for H1OdsCentralFittedTrack.h:



7.58 H1OdsCentralFittedV0.h File Reference

7.58.1 Detailed Description

Declares objects for H1 ODS Central Fitted V0 access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:23

Author
blist

Changelog:

- 15.7.04 BL First version

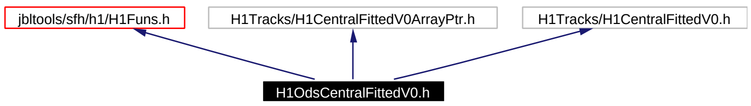
Definition in file **H1OdsCentralFittedV0.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CentralFittedV0ArrayPtr.h"`

`#include "H1Tracks/H1CentralFittedV0.h"`

Include dependency graph for H1OdsCentralFittedV0.h:



7.59 H1OdsCentralTrack.h File Reference

7.59.1 Detailed Description

Declares objects for H1 ODS Central Track access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:23

Author

blist

Changelog:

- 15.7.04 BL First version

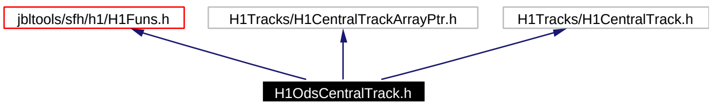
Definition in file **H1OdsCentralTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CentralTrackArrayPtr.h"`

`#include "H1Tracks/H1CentralTrack.h"`

Include dependency graph for H1OdsCentralTrack.h:



7.60 H1OdsCentralVertex.h File Reference

7.60.1 Detailed Description

Declares objects for H1 ODS Central Vertex access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:24

Author
blist

Changelog:

- 15.7.04 BL First version

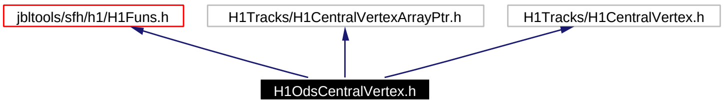
Definition in file **H1OdsCentralVertex.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CentralVertexArrayPtr.h"`

`#include "H1Tracks/H1CentralVertex.h"`

Include dependency graph for H1OdsCentralVertex.h:



7.61 H1OdsCombinedFittedTrack.h File Reference

7.61.1 Detailed Description

Declares objects for H1 ODS Combined Fitted Track access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:25

Author

blist

Changelog:

- 15.7.04 BL First version

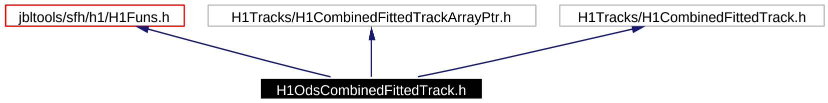
Definition in file **H1OdsCombinedFittedTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CombinedFittedTrackArrayPtr.h"`

`#include "H1Tracks/H1CombinedFittedTrack.h"`

Include dependency graph for H1OdsCombinedFittedTrack.h:



7.62 H1OdsCombinedVertex.h File Reference

7.62.1 Detailed Description

Declares objects for H1 ODS Combined Vertex access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:25

Author

blist

Changelog:

- 15.7.04 BL First version

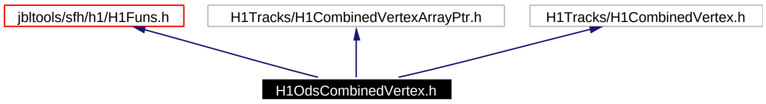
Definition in file **H1OdsCombinedVertex.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1CombinedVertexArrayPtr.h"`

`#include "H1Tracks/H1CombinedVertex.h"`

Include dependency graph for H1OdsCombinedVertex.h:



7.63 H1OdsForwardFittedTrack.h File Reference

7.63.1 Detailed Description

Declares objects for H1 ODS Forward Fitted Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:27

Author
blist

Changelog:

- 15.7.04 BL First version

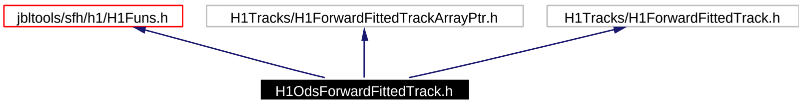
Definition in file **H1OdsForwardFittedTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1ForwardFittedTrackArrayPtr.h"`

`#include "H1Tracks/H1ForwardFittedTrack.h"`

Include dependency graph for H1OdsForwardFittedTrack.h:



7.64 H1OdsForwardIronTrack.h File Reference

7.64.1 Detailed Description

Declares objects for H1 ODS Forward Iron Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:28

Author
blist

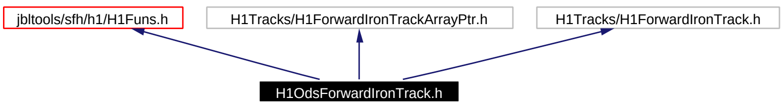
Changelog:

- 15.7.04 BL First version

Definition in file **H1OdsForwardIronTrack.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Tracks/H1ForwardIronTrackArrayPtr.h"
#include "H1Tracks/H1ForwardIronTrack.h"
```

Include dependency graph for H1OdsForwardIronTrack.h:



7.65 H1OdsForwardMuonTrack.h File Reference

7.65.1 Detailed Description

Declares objects for H1 ODS Forward Muon Track access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:29

Author

blist

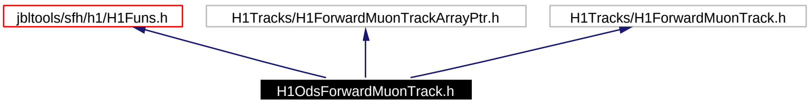
Changelog:

- 15.7.04 BL First version

Definition in file **H1OdsForwardMuonTrack.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Tracks/H1ForwardMuonTrackArrayPtr.h"
#include "H1Tracks/H1ForwardMuonTrack.h"
```

Include dependency graph for H1OdsForwardMuonTrack.h:



7.66 H1OdsForwardTrack.h File Reference

7.66.1 Detailed Description

Declares objects for H1 ODS Forward Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:30

Author
blist

Changelog:

- 15.7.04 BL First version

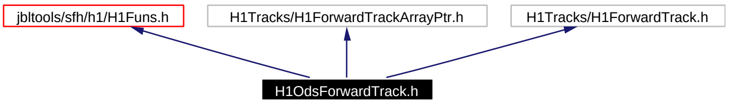
Definition in file **H1OdsForwardTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1ForwardTrackArrayPtr.h"`

`#include "H1Tracks/H1ForwardTrack.h"`

Include dependency graph for H1OdsForwardTrack.h:



7.67 H1OdsForwardVertex.h File Reference

7.67.1 Detailed Description

Declares objects for H1 ODS Forward Vertex access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:30

Author

blist

Changelog:

- 15.7.04 BL First version

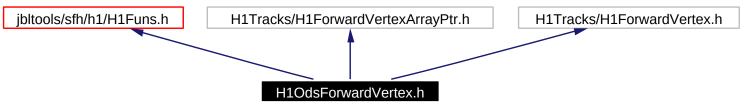
Definition in file **H1OdsForwardVertex.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1ForwardVertexArrayPtr.h"`

`#include "H1Tracks/H1ForwardVertex.h"`

Include dependency graph for H1OdsForwardVertex.h:



7.68 H1OdsFSTFittedTrack.h File Reference

7.68.1 Detailed Description

Declares objects for H1 ODS FST Fitted Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:26

Author
blist

Changelog:

- 15.7.04 BL First version

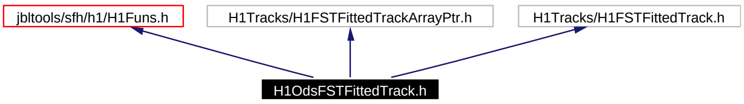
Definition in file **H1OdsFSTFittedTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1FSTFittedTrackArrayPtr.h"`

`#include "H1Tracks/H1FSTFittedTrack.h"`

Include dependency graph for H1OdsFSTFittedTrack.h:



7.69 H1OdsFSTTrack.h File Reference

7.69.1 Detailed Description

Declares objects for H1 ODS FST Track access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:27

Author
blist

Changelog:

- 15.7.04 BL First version

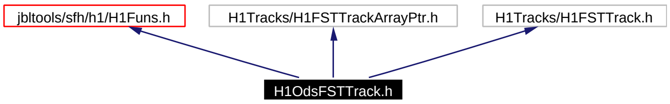
Definition in file **H1OdsFSTTrack.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1FSTTrackArrayPtr.h"`

`#include "H1Tracks/H1FSTTrack.h"`

Include dependency graph for H1OdsFSTTrack.h:



7.70 H1OdsInnerOuterTrackLink.h File Reference

7.70.1 Detailed Description

Declares objects for H1 ODS Inner-Outer Track Link access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:30

Author

blist

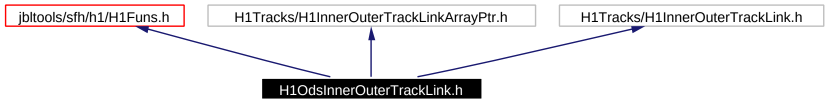
Changelog:

- 15.7.04 BL First version

Definition in file **H1OdsInnerOuterTrackLink.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Tracks/H1InnerOuterTrackLinkArrayPtr.h"
#include "H1Tracks/H1InnerOuterTrackLink.h"
```

Include dependency graph for H1OdsInnerOuterTrackLink.h:



7.71 H1OdsLeptonId.h File Reference

7.71.1 Detailed Description

Declares objects for H1 ODS Lepton ID access.

Author: Benno List, Jenny Boehme

Date
2004/07/16 15:09:31

Author
blist

Changelog:

- 15.7.04 BL First version

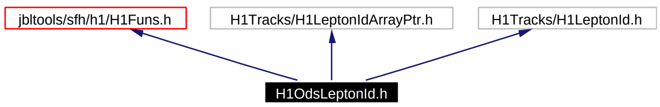
Definition in file **H1OdsLeptonId.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

`#include "H1Tracks/H1LeptonIdArrayPtr.h"`

`#include "H1Tracks/H1LeptonId.h"`

Include dependency graph for H1OdsLeptonId.h:



7.72 H1OdsTrackSegmentLink.h File Reference

7.72.1 Detailed Description

Declares objects for H1 ODS Track-Segment Link Track access.

Author: Benno List, Jenny Boehme

Date

2004/07/16 15:09:32

Author

blist

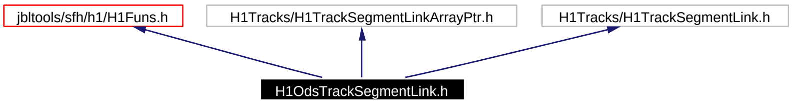
Changelog:

- 15.7.04 BL First version

Definition in file **H1OdsTrackSegmentLink.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Tracks/H1TrackSegmentLinkArrayPtr.h"
#include "H1Tracks/H1TrackSegmentLink.h"
```

Include dependency graph for H1OdsTrackSegmentLink.h:



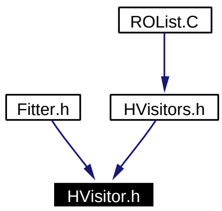
7.73 HVisitor.h File Reference

7.73.1 Detailed Description

Declares **HVisitor** and **ConstHVisitor**.

Definition in file **HVisitor.h**.

This graph shows which files directly or indirectly include this file:



Compounds

- class **HVisitor**

  Abstract base class for a visitor to all histograms.

- class **ConstHVisitor**

  Abstract base class for a constant visitor to all histograms.

7.74 HVisitors.h File Reference

7.74.1 Detailed Description

Declares some useful **HVisitor** subclasses.

Changelog:

- 30.6.04 BL: Moved several classes to rhostuff/RhoVisitors.h

Definition in file **HVisitors.h**.

`#include "jbltools/sfh/HVisitor.h"`

`#include <TObject.h>`

`#include <TH1.h>`

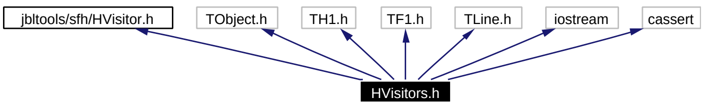
`#include <TF1.h>`

`#include <TLine.h>`

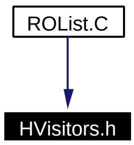
`#include <iostream>`

`#include <cassert>`

Include dependency graph for HVisitors.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **Writer**

  An **HVisitor** *that writes all TObject objects into an (already open) root file.*

- class **Normalizer**

  An **HVisitor** *that normalizes all TH1 objects to a constant.*

- class **BinWidthNormalizer**

  An **HVisitor** *that normalizes all TH1 objects to a constant bin width in x.*

- class **AttMarkerSetter**

An **HVisitor** *that sets the marker attributes color, size, and style.*

- class **AttLineSetter**

  An **HVisitor** *that sets the line attributes lcolor, lstyle, and lwidth.*

- class **AttFillSetter**

  An **HVisitor** *that sets the Fill attributes fcolor, and fstyle.*

- class **ErrorEraser**

  An **HVisitor** *that sets all bin errors to 0 for TH1 histograms.*

- class **MinMaxSetter**

  An **HVisitor** *that sets the minumum and maximum of all TH1 histograms.*

- class **AxisTitleAttribSetter**

  An **HVisitor** *that sets the axis attribues name, color, style, and number of divisions.*

- class **AxisLabelAttribSetter**

  An **HVisitor** *that sets the axis attribues name, color, style, and number of divisions.*

- class **AxisSetter**

  An **HVisitor** *that sets the x axis of a histogram to a given binning.*

7.75 IntFun.C File Reference

7.75.1 Detailed Description

Implements stuff connected with **IntFun** subclasses.

Changelog:

- 21.7.04 BL: First version

CVS Log messages:

- **Log**
  IntFun.C,v
- Revision 1.16 2005/11/22 09:00:17 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
- Revision 1.15 2005/11/21 21:43:40 blist
- Reasonable random numbers, and test
- Revision 1.14 2005/11/21 19:58:58 blist
- New classes **GaussRandomFun**, **Poisson1Fun**, **Poisson2Fun**
- Revision 1.13 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **IntFun.C**.

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/FloatFun.h"`

`#include "jbltools/sfh/IntFunHelpers.h"`

`#include "jbltools/sfh/JBLMath.h"`

`#include <cstdlib>`

`#include <iostream>`

`#include <climits>`

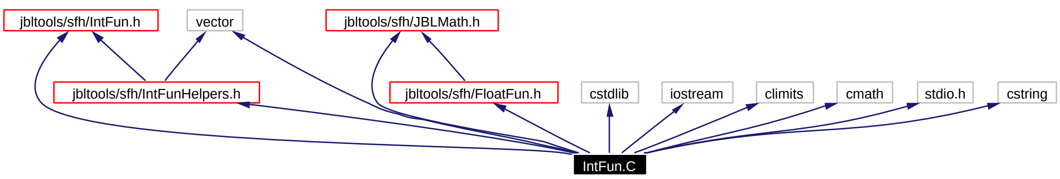
`#include <cmath>`

`#include <stdio.h>`

`#include <cstring>`

`#include <vector>`

Include dependency graph for IntFun.C:



Functions

- **IntFun** & **cached** (const **ROListPoR** &rol, const **IntFunPoR** &fun, const **FillIterator-PoR** &iter)
- **FloatFun** & **Float** (const **IntFun** &arg)
- **IntFun** & **operator+** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator+** (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator+** (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator-** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator-** (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator-** (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator-** (const **IntFun** &arg_)
- **IntFun** & **operator+** (**IntFun** &arg_)
- const **IntFun** & **operator+** (const **IntFun** &arg_)
- **IntFun** & **operator** * (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator/** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator** * (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator** * (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator/** (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator &** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator**| (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator**^ (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator &** (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator &** (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator**| (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator**| (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator**^ (const **IntFun** &lhs_, int rhs_)
- **IntFun** & **operator**^ (int lhs_, const **IntFun** &rhs_)
- **IntFun** & **operator**~ (const **IntFun** &arg_)
- **IntFun** & **square** (const **IntFun** &arg_)
- **IntFun** & **abs** (const **IntFun** &arg_)
- **IntFunComp** & **operator**< (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**> (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**<**=** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**>**=** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator==** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator!=** (const **IntFun** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**< (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**<**=** (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**> (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**>**=** (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator==** (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator!=** (int lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator**< (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator**<**=** (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator**> (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator**>**=** (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator==** (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator!=** (const **IntFun** &lhs_, int rhs_)
- **IntFunComp** & **operator**< (const **IntFunComp** &lhs_, const **IntFun** &rhs_)

- **IntFunComp** & **operator<=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator>** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator>=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator==** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator!=** (const **IntFunComp** &lhs_, const **IntFun** &rhs_)
- **IntFunComp** & **operator<** (const **IntFunComp** &lhs_, int rhs_)
- **IntFunComp** & **operator<=** (const **IntFunComp** &lhs_, int rhs_)
- **IntFunComp** & **operator>** (const **IntFunComp** &lhs_, int rhs_)
- **IntFunComp** & **operator>=** (const **IntFunComp** &lhs_, int rhs_)
- **IntFunComp** & **operator==** (const **IntFunComp** &lhs_, int rhs_)
- **IntFunComp** & **operator!=** (const **IntFunComp** &lhs_, int rhs_)
- **IntFun** & **max** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)
- **IntFun** & **min** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)
- **IntFun** & **sum** (const **IntFunPoRConst** &fun, const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)
- **IntFun** & **count** (const **BaseCutPoRConst** &cut, const **FillIteratorPoR** &intiter, const **FillIteratorPoRConst** &extiter)

7.76 IntFun.h File Reference

7.76.1 Detailed Description

Defines arithmetic operators between and functions of **IntFun** objects.

Changelog:

- 5.7.04 BL: First version
- 19.7.04 BL: Added bitwise operations, changed internal helper classes to store **IntFun** references instead of pointers
- 4.5.05 BL: Bug in operator> fixed, added sum and difference with scalar
- 1.6.05 BL: Added division operators and classes
- 2.6.05 BL: Adapted most classes to new naming scheme
- 3.6.05 BL: Moved many helper classes into **IntFunHelpers.h**
- 4.6.05 BL: Moved more helper classes into **IntFunHelpers.h**, removed dependency from , decoupled **IntFun.h** from many other header files
- 8.6.05 BL: "inline" removed for square
- 22.11.05 BL: Moved **Poisson1Fun**, Posson2Fun to Randoms.h

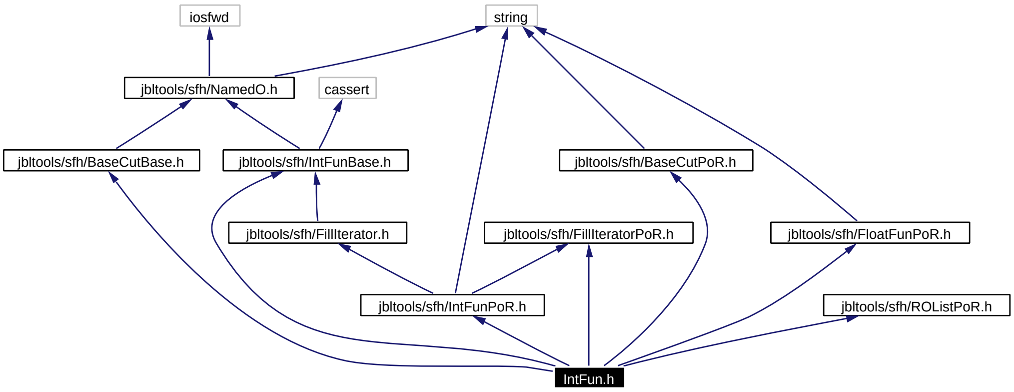
CVS Log messages:

Log
- **IntFun.h**,v
- Revision 1.22 2005/11/22 09:00:18 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

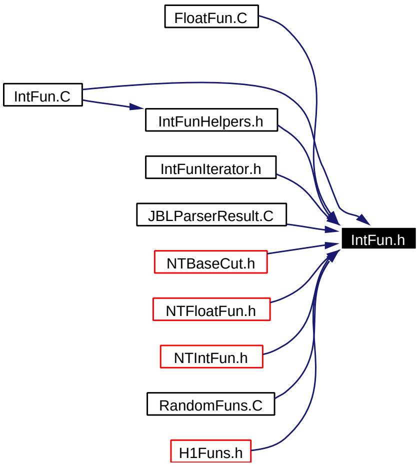
Definition in file **IntFun.h**.

```
#include "jbltools/sfh/IntFunBase.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/BaseCutPoR.h"
#include "jbltools/sfh/FillIteratorPoR.h"
#include "jbltools/sfh/FloatFunPoR.h"
#include "jbltools/sfh/IntFunPoR.h"
#include "jbltools/sfh/ROListPoR.h"
```

Include dependency graph for IntFun.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ConstIntFun**

  A **IntFun** *that returns a constant value.*

- class **IntFunComp**

  Abstract base class for comparisons between **IntFun** *objects.*

Functions

- const **IntFun** & **operator+** (const **IntFun** &arg_)

7.77 IntFunBase.h File Reference

7.77.1 Detailed Description

Declares class **IntFun**.

Changelog:

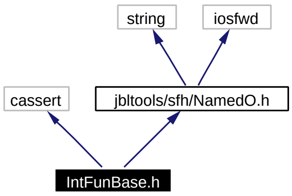
- 5.7.04 BL: First version

Definition in file **IntFunBase.h**.

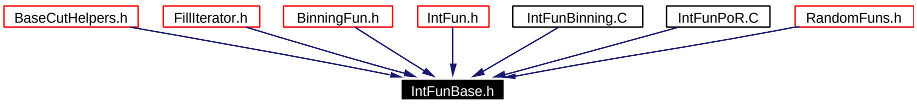
`#include <cassert>`

`#include "jbltools/sfh/NamedO.h"`

Include dependency graph for IntFunBase.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **IntFun**

  Base class for objects that return a int number for each event.

7.78 IntFunBinning.C File Reference

7.78.1 Detailed Description

Implements **IntFunBinning**.

Author: Benno List, Jenny Böhme

Date

2006/06/30 11:05:00

Author

blist

Changelog:

- 4.8.04 BL: First version, based on **FloatFunBinning**
- 23.8.05 BL: Don't access binedges anymore
- 30.6.06 BL: Bug fix: initvarname wrote 0 behind last allocated char

CVS Log messages:

- **Log**
  IntFunBinning.C,v
- Revision 1.4 2006/06/30 11:05:00 blist
- Bug fix in **FloatFunBinning.C** **IntFunBinning.C** (segfault!)
- **SetOfHistograms** has new getSummary() method

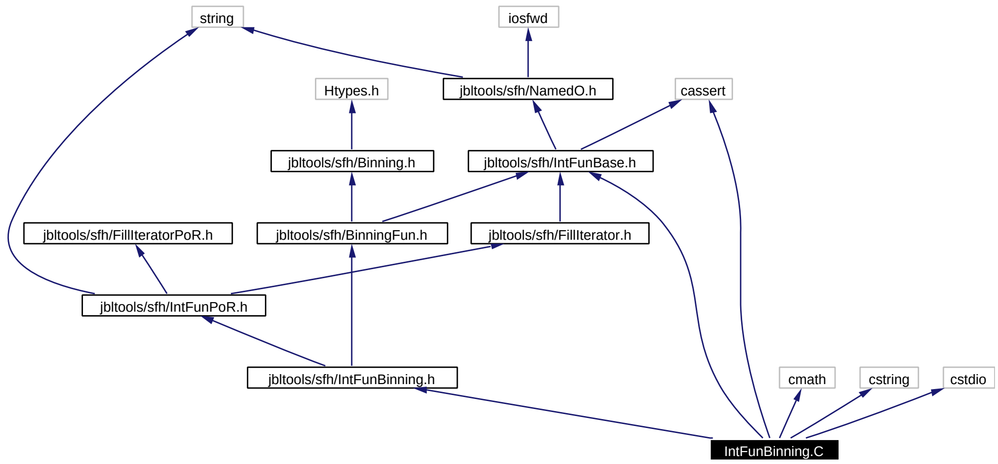
- Revision 1.3 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.2 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **IntFunBinning.C**.

```
#include "jbltools/sfh/IntFunBinning.h"
#include "jbltools/sfh/IntFunBase.h"
#include <cassert>
#include <cmath>
#include <cstring>
#include <cstdio>
```

Include dependency graph for IntFunBinning.C:



7.79 IntFunBinning.h File Reference

7.79.1 Detailed Description

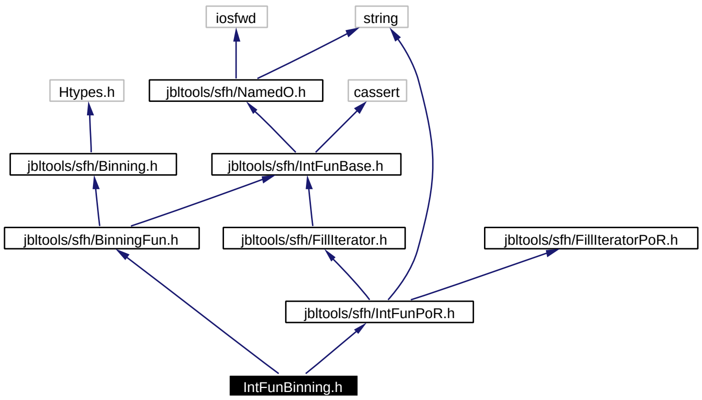
Declares **IntFunBinning**.

Definition in file **IntFunBinning.h**.

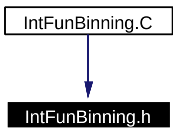
`#include "jbltools/sfh/BinningFun.h"`

`#include "jbltools/sfh/IntFunPoR.h"`

Include dependency graph for IntFunBinning.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **IntFunBinning**

  A class to get a concrete **BinningFun** *subclass from a* **IntFun***.*

7.80 IntFunHelpers.h File Reference

7.80.1 Detailed Description

Declares various helper classes for operators between **IntFun** objects.

Changelog:

- 3.6.05 BL: First version (taken out of **IntFun.h**)
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**
- 8.11.05 BL: Added getIterator for subclasses of **IntFunComp**

Definition in file **IntFunHelpers.h**.

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/CachedO.h"`

`#include "jbltools/sfh/ROListPoR.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

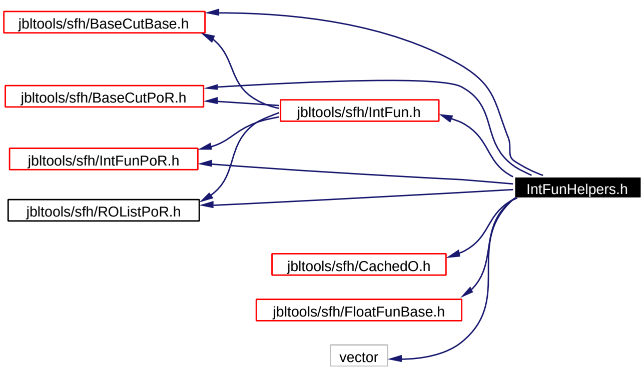
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/IntFunPoR.h"`

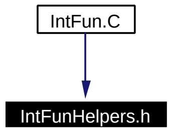
`#include "jbltools/sfh/FloatFunBase.h"`

`#include <vector>`

Include dependency graph for IntFunHelpers.h:



This graph shows which files directly or indirectly include this file:



7.81 IntFunIterator.h File Reference

7.81.1 Detailed Description

Declares class **IntFunIterator**.

Changelog: 5.7.04 BL First version

Definition in file **IntFunIterator.h**.

`#include "jbltools/sfh/IntFun.h"`

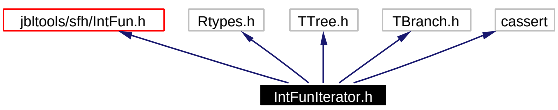
`#include <Rtypes.h>`

`#include <TTree.h>`

`#include <TBranch.h>`

`#include <cassert>`

Include dependency graph for IntFunIterator.h:



Compounds

- class **IntFunIterator**

  A **FillIterator** *from an* **IntFun** *object.*

7.82 IntFunPoR.C File Reference

7.82.1 Detailed Description

Implements class **IntFunPoR**.

Changelog:

- 29.5.05 BL: First version

CVS Log messages:

- **Log**
  IntFunPoR.C,v
- Revision 1.4 2005/07/08 15:01:34 blist
- Added identity string to all .C files

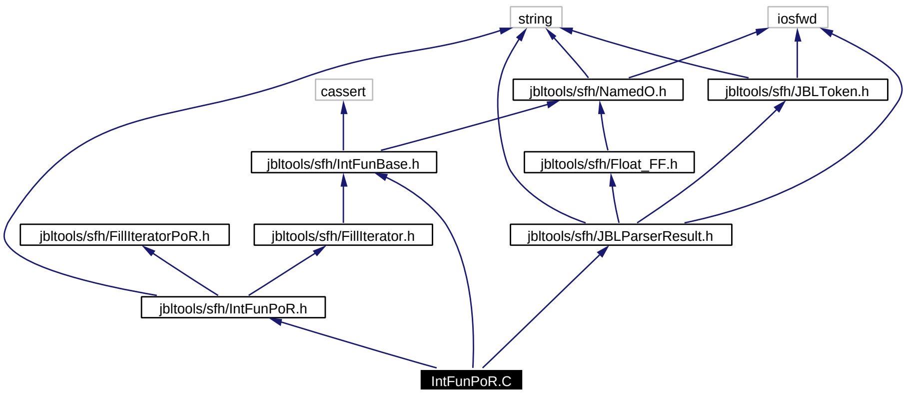
Definition in file **IntFunPoR.C**.

`#include "jbltools/sfh/IntFunPoR.h"`

`#include "jbltools/sfh/IntFunBase.h"`

`#include "jbltools/sfh/JBLParserResult.h"`

Include dependency graph for IntFunPoR.C:



7.83 IntFunPoR.h File Reference

7.83.1 Detailed Description

Declares classes **IntFunPoR** and **IntFunPoRConst**.

Author: Benno List, Jenny Böhme

Date
2005/06/29 14:40:18

Author
blist

Changelog:

- 15.7.04 JB: First version

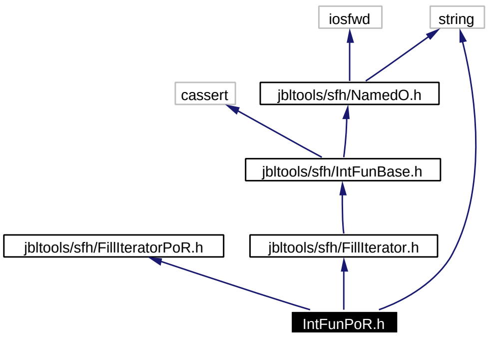
Definition in file **IntFunPoR.h**.

`#include "jbltools/sfh/FillIteratorPoR.h"`

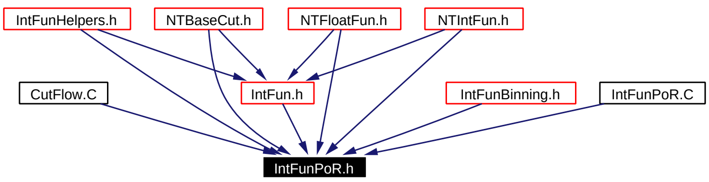
`#include "jbltools/sfh/FillIterator.h"`

`#include <string>`

Include dependency graph for IntFunPoR.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **IntFunPoR**

IntFun *Pointer or Reference.*

- class **IntFunPoRConst**

  Constant **IntFun** *Pointer or Reference.*

7.84 JBLArrayInitializer.C File Reference

7.84.1 Detailed Description

Implements helper class for initialization of global access references.

Author: Benno List, Jenny Boehme

Date

2005/06/08 06:57:41

Author

blist

Changelog:

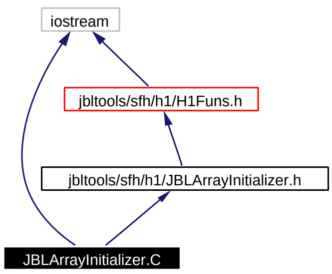
- 8.6.05 BL First version

Definition in file **JBLArrayInitializer.C**.

`#include "jbltools/sfh/h1/JBLArrayInitializer.h"`

`#include <iostream>`

Include dependency graph for JBLArrayInitializer.C:



Functions

- void **JBLArrayInitializer_message** (const char *branchname)

7.85 JBLArrayInitializer.h File Reference

7.85.1 Detailed Description

Declares helper class for initialization of global access references.

Author: Benno List, Jenny Boehme

Date

2005/06/08 07:57:19

Author

blist

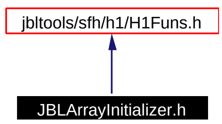
Changelog:

- 8.6.05 BL First version

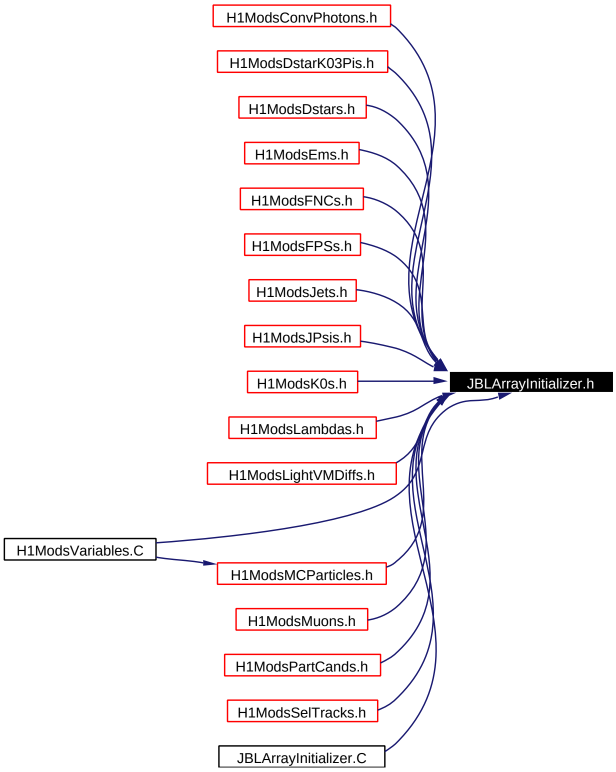
Definition in file **JBLArrayInitializer.h**.

`#include "jbltools/sfh/h1/H1Funs.h"`

Include dependency graph for JBLArrayInitializer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **JBLArrayInitializer_message** (const char *branchname)

7.86 JBLHatVariables.C File Reference

7.86.1 Detailed Description

Defines and initializes objects for H1 HAT variable access.

Important: Generate the contents using script hatvar.pl!

Author: Benno List, Jenny Boehme

Date

2006/09/07 11:01:37

Author

zimmerto

Changelog:

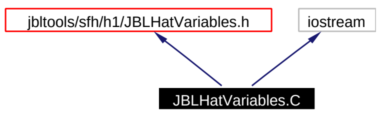
- 8.6.05: BL First version
- 10.5.06: BL, TZ: Add H1OO 2.8 version

Definition in file **JBLHatVariables.C**.

`#include "jbltools/sfh/h1/JBLHatVariables.h"`

`#include <iostream>`

Include dependency graph for JBLHatVariables.C:



7.87 JBLHatVariables.h File Reference

7.87.1 Detailed Description

Declares objects for H1 HAT variable access.

Important: Generate the contents using script hatvar.pl!

Author: Benno List, Jenny Boehme

Date
2006/09/07 11:01:37

Author
zimmerto

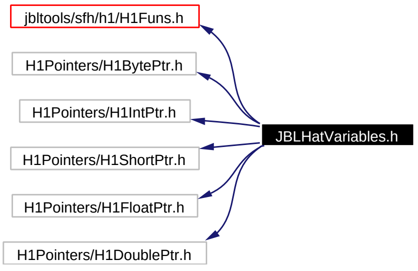
Changelog:

- 8.6.05: BL First version
- 10.5.06: BL, TZ: Add H1OO 2.8 version

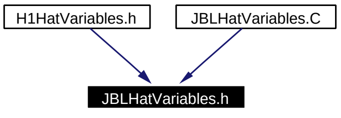
Definition in file **JBLHatVariables.h**.

```
#include "jbltools/sfh/h1/H1Funs.h"
#include "H1Pointers/H1BytePtr.h"
#include "H1Pointers/H1IntPtr.h"
#include "H1Pointers/H1ShortPtr.h"
#include "H1Pointers/H1FloatPtr.h"
#include "H1Pointers/H1DoublePtr.h"
```

Include dependency graph for JBLHatVariables.h:



This graph shows which files directly or indirectly include this file:



7.88 JBLMath.h File Reference

7.88.1 Detailed Description

Declares various useful functions.

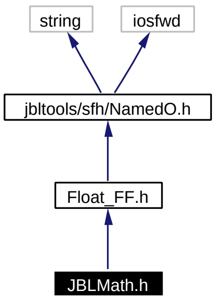
Changelog:

- 3.6.05 BL: First version (taken out of **FloatFun.h**), new function **square(int)**
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

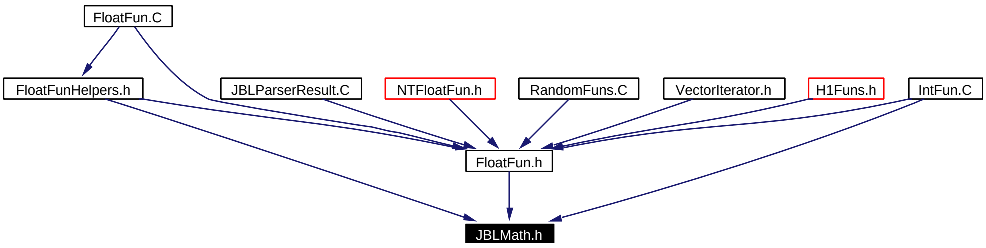
Definition in file **JBLMath.h**.

`#include "Float_FF.h"`

Include dependency graph for JBLMath.h:



This graph shows which files directly or indirectly include this file:



Functions

- Float_FF **square** (const Float_FF arg)
  The square of a Float_FF.
- int **square** (const int arg)
  The square of an int.
- Float_FF **rad** (Float_FF deg)
  Conversion from degrees to radians.

- Float_FF **deg** (Float_FF rad)

  Conversion from radians to degrees.

7.88.2 Function Documentation

7.88.2.1 Float_FF square (const Float_FF *arg*) [inline]

The square of a Float_FF.

Parameters:

arg The argument

Definition at line 16 of file JBLMath.h.

7.88.2.2 int square (const int *arg*) [inline]

The square of an int.

Parameters:

arg The argument

Definition at line 23 of file JBLMath.h.

Referenced by IntFun::operator *(), and FloatFun::operator *().

7.88.2.3 Float_FF rad (Float_FF *deg*) [inline]

Conversion from degrees to radians.

Parameters:

deg The value in degrees

Definition at line 30 of file JBLMath.h.

7.88.2.4 Float_FF deg (Float_FF *rad*) [inline]

Conversion from radians to degrees.

Parameters:

rad The value in radians

Definition at line 37 of file JBLMath.h.

7.89 JBLParserResult.C File Reference

7.89.1 Detailed Description

Implements class **JBLParserResult**.

Changelog:

- 31.5.05 BL: First version
- 8.6.05 BL: Bug fix for unary operators on constants
- 14.6.05 BL: Added automatic initialization, moved maps completely into **JBLParser-Result.C**
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

CVS Log messages:

- **Log**
  JBLParserResult.C,v
- Revision 1.11 2005/07/21 11:20:12 blist
- Use Float_FF instead of float as result type of **FloatFun**
- Revision 1.10 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **JBLParserResult.C**.

`#include "jbltools/sfh/JBLParserResult.h"`

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/FloatFun.h"`

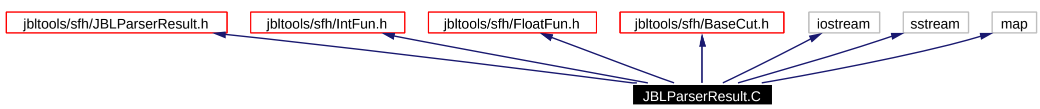
`#include "jbltools/sfh/BaseCut.h"`

`#include <iostream>`

`#include <sstream>`

`#include <map>`

Include dependency graph for JBLParserResult.C:



Namespaces

- namespace **std**

Typedefs

- typedef std::map< std::string, JBLParserResult::Fun_f_f > **Map_f_f**
- typedef std::map< std::string, JBLParserResult::Fun_f_f_f > **Map_f_f_f**
- typedef std::map< std::string, JBLParserResult::Fun_f_f_i > **Map_f_f_i**
- typedef std::map< std::string, JBLParserResult::Fun_i_i > **Map_i_i**
- typedef std::map< std::string, JBLParserResult::Fun_b_b > **Map_b_b**
- typedef std::map< std::string, JBLParserResult::Fun_FF_FF > **Map_FF_FF**
- typedef std::map< std::string, JBLParserResult::Fun_FF_FF_FF > **Map_FF_FF_FF**
- typedef std::map< std::string, JBLParserResult::Fun_FF_FF_IF > **Map_FF_FF_IF**
- typedef std::map< std::string, JBLParserResult::Fun_IF_IF > **Map_IF_IF**
- typedef std::map< std::string, JBLParserResult::Fun_BC_BC > **Map_BC_BC**

Functions

- istream & **operator>>** (istream &is, **JBLParserResult** &result)
- ostream & **operator<<** (ostream &os, const **JBLParserResult** &result)

    Output a tokenJBLParserResult.

7.90 JBLParserResult.h File Reference

7.90.1 Detailed Description

Declares class **JBLParserResult**.

Changelog:

- 31.5.05 BL: First version
- 14.6.05 BL: Added automatic initialization, moved maps completely into **JBLParser-Result.C**
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

Definition in file **JBLParserResult.h**.

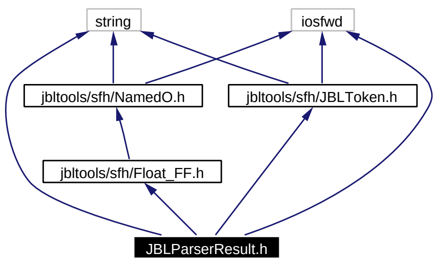
`#include "jbltools/sfh/JBLToken.h"`

`#include "jbltools/sfh/Float_FF.h"`

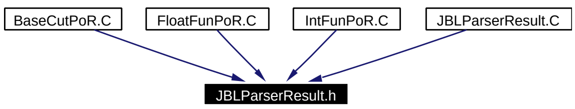
`#include <string>`

`#include <iosfwd>`

Include dependency graph for JBLParserResult.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **JBLParserResult**

  Returns the result of parsing a string that defines a function object.

Functions

- std::ostream & **operator**<< (std::ostream &os, const **JBLParserResult** &result)

Output a tokenJBLParserResult.

7.91 JBLToken.C File Reference

7.91.1 Detailed Description

Implements class **JBLToken**.

Changelog:

- 31.5.05 BL: First version

CVS Log messages:

- **Log**
  JBLToken.C,v
- Revision 1.8 2005/07/08 15:01:34 blist
- Added identity string to all .C files

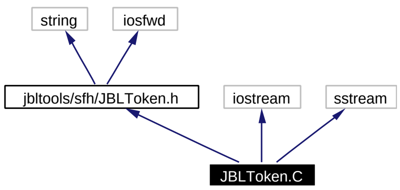
Definition in file **JBLToken.C**.

`#include "jbltools/sfh/JBLToken.h"`

`#include <iostream>`

`#include <sstream>`

Include dependency graph for JBLToken.C:



7.92 JBLToken.h File Reference

7.92.1 Detailed Description

Declares class **JBLToken**.

Changelog:

- 31.5.05 BL: First version

CVS Log messages:

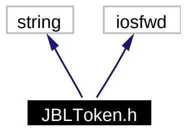
- **Log**
  JBLToken.h,v
- Revision 1.6 2005/11/22 09:00:18 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

Definition in file **JBLToken.h**.

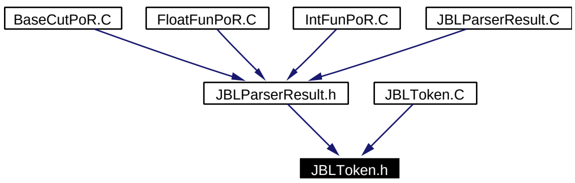
`#include <string>`

`#include <iosfwd>`

Include dependency graph for JBLToken.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **JBLToken**

  Tokeniser for string parsing.

7.93 MatrixOfH.C File Reference

7.93.1 Detailed Description

Implements class **MatrixOfH**.

Changelog:

- 13.5.05 BL, RW: Bug fix in getSet for idim=2

CVS Log messages:

- **Log**
  MatrixOfH.C,v
- Revision 1.5 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **MatrixOfH.C**.

`#include "jbltools/sfh/MatrixOfH.h"`

`#include "jbltools/sfh/SetOfHistograms.h"`

`#include "jbltools/sfh/RegH1F.h"`

`#include "jbltools/sfh/BinningFun.h"`

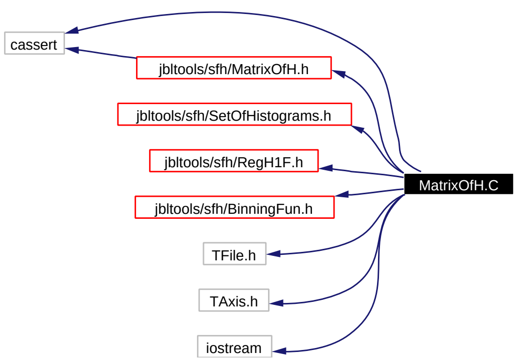
`#include <TFile.h>`

`#include <TAxis.h>`

`#include <iostream>`

`#include <cassert>`

Include dependency graph for MatrixOfH.C:



7.94 MatrixOfH.h File Reference

7.94.1 Detailed Description

Declares class **MatrixOfH**.

Changelog:

- 30.6.04 BL: Add documentation

Definition in file **MatrixOfH.h**.

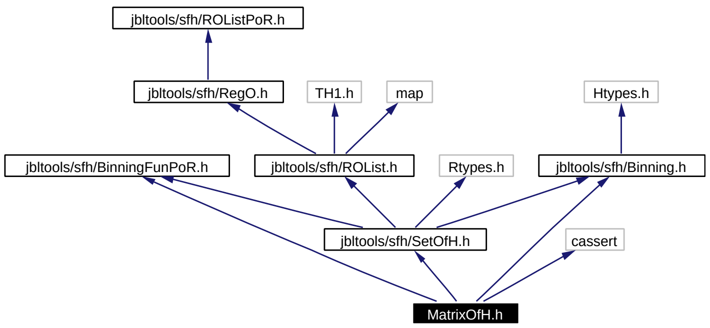
`#include "jbltools/sfh/SetOfH.h"`

`#include "jbltools/sfh/BinningFunPoR.h"`

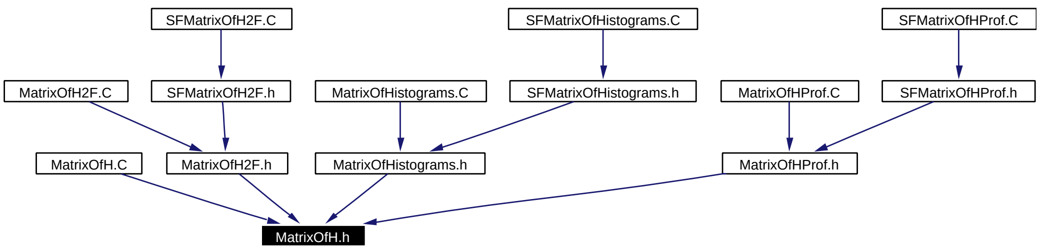
`#include "jbltools/sfh/Binning.h"`

`#include <cassert>`

Include dependency graph for MatrixOfH.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **MatrixOfH**

  A 2-dimensional collection of 1-dimensional histograms that have the same axis.

7.95 MatrixOfH2F.C File Reference

7.95.1 Detailed Description

Implements class **MatrixOfH2F**.

Changelog:

- 14.6.05 BL: First version (from **SetOfH2F**)

CVS Log messages:

- **Log**
  MatrixOfH2F.C,v
- Revision 1.3 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **MatrixOfH2F.C**.

`#include "jbltools/sfh/MatrixOfH2F.h"`

`#include "jbltools/sfh/RegH2F.h"`

`#include "jbltools/sfh/BinningFun.h"`

`#include <TFile.h>`

`#include <TF1.h>`

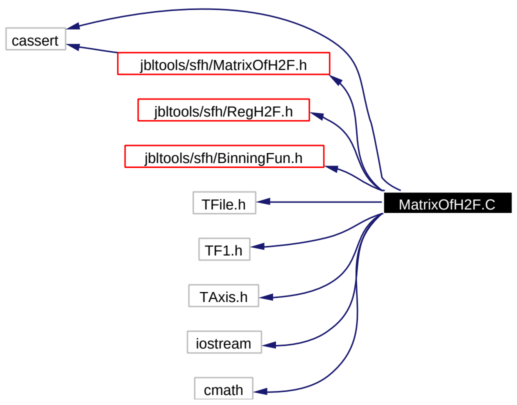
`#include <TAxis.h>`

`#include <iostream>`

`#include <cmath>`

`#include <cassert>`

Include dependency graph for MatrixOfH2F.C:



7.96 MatrixOfH2F.h File Reference

7.96.1 Detailed Description

Declares class **MatrixOfH2F**.

Changelog:

- 16.6.05 BL: First version (from **MatrixOfHProf.h**)

Definition in file **MatrixOfH2F.h**.

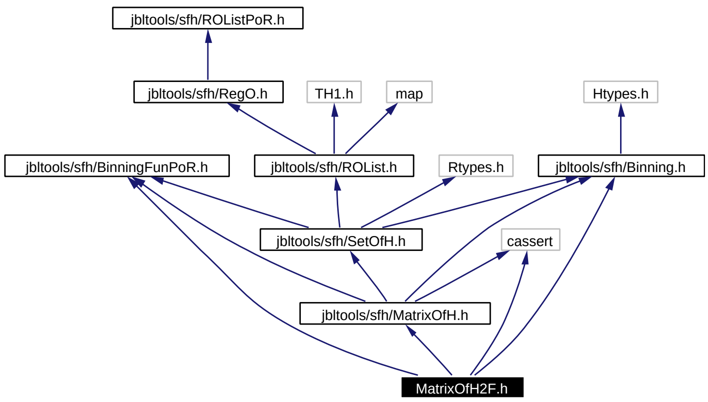
`#include "jbltools/sfh/MatrixOfH.h"`

`#include "jbltools/sfh/BinningFunPoR.h"`

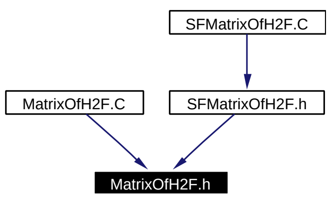
`#include "jbltools/sfh/Binning.h"`

`#include <cassert>`

Include dependency graph for MatrixOfH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **MatrixOfH2F**

  A 2-dimensional collection of 2-dimensional histograms that have the same axis.

7.97 MatrixOfHistograms.C File Reference

7.97.1 Detailed Description

Implements class **MatrixOfHistograms**.

Changelog:

- 13.5.05 BL, RW: Bug fix in getSet for idim=2

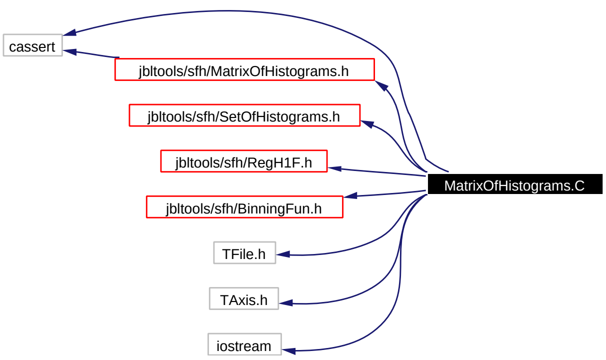
CVS Log messages:

- **Log**
  MatrixOfHistograms.C,v
- Revision 1.12 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **MatrixOfHistograms.C**.

```
#include "jbltools/sfh/MatrixOfHistograms.h"
#include "jbltools/sfh/SetOfHistograms.h"
#include "jbltools/sfh/RegH1F.h"
#include "jbltools/sfh/BinningFun.h"
#include <TFile.h>
#include <TAxis.h>
#include <iostream>
#include <cassert>
```

Include dependency graph for MatrixOfHistograms.C:



7.98 MatrixOfHistograms.h File Reference

7.98.1 Detailed Description

Declares class **MatrixOfHistograms**.

Changelog:

- 30.6.04 BL: Add documentation

Definition in file **MatrixOfHistograms.h**.

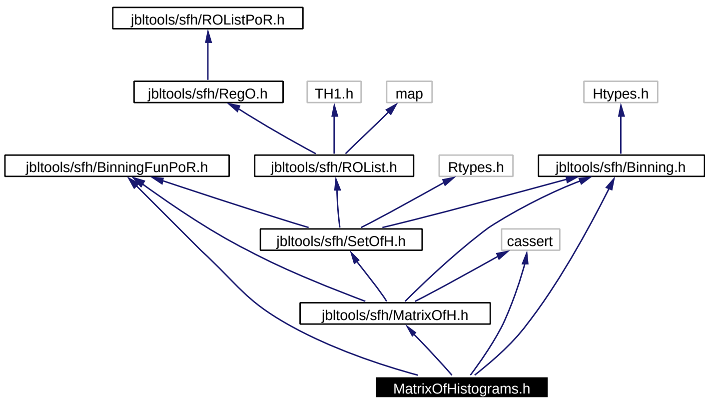
`#include "jbltools/sfh/MatrixOfH.h"`

`#include "jbltools/sfh/BinningFunPoR.h"`

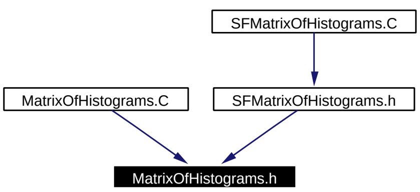
`#include "jbltools/sfh/Binning.h"`

`#include <cassert>`

Include dependency graph for MatrixOfHistograms.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **MatrixOfHistograms**

  A 2-dimensional collection of 1-dimensional histograms that have the same axis.

7.99 MatrixOfHProf.C File Reference

7.99.1 Detailed Description

Implements class **MatrixOfHProf**.

Changelog:

- 6.6.05 BL: First version (from **MatrixOfHistograms.C**)
- 7.6.05 BL: Made compilable

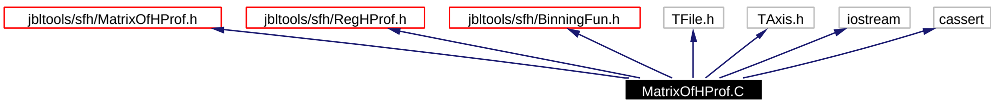
CVS Log messages:

- **Log**
  MatrixOfHProf.C,v
- Revision 1.3 2005/07/08 15:01:34 blist
- Added identity string to all .C files

Definition in file **MatrixOfHProf.C**.

```
#include "jbltools/sfh/MatrixOfHProf.h"
#include "jbltools/sfh/RegHProf.h"
#include "jbltools/sfh/BinningFun.h"
#include <TFile.h>
#include <TAxis.h>
#include <iostream>
#include <cassert>
```

Include dependency graph for MatrixOfHProf.C:



7.100 MatrixOfHProf.h File Reference

7.100.1 Detailed Description

Declares class **MatrixOfHProf**.

Changelog:

- 6.6.05 BL: First version (from **MatrixOfHistograms.h**)

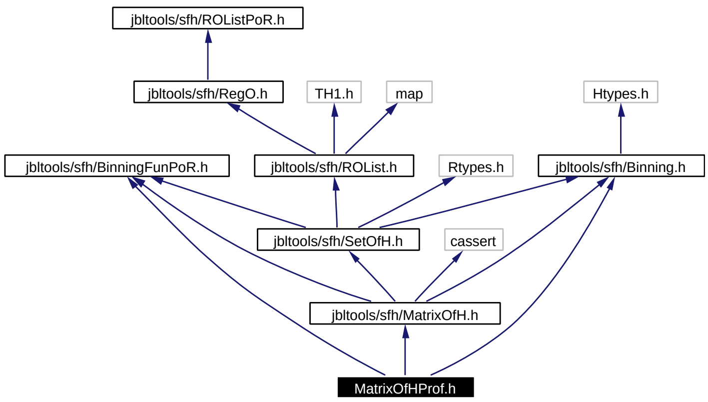
Definition in file **MatrixOfHProf.h**.

`#include "jbltools/sfh/MatrixOfH.h"`

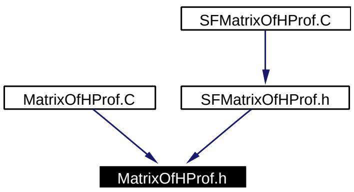
`#include "jbltools/sfh/BinningFunPoR.h"`

`#include "jbltools/sfh/Binning.h"`

Include dependency graph for MatrixOfHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **MatrixOfHProf**

  A 2-dimensional collection of 1-dimensional histograms that have the same axis.

7.101 NamedBinning.h File Reference

7.101.1 Detailed Description

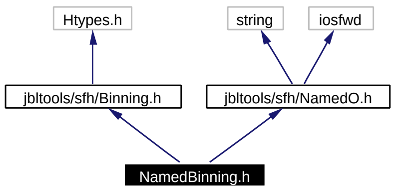
Declares class **NamedBinning**.

Definition in file **NamedBinning.h**.

`#include "jbltools/sfh/Binning.h"`

`#include "jbltools/sfh/NamedO.h"`

Include dependency graph for NamedBinning.h:



Compounds

- class **NamedBinning**

  Abstract base class for function objects that sort events into bins.

7.102 NamedO.C File Reference

7.102.1 Detailed Description

Implements class **NamedO**.

Changelog:

- 29.5.05 BL: First version
- 7.6.05 BL: Added str(double), and printing method
- 25.4.06 BL: Error message in case of overwriting a named object
- 9.5.06 BL: No assert in case of overwriting a named object
- 30.6.06 BL: No error message for duplicate names anymore

CVS Log messages:

- **Log**
  NamedO.C,v
- Revision 1.11 2006/06/30 11:05:00 blist
- Bug fix in **FloatFunBinning.C IntFunBinning.C** (segfault!)
- **SetOfHistograms** has new getSummary() method

- Revision 1.10 2006/05/09 14:30:06 blist
- No assert in case of overwriting a named object

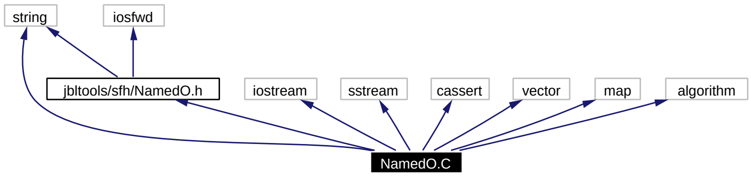
- Revision 1.9 2006/04/25 13:47:05 boehmej
- new protection against overwriting NamedOs

- Revision 1.8 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **NamedO.C**.

```
#include "jbltools/sfh/NamedO.h"
#include <iostream>
#include <sstream>
#include <cassert>
#include <vector>
#include <map>
#include <algorithm>
#include <string>
```

Include dependency graph for NamedO.C:



Typedefs

- typedef std::map< std::string, **NamedO** * > **NamedObjectsType**

7.103 NamedO.h File Reference

7.103.1 Detailed Description

Declares class **NamedO**.

Changelog:

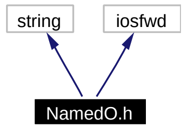
- 29.5.05 BL: First version

Definition in file **NamedO.h**.

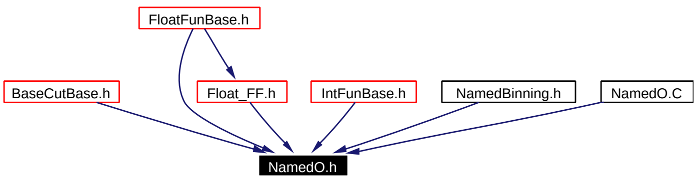
`#include <string>`

`#include <iosfwd>`

Include dependency graph for NamedO.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **NamedO**

    Base class for heap objects that have a name.

7.104 NTBaseCut.C File Reference

7.104.1 Detailed Description

Implements class **NTBaseCut**.

Changelog: 4.6.05 BL First version

CVS Log messages:

Log

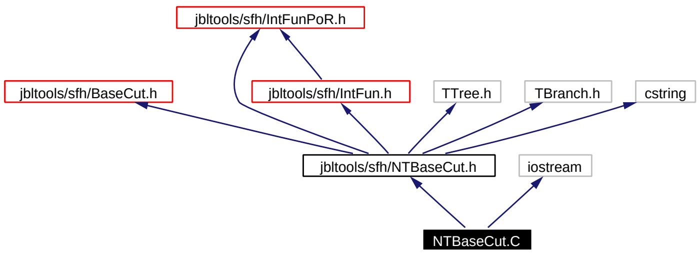
- **NTBaseCut.C**,v
- Revision 1.4 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **NTBaseCut.C**.

`#include "jbltools/sfh/NTBaseCut.h"`

`#include <iostream>`

Include dependency graph for NTBaseCut.C:



7.105 NTBaseCut.h File Reference

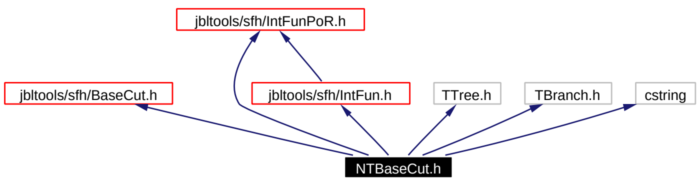
7.105.1 Detailed Description

Declares class **NTBaseCut**.

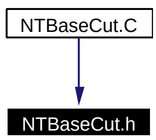
Definition in file **NTBaseCut.h**.

```
#include "jbltools/sfh/BaseCut.h"
#include "jbltools/sfh/IntFun.h"
#include "jbltools/sfh/IntFunPoR.h"
#include <TTree.h>
#include <TBranch.h>
#include <cstring>
```

Include dependency graph for NTBaseCut.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **NTBaseCut**

  Returns a bool value from an Ntuple (abstract base class for **NTBaseCutT** *templates).*

- class **NTBaseCutT**

  Returns a bool value from an Ntuple.

Functions

- template<class Ntuple, class T, unsigned int n> **NTBaseCutT**< Ntuple, T > & **ntbasecut** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_="?", const **IntFunPoR** &iter_=0)
- template<class Ntuple, class T, unsigned int n> **NTBaseCutT**< Ntuple, T > & **ntbasecut** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_="?", int offset_)

7.105.2 Function Documentation

7.105.2.1 template<class Ntuple, class T, unsigned int n> NTBaseCutT<Ntuple, T>& ntbasecut (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_* = "?", const IntFunPoR & *iter_* = 0)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 228 of file NTBaseCut.h.

7.105.2.2 template<class Ntuple, class T, unsigned int n> NTBaseCutT<Ntuple, T>& ntbasecut (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_* = "?", int *offset_*)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset

Definition at line 240 of file NTBaseCut.h.

7.106 NTFillIterator.h File Reference

7.106.1 Detailed Description

Declares class **NTFillIterator**.

Definition in file **NTFillIterator.h**.

`#include "jbltools/sfh/FillIterator.h"`

`#include <Rtypes.h>`

`#include <TTree.h>`

`#include <TBranch.h>`

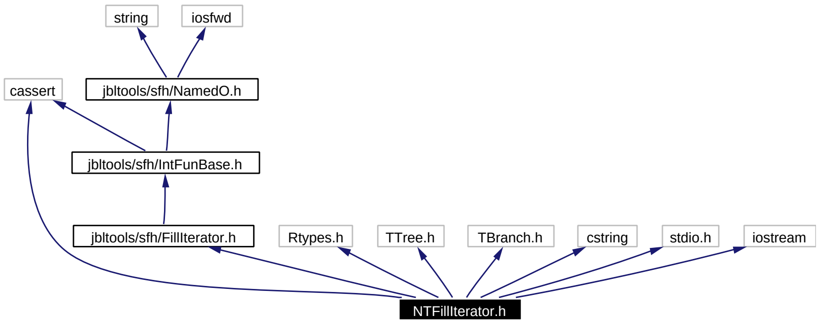
`#include <cassert>`

`#include <cstring>`

`#include <stdio.h>`

`#include <iostream>`

Include dependency graph for NTFillIterator.h:



Compounds

- class **NTFillIterator**

  A **FillIterator** *from an ntuple variable.*

7.107 NTFloatFun.C File Reference

7.107.1 Detailed Description

Implements class **NTFloatFun**.

Changelog:

- 4.6.05 BL First version
- 21.7.05 BL: Use Float_FF instead of float as result type of **FloatFun**

CVS Log messages:

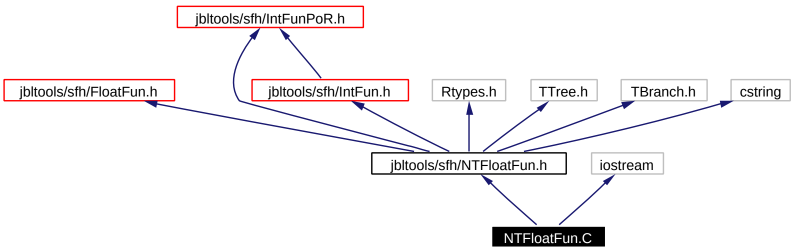
- **Log**
  NTFloatFun.C,v
- Revision 1.5 2005/07/21 11:20:12 blist
- Use Float_FF instead of float as result type of **FloatFun**
- Revision 1.4 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **NTFloatFun.C**.

`#include "jbltools/sfh/NTFloatFun.h"`

`#include <iostream>`

Include dependency graph for NTFloatFun.C:



7.108 NTFloatFun.h File Reference

7.108.1 Detailed Description

Declares class **NTFloatFun**.

Definition in file **NTFloatFun.h**.

`#include "jbltools/sfh/FloatFun.h"`

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/IntFunPoR.h"`

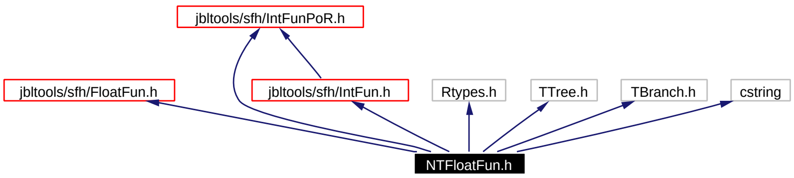
`#include <Rtypes.h>`

`#include <TTree.h>`

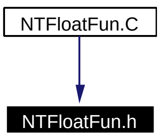
`#include <TBranch.h>`

`#include <cstring>`

Include dependency graph for NTFloatFun.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **NTFloatFun**

  Returns a Float_FF value from an Ntuple (abstract base class for **NTFloatFunT** *templates).*

- class **NTFloatFunT**

  Returns a Float_FF value from an Ntuple.

- class **NTFloatFunG**

  Returns a Float_FF value from an Ntuple getter method.

- class **NTFloatFunGA**

  Returns a Float_FF value from an Ntuple getter method with an argument.

- class **NTFloatFunGF**

Returns a Float_FF value from free getter function.

Functions

- template<class Ntuple, class T, unsigned int n> **NTFloatFunT**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_="?", const **IntFunPoR** &iter_=0)
- template<class Ntuple, class T, unsigned int n> **NTFloatFunT**< Ntuple, T > & **ntfloatfun** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_, int offset_)

7.108.2 Function Documentation

7.108.2.1 template<class Ntuple, class T, unsigned int n> NTFloatFunT<Ntuple, T>& ntfloatfun (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_* = "?", const IntFunPoR & *iter_* = 0)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 480 of file NTFloatFun.h.

7.108.2.2 template<class Ntuple, class T, unsigned int n> NTFloatFunT<Ntuple, T>& ntfloatfun (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_*, int *offset_*)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset

Definition at line 492 of file NTFloatFun.h.

7.109 NTIntFun.C File Reference

7.109.1 Detailed Description

Implements class **NTIntFun**.

Changelog: 4.6.05 BL First version

CVS Log messages:

Log

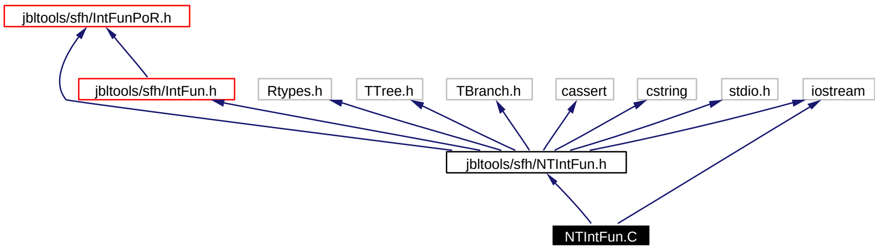
- **NTIntFun.C**,v
- Revision 1.4 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **NTIntFun.C**.

`#include "jbltools/sfh/NTIntFun.h"`

`#include <iostream>`

Include dependency graph for NTIntFun.C:



7.110 NTIntFun.h File Reference

7.110.1 Detailed Description

Declares class **NTIntFun**.

Definition in file **NTIntFun.h**.

`#include "jbltools/sfh/IntFun.h"`

`#include "jbltools/sfh/IntFunPoR.h"`

`#include <Rtypes.h>`

`#include <TTree.h>`

`#include <TBranch.h>`

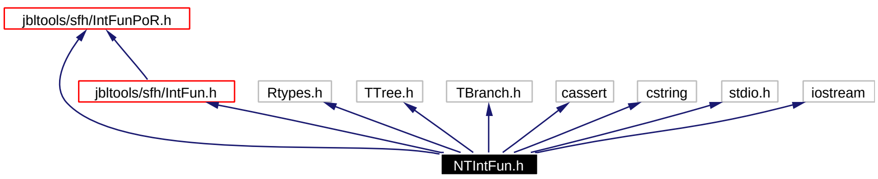
`#include <cassert>`

`#include <cstring>`

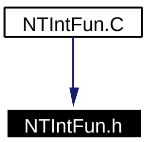
`#include <stdio.h>`

`#include <iostream>`

Include dependency graph for NTIntFun.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **NTIntFun**

  Returns a int value from an Ntuple (abstract base class for **NTIntFunT** *templates).*

- class **NTIntFunT**

  Returns a int value from an Ntuple.

Functions

- template<class Ntuple, class T, unsigned int n> **NTIntFunT**< Ntuple, T > & **ntintfun** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_="?", const **IntFunPoR** &iter_=0)
- template<class Ntuple, class T, unsigned int n> **NTIntFunT**< Ntuple, T > & **ntintfun** (const Ntuple &nt, T(Ntuple::*p_mem)[n], const char *name_="?", int offset_)

7.110.2 Function Documentation

7.110.2.1 template<class Ntuple, class T, unsigned int n> NTIntFunT<Ntuple, T>& ntintfun (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_* = "?", const IntFunPoR & *iter_* = 0)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

iter_ An optional iterator

Definition at line 236 of file NTIntFun.h.

7.110.2.2 template<class Ntuple, class T, unsigned int n> NTIntFunT<Ntuple, T>& ntintfun (const Ntuple & *nt*, T Ntuple::* *p_mem*[n], const char * *name_* = "?", int *offset_*)

Parameters:

nt The ntuple

p_mem Member pointer

name_ The object's name

offset_ An offset

Definition at line 247 of file NTIntFun.h.

7.111 RandomFuns.C File Reference

7.111.1 Detailed Description

Implements classes for Random function objects.

Changelog:

- 22.11.05 BL: First version
- 29.11.05 BL: Added underflow and overflow bins to **HistoRandomFun** and **HistoRandomIntFun**

CVS Log messages:

- **Log**
  RandomFuns.C,v
- Revision 1.5 2005/11/29 20:03:31 blist
- Added underflow and overflow bins to **HistoRandomFun** and **HistoRandomIntFun**
- Revision 1.4 2005/11/23 12:11:33 blist
- index in **RandomFun** now determines number of iterations
- Revision 1.3 2005/11/22 21:57:05 blist
- New class **HistoRandomIntFun**, with test
- Revision 1.2 2005/11/22 20:36:18 blist
- Better random number generation
- Revision 1.1 2005/11/22 09:00:18 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

Definition in file **RandomFuns.C**.

`#include "jbltools/sfh/RandomFuns.h"`

`#include "jbltools/sfh/IntFun.h"`

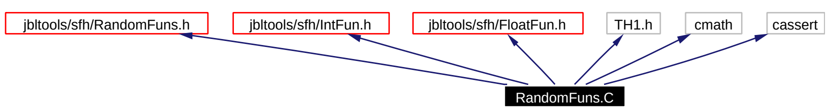
`#include "jbltools/sfh/FloatFun.h"`

`#include "TH1.h"`

`#include <cmath>`

`#include <cassert>`

Include dependency graph for RandomFuns.C:



7.111.2 Function Documentation

7.111.2.1 IntFun& poisson (const FloatFunPoR & *rnd*, double *mean*) `[related]`

Parameters:

rnd A random function

mean The mean value

Definition at line 308 of file RandomFuns.C.

References FloatFunPoR::pff.

7.112 RandomFuns.h File Reference

7.112.1 Detailed Description

Declares classes for Random function objects.

Changelog:

- 22.11.05 BL: First version

CVS Log messages:

- **Log**
  RandomFuns.h,v
- Revision 1.2 2005/11/22 21:57:05 blist
- New class **HistoRandomIntFun**, with test
- Revision 1.1 2005/11/22 09:00:18 blist
- Added class **HistoRandomFun**, new files **RandomFuns.h**, .C
-

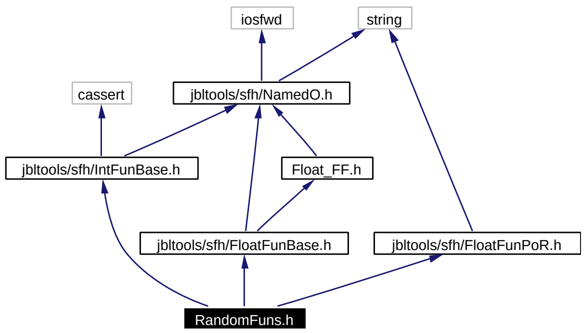
Definition in file **RandomFuns.h**.

`#include "jbltools/sfh/IntFunBase.h"`

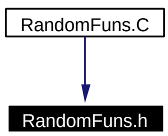
`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

Include dependency graph for RandomFuns.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RandomFun**

  A **FloatFun** *that returns a Random value, taking an intfun as seed.*

- class **GaussRandomFun**

  A **FloatFun** *that returns a gaussian distributed value from a random number.*

- class **Poisson1Fun**

  A **IntFun** *that returns a Poisson-distributed value from a uniform random number.*

- class **Poisson2Fun**

  A **IntFun** *that returns a Poisson-distributed value from a uniform random number (gaussian approximation).*

- class **HistoRandomFun**

  A **FloatFun** *that returns a random number distributed according to a histogram.*

- class **HistoRandomIntFun**

  A **FloatFun** *that returns a random number distributed according to a histogram.*

7.113 RegH1F.C File Reference

7.113.1 Detailed Description

Implements class **RegH1F**.

Author: Benno List, Jenny Böhme

Date
2005/11/11 17:21:16

Author
rweber

Changelog:

- 4.6.04 BL: Add new constructor using a Binning object
- 8.6.04 BL: Use HListPoR, register histos through RegH constructor
- 12.5.05 BL: Added Divide method, call SumW2 in all constructors!
- 13.7.05 BL: Added possibility of alphanumeric bin labels
- 11.11.05 RW: Added DrawFunc to draw specific Fit Function

CVS Log messages:

- **Log**
  RegH1F.C,v
- Revision 1.15 2005/11/11 17:21:16 rweber
- Added DrawFunc to **RegH1F** and **ROList** to plot selected Fit Function
- In **ROList::Draw** routines: Changed 'bool log' to 'Int_t log' for more logarithimc options:
- 0 = none, 1 = y-axis in log scale, 2 = x-axis in log, 3 = x- and y- axis in log

- Revision 1.14 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**

- Revision 1.13 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **RegH1F.C**.

`#include "jbltools/sfh/RegH1F.h"`

`#include "jbltools/sfh/Binning.h"`

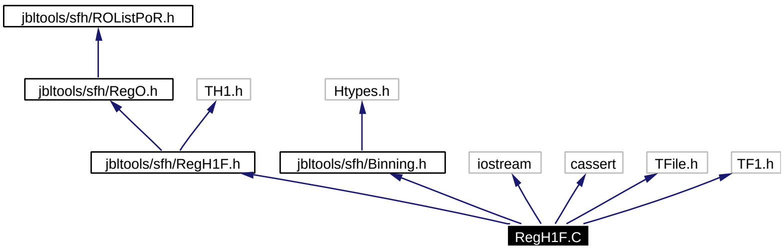
`#include <iostream>`

`#include <cassert>`

`#include <TFile.h>`

`#include <TF1.h>`

Include dependency graph for RegH1F.C:



7.114 RegH1F.h File Reference

7.114.1 Detailed Description

Declares class **RegH1F**.

Changelog:

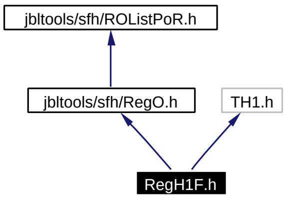
- 9.6.04 BL: Added documentation
- 10.6.04 BL: Derived from **RegO** instead of RegH
- 12.5.05 BL: Added Divide method
- 15.6.05 BL: Signature of the Divide method corrected (const TH1∗)
- 11.11.05 RW: Added DrawFunc to draw specific Fit Function

Definition in file **RegH1F.h**.

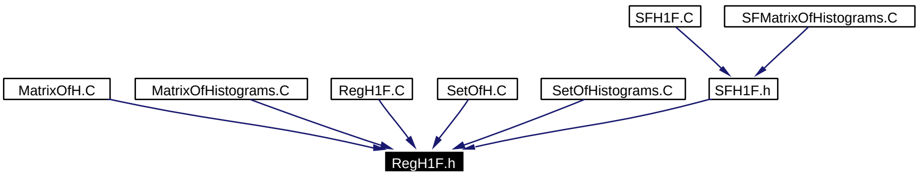
`#include "jbltools/sfh/RegO.h"`

`#include <TH1.h>`

Include dependency graph for RegH1F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RegH1F**

  A registered 1-dimensional TH1F histogram.

7.115 RegH2F.C File Reference

7.115.1 Detailed Description

Implements class **RegH2F**.

Author: Jenny Böhme, Benno List

Date

2005/07/13 11:35:50

Author

blist

Changelog:

- 9.6.04 BL: Use HListPoR, added constructor with Binning objects
- 14.6.05 BL: Call Sumw2() in all constructors
- 13.7.05 BL: Added possibility of alphanumeric bin labels

CVS Log messages:

- **Log**
  RegH2F.C,v
- Revision 1.7 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**
- Revision 1.6 2005/07/08 15:01:35 blist
- Added identity string to all .C files

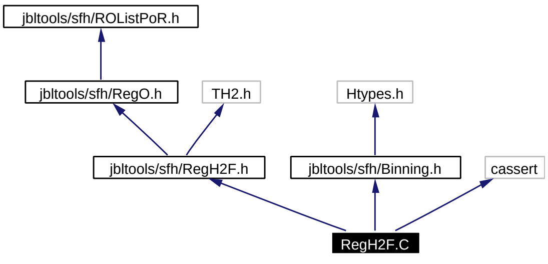
Definition in file **RegH2F.C**.

`#include "jbltools/sfh/RegH2F.h"`

`#include "jbltools/sfh/Binning.h"`

`#include <cassert>`

Include dependency graph for RegH2F.C:



7.116 RegH2F.h File Reference

7.116.1 Detailed Description

Declares class **RegH2F**.

Changelog:

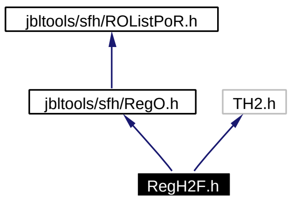
- 10.6.04 BL: Derived from **RegO** instead of RegH
- 9.6.04 BL: Add documentation

Definition in file **RegH2F.h**.

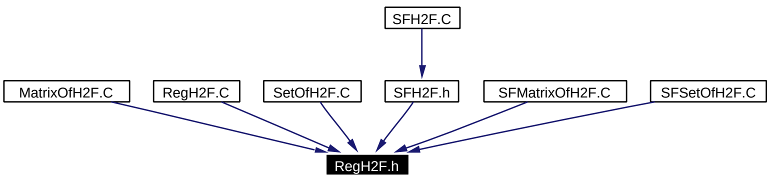
`#include "jbltools/sfh/RegO.h"`

`#include <TH2.h>`

Include dependency graph for RegH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RegH2F**

  A registered 2-dimensional TH2F histogram.

7.117 RegHProf.C File Reference

7.117.1 Detailed Description

Implements class **RegHProf**.

Author: Jenny Böhme, Benno List

Date

2005/07/13 11:35:50

Author

blist

Changelog:

- 9.6.04 BL: Use HListPoR, added constructor with Binning objects
- 14.6.05 BL: Call Sumw2() in all constructors
- 13.7.05 BL: Added possibility of alphanumeric bin labels

CVS Log messages:

Log

- **RegHProf.C**,v
- Revision 1.7 2005/07/13 11:35:50 blist
- Alphanumeric labels possible for Binning, implemented for **CutBinning**
- Revision 1.6 2005/07/08 15:01:35 blist
- Added identity string to all .C files

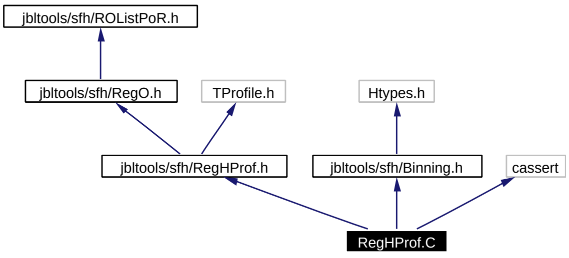
Definition in file **RegHProf.C**.

`#include "jbltools/sfh/RegHProf.h"`

`#include "jbltools/sfh/Binning.h"`

`#include <cassert>`

Include dependency graph for RegHProf.C:



7.118 RegHProf.h File Reference

7.118.1 Detailed Description

Declares class **RegHProf**.

Changelog:

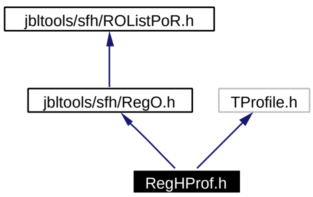
- 9.6.04 BL: Add documentation
- 10.6.04 BL: Derived from **RegO** instead of RegH

Definition in file **RegHProf.h**.

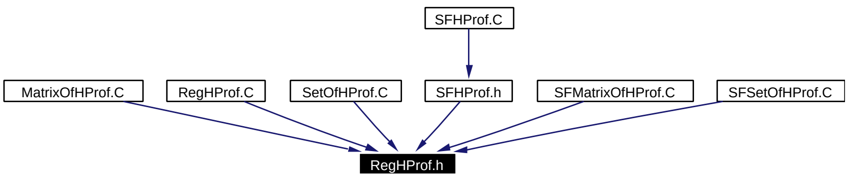
`#include "jbltools/sfh/RegO.h"`

`#include <TProfile.h>`

Include dependency graph for RegHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RegHProf**

  A registered 1-dimensional TProfile histogram.

7.119 RegO.C File Reference

7.119.1 Detailed Description

Implements class **RegO**.

Author: Benno List, Jenny Böhme

Date

2005/07/08 15:01:35

Author

blist

Changelog:

- 10.6.06 BL First version (adapted from RegH.C)

CVS Log messages:

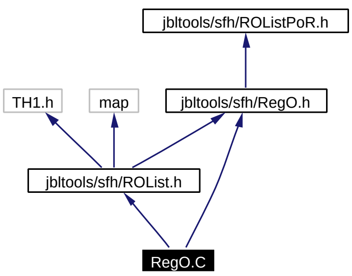
- **Log**
  RegO.C,v
- Revision 1.2 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **RegO.C**.

`#include "jbltools/sfh/RegO.h"`

`#include "jbltools/sfh/ROList.h"`

Include dependency graph for RegO.C:



7.120 RegO.h File Reference

7.120.1 Detailed Description

Declares class **RegO**.

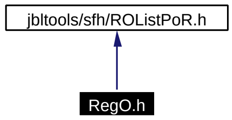
Changelog:

- 10.6.06 BL First version (adapted from RegH.h)

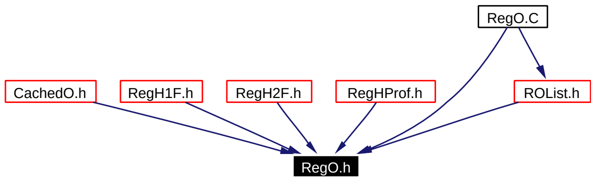
Definition in file **RegO.h**.

`#include "jbltools/sfh/ROListPoR.h"`

Include dependency graph for RegO.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **RegO**

  Mix-in class for registered objects.

7.121 ROList.C File Reference

7.121.1 Detailed Description

Implements class **ROList**.

Changelog:

- 10.6.06 BL First version (adapted from HList.C)
- 27.5.05 BL: Added "recursive" flag to Draw(...)
- 11.11.05 RW: Added DrawFunc to draw specific Fit Function
- 15.5.06 TZ: Added "backward" flag to DrawCutFlow

CVS Log messages:

- **Log**
  ROList.C,v
- Revision 1.15 2006/05/15 13:28:19 zimmerto
- 15.5.06 TZ: Added flag "backward" to **ROList::DrawCutFlow**. This is done in
- a way which should not affect any existing programs.

- Revision 1.14 2005/11/11 17:21:15 rweber
- Added DrawFunc to **RegH1F** and **ROList** to plot selected Fit Function
- In **ROList::Draw** routines: Changed 'bool log' to 'Int_t log' for more logarithimc options:
- 0 = none, 1 = y-axis in log scale, 2 = x-axis in log, 3 = x- and y- axis in log

- Revision 1.13 2005/09/29 08:56:40 boehmej
- bugfix in **Binning.C**

- Revision 1.12 2005/07/11 08:27:35 blist
- New method DrawCutFlow in **ROList**

- Revision 1.11 2005/07/08 15:01:35 blist
- Added identity string to all .C files

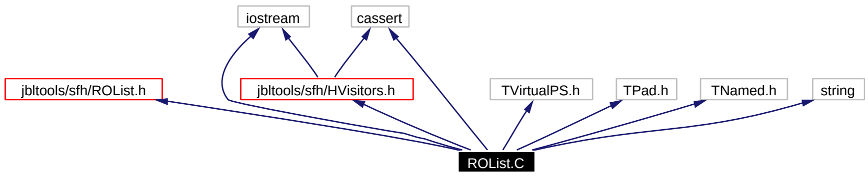
Definition in file **ROList.C**.

```
#include "jbltools/sfh/ROList.h"
#include "jbltools/sfh/HVisitors.h"
#include <TVirtualPS.h>
#include <TPad.h>
#include <TNamed.h>
#include <iostream>
```

`#include <cassert>`

`#include <string>`

Include dependency graph for ROList.C:



7.122 ROList.h File Reference

7.122.1 Detailed Description

Declares class **ROList**.

Changelog:

- 10.6.06 BL First version (adapted from HList.h)
- 11.11.05 RW: Added DrawFunc to draw specific Fit Function
- 15.5.06 TZ: Added flag 'backward' to DrawCutFlow

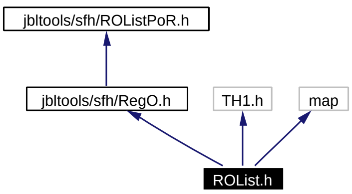
Definition in file **ROList.h**.

`#include "jbltools/sfh/RegO.h"`

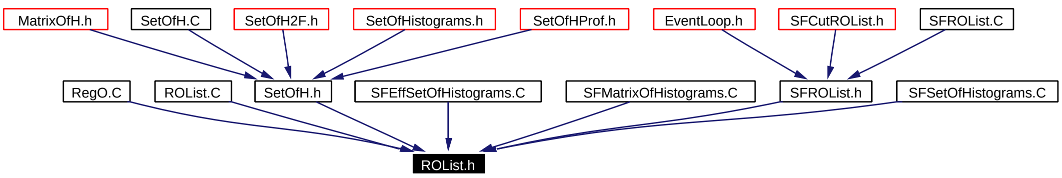
`#include <TH1.h>`

`#include <map>`

Include dependency graph for ROList.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **ROList**

  *Stores a List of registered Histos (***RegO***), also recursively.*

7.123 ROListPoR.h File Reference

7.123.1 Detailed Description

Declares classes **ROListPoR** and **ROListPoRConst**.

Author: Benno List, Jenny Böhme

Date
2004/06/29 13:03:38

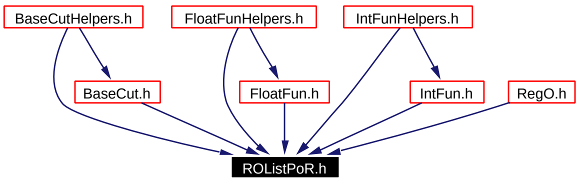
Author
blist

Changelog:

- 10.6.06 BL First version (adapted from HListPoR.h)

Definition in file **ROListPoR.h**.

This graph shows which files directly or indirectly include this file:



Compounds

- class **ROListPoR**
  ROList *Pointer or Reference.*
- class **ROListPoRConst**
  Constant **ROList** *Pointer or Reference.*

7.124 SetOfH.C File Reference

7.124.1 Detailed Description

Implements class **SetOfH**.

Changelog:

- 21.5.05 BL: First version, taken from SetOgHProf
- 21.5.05 BL: Protect genBinName and genBinTitle against zero return value from **BinningFun::getBinName** and **BinningFun::getBinTitle**, allow values of i greater than given by BinningFun::getNBins.
- 27.5.05 BL: Use iterators to loop over entries
- 17.6.05 BL: Bug fix in Add
- 6.9.05 BL: Bug fix in Divide for three Sets

CVS Log messages:

Log

- **SetOfH.C**,v
- Revision 1.8 2005/09/06 11:27:25 blist
- Bug fix in **SetOfH::Divide** for three **SetOfH**
- Revision 1.7 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SetOfH.C**.

```
#include "jbltools/sfh/SetOfH.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/RegH1F.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SetOfH.C:

7.125 SetOfH.h File Reference

7.125.1 Detailed Description

Declares class **SetOfH**.

Changelog:

- 21.5.05 BL: First version (taken from **SetOfHProf**)

Definition in file **SetOfH.h**.

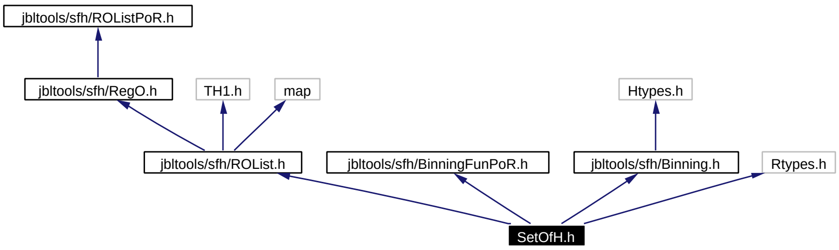
`#include "jbltools/sfh/ROList.h"`

`#include "jbltools/sfh/BinningFunPoR.h"`

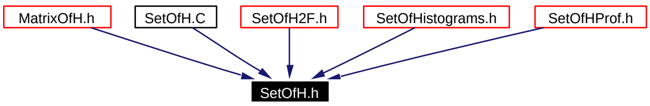
`#include "jbltools/sfh/Binning.h"`

`#include <Rtypes.h>`

Include dependency graph for SetOfH.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SetOfH**

    Base class for collection of histograms that have the same axis/axes.

7.126 SetOfH2F.C File Reference

7.126.1 Detailed Description

Implements class **SetOfH2F**.

Changelog:

- 10.6.05 BL: First version

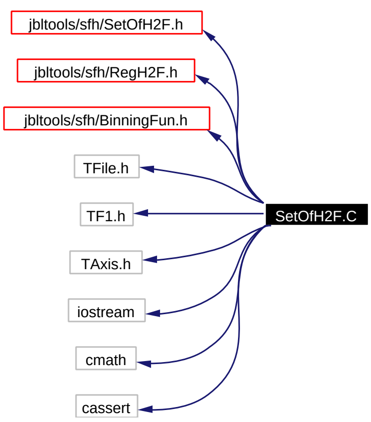
CVS Log messages:

- **Log**
  SetOfH2F.C,v
- Revision 1.2 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SetOfH2F.C**.

```
#include "jbltools/sfh/SetOfH2F.h"
#include "jbltools/sfh/RegH2F.h"
#include "jbltools/sfh/BinningFun.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SetOfH2F.C:



7.127 SetOfH2F.h File Reference

7.127.1 Detailed Description

Declares class **SetOfH2F**.

Changelog:

- 13.5.05 BL: First version

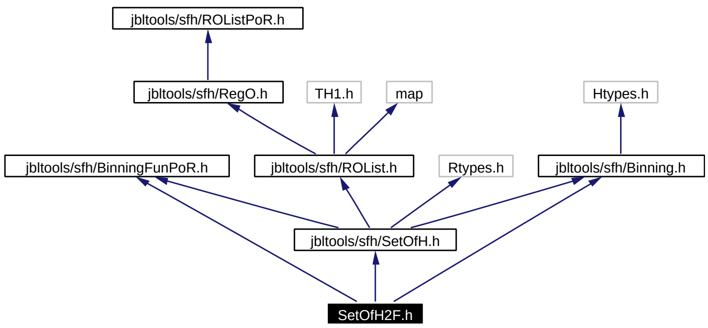
Definition in file **SetOfH2F.h**.

`#include "jbltools/sfh/SetOfH.h"`

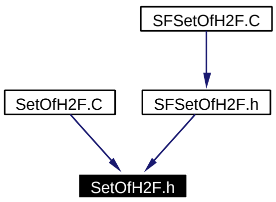
`#include "jbltools/sfh/BinningFunPoR.h"`

`#include "jbltools/sfh/Binning.h"`

Include dependency graph for SetOfH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SetOfH2F**

  A collection of profile histograms that have the same axis.

7.128 SetOfHistograms.C File Reference

7.128.1 Detailed Description

Implements class **SetOfHistograms**.

Changelog:

- 21.5.05 BL: Protect genBinName and genBinTitle against zero return value from **BinningFun::getBinName** and **BinningFun::getBinTitle**, allow values of i greater than given by BinningFun::getNBins.

CVS Log messages:

- **Log**
  SetOfHistograms.C,v

- Revision 1.22 2006/06/30 11:05:00 blist

- Bug fix in **FloatFunBinning.C** **IntFunBinning.C** (segfault!)

- **SetOfHistograms** has new getSummary() method

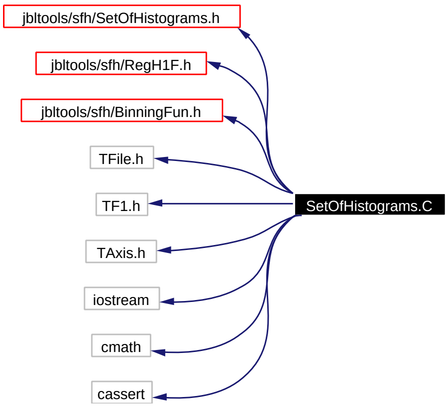
- Revision 1.21 2005/07/08 15:01:35 blist

- Added identity string to all .C files

Definition in file **SetOfHistograms.C**.

```
#include "jbltools/sfh/SetOfHistograms.h"
#include "jbltools/sfh/RegH1F.h"
#include "jbltools/sfh/BinningFun.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SetOfHistograms.C:


jbltools/sfh/SetOfHistograms.h
jbltools/sfh/RegH1F.h
jbltools/sfh/BinningFun.h
TFile.h
TF1.h
TAxis.h
iostream
cmath
cassert
SetOfHistograms.C

7.129 SetOfHistograms.h File Reference

7.129.1 Detailed Description

Declares class **SetOfHistograms**.

Changelog:

- 22.6.04 BL: Added more documentation
- 1.7.04 BL: Finalized documentation
- 27.5.05 BL: Access private members of **SetOfH** through accessors

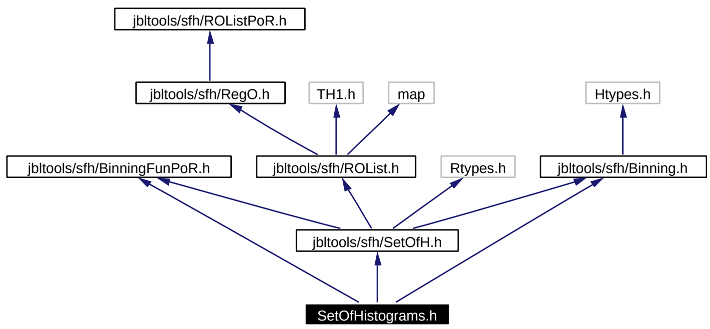
Definition in file **SetOfHistograms.h**.

`#include "jbltools/sfh/SetOfH.h"`

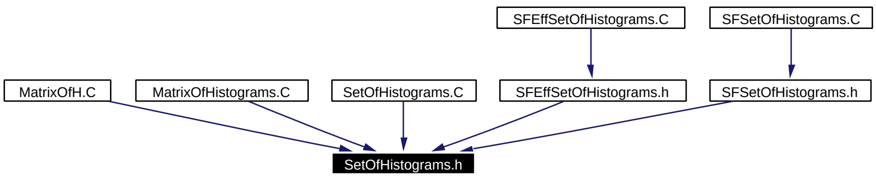
`#include "jbltools/sfh/BinningFunPoR.h"`

`#include "jbltools/sfh/Binning.h"`

Include dependency graph for SetOfHistograms.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SetOfHistograms**

  A collection of 1-dimensional histograms that have the same axis.

7.130 SetOfHProf.C File Reference

7.130.1 Detailed Description

Implements class **SetOfHProf**.

Changelog:

- 21.5.05 BL: Protect genBinName and genBinTitle against zero return value from **BinningFun::getBinName** and **BinningFun::getBinTitle**, allow values of i greater than given by BinningFun::getNBins.
- 27.5.05 BL: Access private members of **SetOfH** through accessors

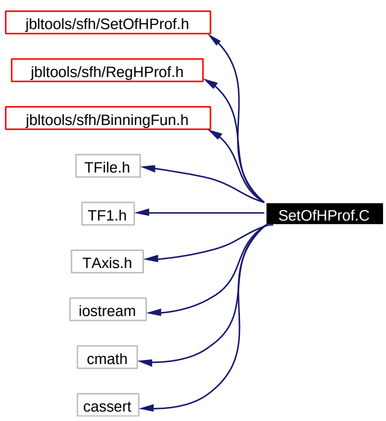
CVS Log messages:

- **Log**
  SetOfHProf.C,v
- Revision 1.6 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SetOfHProf.C**.

```
#include "jbltools/sfh/SetOfHProf.h"
#include "jbltools/sfh/RegHProf.h"
#include "jbltools/sfh/BinningFun.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SetOfHProf.C:



7.131 SetOfHProf.h File Reference

7.131.1 Detailed Description

Declares class **SetOfHProf**.

Changelog:

- 13.5.05 BL: First version

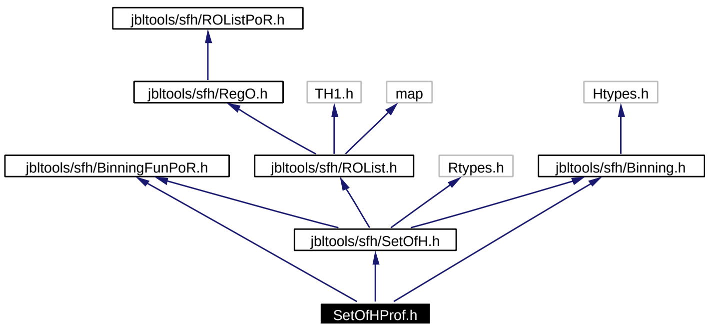
Definition in file **SetOfHProf.h**.

`#include "jbltools/sfh/SetOfH.h"`

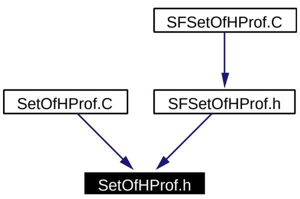
`#include "jbltools/sfh/BinningFunPoR.h"`

`#include "jbltools/sfh/Binning.h"`

Include dependency graph for SetOfHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SetOfHProf**

  A collection of profile histograms that have the same axis.

7.132 SFCutROList.C File Reference

7.132.1 Detailed Description

Implements class **SFCutROList**.

Changelog:

- 10.6.04 BL: First version, adapted from SFHList.C
- 25.4.06 BL: Bug fix: Ignore iterator dependent cut in setCut

CVS Log messages:

- **Log**
  SFCutROList.C,v
- Revision 1.2 2006/04/25 13:47:54 boehmej
- bugfix in **SFCutROList::setCut**
- Revision 1.1 2005/07/18 07:02:08 blist
- New class **SFCutROList**

Definition in file **SFCutROList.C**.

`#include "jbltools/sfh/SFCutROList.h"`

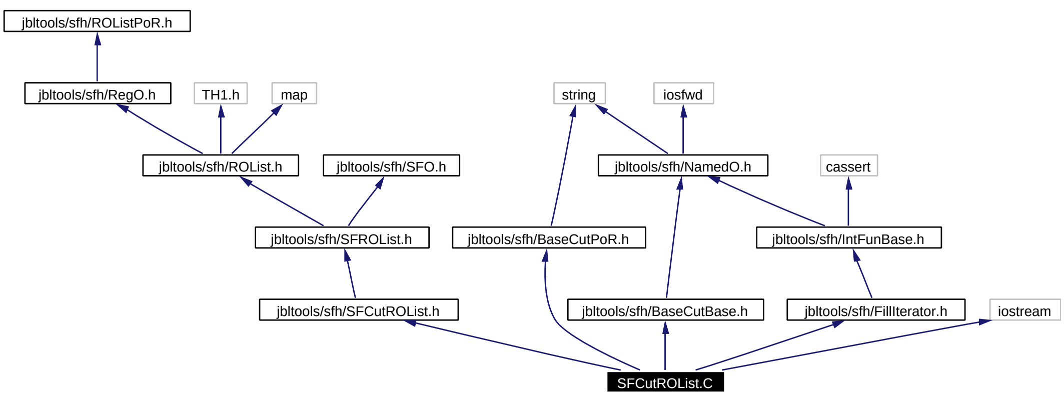
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include <iostream>`

Include dependency graph for SFCutROList.C:



7.133 SFCutROList.h File Reference

7.133.1 Detailed Description

Declares class **SFCutROList**.

Changelog:

- 18.7.05 BL: First version, adapted from **SFROList.h**

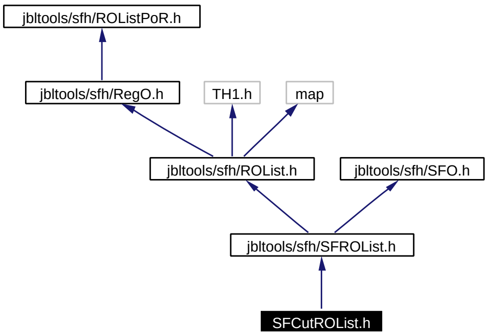
CVS Log messages:

- **Log**
  SFCutROList.h,v
- Revision 1.2 2005/07/18 07:04:25 blist
- CVS log in **SFCutROList.h** shifted
- Revision 1.1 2005/07/18 07:02:08 blist
- New class **SFCutROList**

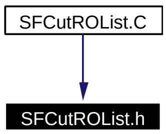
Definition in file **SFCutROList.h**.

`#include "jbltools/sfh/SFROList.h"`

Include dependency graph for SFCutROList.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFCutROList**
  A list of Self-Filling objects with a cut.

7.134 SFEffSetOfHistograms.C File Reference

7.134.1 Detailed Description

Implements class **SFEffSetOfHistograms**.

CVS Log messages:

- **Log**
  SFEffSetOfHistograms.C,v
- Revision 1.4 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFEffSetOfHistograms.C**.

`#include "jbltools/sfh/SFEffSetOfHistograms.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/BinningFun.h"`

`#include "jbltools/sfh/ROList.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include <TVirtualPS.h>`

`#include <TPad.h>`

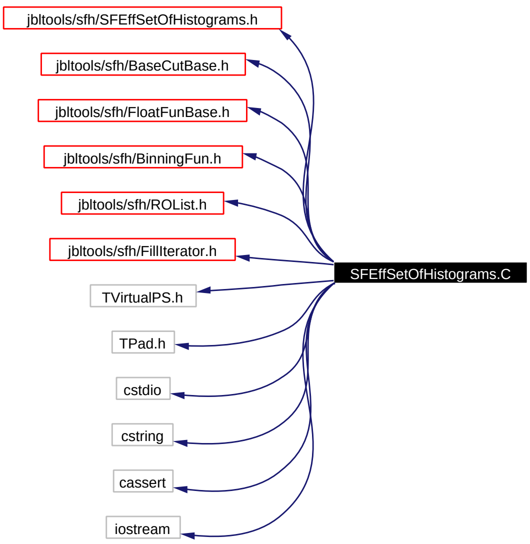
`#include <cstdio>`

`#include <cstring>`

`#include <cassert>`

`#include <iostream>`

Include dependency graph for SFEffSetOfHistograms.C:



7.135 SFEffSetOfHistograms.h File Reference

7.135.1 Detailed Description

Declares class **SFEffSetOfHistograms**.

Changelog:

- 14.7.04 BL First version

Definition in file **SFEffSetOfHistograms.h**.

`#include "jbltools/sfh/SetOfHistograms.h"`

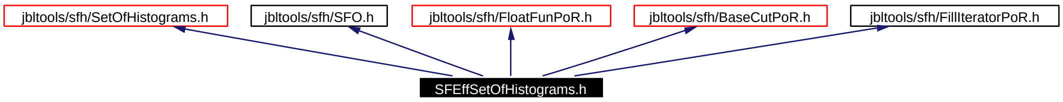
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

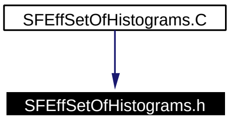
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFEffSetOfHistograms.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFEffSetOfHistograms**

  A self-filling collection of 1-D histos with the same axis for efficiency calculation.

7.136 SFH1F.C File Reference

7.136.1 Detailed Description

Implements class **SFH1F**.

Author: Benno List, Jenny Böhme

Date
2005/08/23 12:05:44

Author
blist

Changelog:

- 4.6.04 BL: Add new constructor using a Binning object
- 8.6.04 BL: Use HListPoR
- 23.8.05 BL: Use Float_FF for weight in Fill()

CVS Log messages:

- **Log**
  SFH1F.C,v
- Revision 1.16 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.15 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFH1F.C**.

`#include "jbltools/sfh/SFH1F.h"`

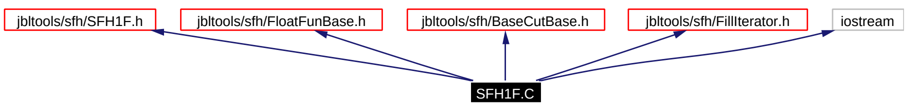
`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include <iostream>`

Include dependency graph for SFH1F.C:



7.137 SFH1F.h File Reference

7.137.1 Detailed Description

Declares class **SFH1F**.

Changelog:

- 9.6.04 BL: Added more documentation
- 21.7.04 BL: Introduced checkTwoIterators, checkOneIterator

Definition in file **SFH1F.h**.

`#include "jbltools/sfh/SFO.h"`

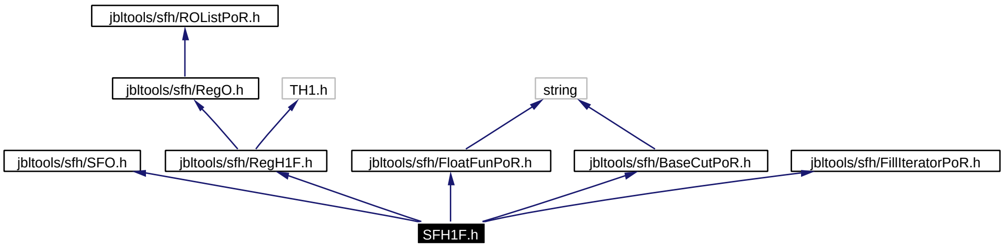
`#include "jbltools/sfh/RegH1F.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

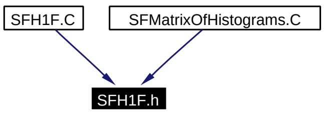
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFH1F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFH1F**

  A self-filling 1-dimensional TH1F histogram.

7.138 SFH2F.C File Reference

7.138.1 Detailed Description

Implements class **SFH2F**.

Author: Jenny Böhme, Benno List

Date

2005/08/23 12:05:44

Author

blist

Changelog:

- 9.6.04 BL: documentation ugraded, added constructor with Binning objects
- 23.8.05 BL: Use Float_FF for weight in Fill()

CVS Log messages:

- **Log**
  SFH2F.C,v
- Revision 1.13 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.12 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFH2F.C**.

`#include "jbltools/sfh/SFH2F.h"`

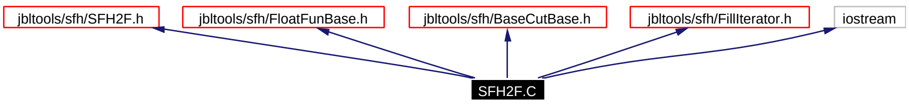
`#include "jbltools/sfh/FloatFunBase.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include <iostream>`

Include dependency graph for SFH2F.C:



7.139 SFH2F.h File Reference

7.139.1 Detailed Description

Declares class **SFH2F**.

Changelog:

- 9.6.04 BL: Add documentation

Definition in file **SFH2F.h**.

`#include "jbltools/sfh/SFO.h"`

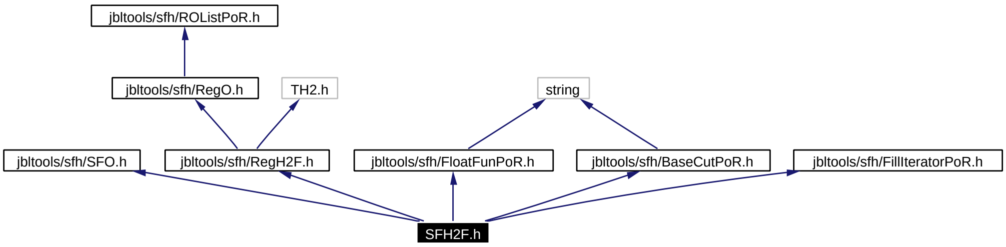
`#include "jbltools/sfh/RegH2F.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

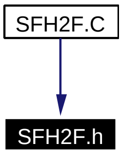
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFH2F**

  A self-filling 2-dimensional TH2F histogram.

7.140 SFHProf.C File Reference

7.140.1 Detailed Description

Implements class **SFHProf**.

Author: Jenny Böhme, Benno List

Date

2005/08/23 12:05:44

Author

blist

Changelog:

- 9.6.04 BL: Use HListPoR, **FloatFunPoR**, **BaseCutPoR**, **FillIteratorPoR**, documentation ugraded,
- 23.8.05 BL: Use Float_FF for weight in Fill()

CVS Log messages:

- **Log**
  SFHProf.C,v
- Revision 1.13 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)
- Revision 1.12 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFHProf.C**.

`#include "jbltools/sfh/SFHProf.h"`

`#include "jbltools/sfh/FloatFunBase.h"`

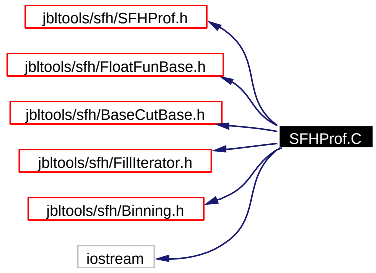
`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include "jbltools/sfh/Binning.h"`

`#include <iostream>`

Include dependency graph for SFHProf.C:


jbltools/sfh/SFHProf.h
jbltools/sfh/FloatFunBase.h
jbltools/sfh/BaseCutBase.h
jbltools/sfh/FillIterator.h
jbltools/sfh/Binning.h
iostream
SFHProf.C

7.141 SFHProf.h File Reference

7.141.1 Detailed Description

Declares class **SFHProf**.

Changelog:

- 9.6.04 BL: Added more documentation
- 21.7.04 BL: Introduced checkTwoIterators, checkOneIterator

Definition in file **SFHProf.h**.

`#include "jbltools/sfh/RegHProf.h"`

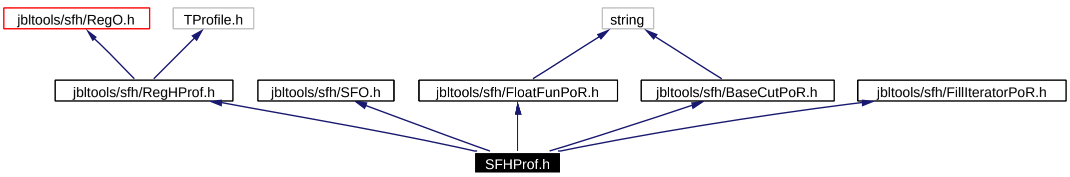
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

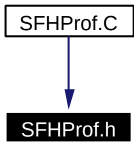
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFHProf**

  A self-filling 1-dimensional TProfile histogram.

7.142 SFMatrixOfH2F.C File Reference

7.142.1 Detailed Description

Implements class **SFMatrixOfH2F**.

Changelog:

- 14.6.05 BL: First version (from **SetOfH2F**)
- 23.8.05 BL: Use Float_FF for weight, x, and y in Fill()

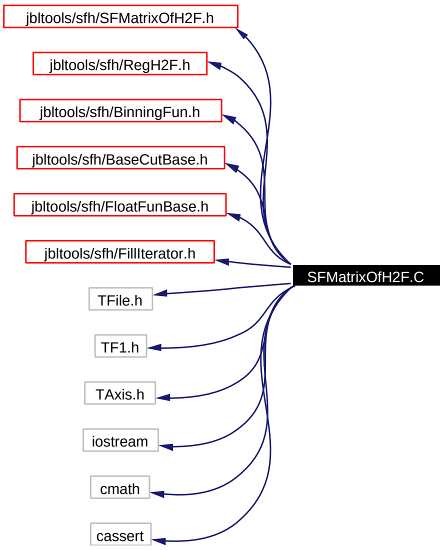
CVS Log messages:

- **Log**
  SFMatrixOfH2F.C,v
- Revision 1.3 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)
- Revision 1.2 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFMatrixOfH2F.C**.

```
#include "jbltools/sfh/SFMatrixOfH2F.h"
#include "jbltools/sfh/RegH2F.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/FloatFunBase.h"
#include "jbltools/sfh/FillIterator.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SFMatrixOfH2F.C:


jbltools/sfh/SFMatrixOfH2F.h
jbltools/sfh/RegH2F.h
jbltools/sfh/BinningFun.h
jbltools/sfh/BaseCutBase.h
jbltools/sfh/FloatFunBase.h
jbltools/sfh/FillIterator.h
SFMatrixOfH2F.C
TFile.h
TF1.h
TAxis.h
iostream
cmath
cassert

7.143 SFMatrixOfH2F.h File Reference

7.143.1 Detailed Description

Declares class **SFMatrixOfH2F**.

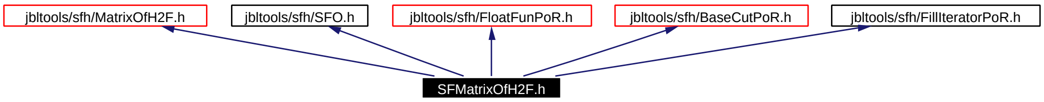
Changelog:

- 16.6.05 BL: First version (from **MatrixOfHProf.h**)

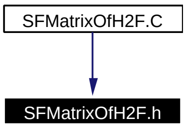
Definition in file **SFMatrixOfH2F.h**.

```
#include "jbltools/sfh/MatrixOfH2F.h"
#include "jbltools/sfh/SFO.h"
#include "jbltools/sfh/FloatFunPoR.h"
#include "jbltools/sfh/BaseCutPoR.h"
#include "jbltools/sfh/FillIteratorPoR.h"
```

Include dependency graph for SFMatrixOfH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFMatrixOfH2F**

  A 2-dimensional collection of 2-dimensional histograms that have the same axis.

7.144 SFMatrixOfHistograms.C File Reference

7.144.1 Detailed Description

Implements class **SFMatrixOfHistograms**.

Changelog:

- 11.6.05 BL: Bug-fix in Fill() method!
- Changed Fill() method to apply also to non-exclusive binnings
- 23.8.05 BL: Use Float_FF for weight and x in Fill()

CVS Log messages:

- **Log**
  SFMatrixOfHistograms.C,v
- Revision 1.18 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.17 2005/07/08 14:51:20 blist
- Use CVS Id in **SFMatrixOfHistograms**

- Revision 1.16 2005/07/08 14:49:14 blist
- Added identity string to **SFMatrixOfHistograms**,
- disabled warnings about unused variables in Makefile

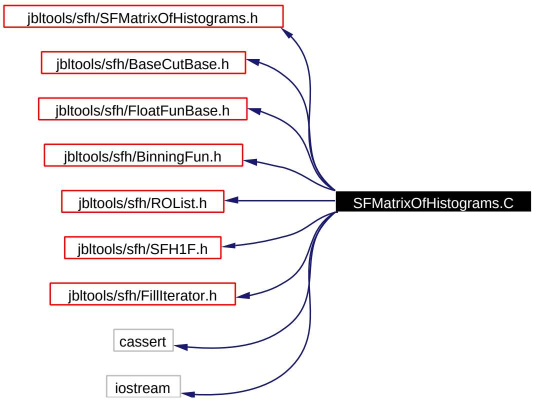
CVS Id:

- Id: /afs/desy.de/user/b/blist/.cvs/jbltools/sfh/SFMatrixOfHistograms.C,v 1.16 2005/07/08 14:49:14 blist Exp $

Definition in file **SFMatrixOfHistograms.C**.

```
#include "jbltools/sfh/SFMatrixOfHistograms.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/FloatFunBase.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/ROList.h"
#include "jbltools/sfh/SFH1F.h"
#include "jbltools/sfh/FillIterator.h"
#include <cassert>
#include <iostream>
```

Include dependency graph for SFMatrixOfHistograms.C:



7.145 SFMatrixOfHistograms.h File Reference

7.145.1 Detailed Description

Declares class **SFMatrixOfHistograms**.

Changelog:

- 30.6.04 BL: Add documentation
- 21.7.04 BL: Introduced checkTwoIterators, checkOneIterator

CVS Log messages:

- **Log**
  SFMatrixOfHistograms.h,v
- Revision 1.12 2005/07/08 14:49:14 blist
- Added identity string to **SFMatrixOfHistograms**,
- disabled warnings about unused variables in Makefile

Definition in file **SFMatrixOfHistograms.h**.

`#include "jbltools/sfh/MatrixOfHistograms.h"`

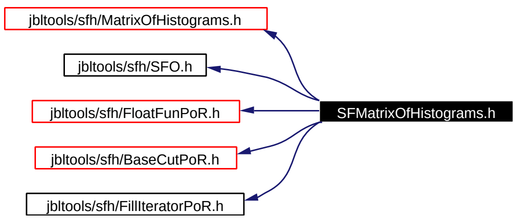
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

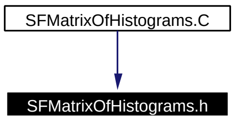
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFMatrixOfHistograms.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFMatrixOfHistograms**

A self-filling 2-dimensional collection of 1-D histos with the same axis.

7.146 SFMatrixOfHProf.C File Reference

7.146.1 Detailed Description

Implements class **SFMatrixOfHProf**.

Changelog:

- 7.6.05 BL: First version (from **SFMatrixOfHProf.C**)
- 14.6.05 BL: Bug-fix in Fill() method!
- Changed Fill() method to apply also to non-exclusive binnings
- 23.8.05 BL: Use Float_FF for weight, x, and y in Fill()

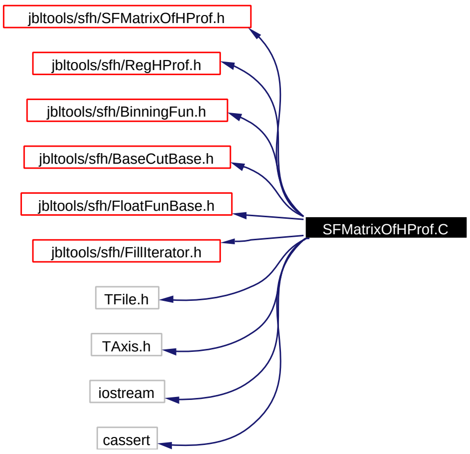
CVS Log messages:

- **Log**
  SFMatrixOfHProf.C,v
- Revision 1.5 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)
- Revision 1.4 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFMatrixOfHProf.C**.

```
#include "jbltools/sfh/SFMatrixOfHProf.h"
#include "jbltools/sfh/RegHProf.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/FloatFunBase.h"
#include "jbltools/sfh/FillIterator.h"
#include <TFile.h>
#include <TAxis.h>
#include <iostream>
#include <cassert>
```

Include dependency graph for SFMatrixOfHProf.C:


jbltools/sfh/SFMatrixOfHProf.h
jbltools/sfh/RegHProf.h
jbltools/sfh/BinningFun.h
jbltools/sfh/BaseCutBase.h
jbltools/sfh/FloatFunBase.h
jbltools/sfh/FillIterator.h
TFile.h
TAxis.h
iostream
cassert
SFMatrixOfHProf.C

7.147 SFMatrixOfHProf.h File Reference

7.147.1 Detailed Description

Declares class **SFMatrixOfHProf**.

Changelog:

- 7.6.05 BL: First version (from **SFMatrixOfHProf.h**)

Definition in file **SFMatrixOfHProf.h**.

`#include "jbltools/sfh/MatrixOfHProf.h"`

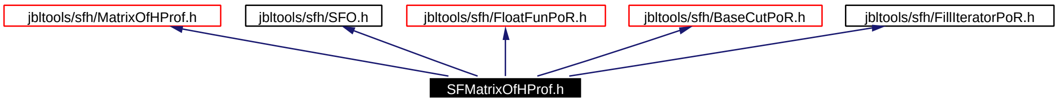
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

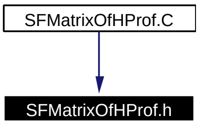
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFMatrixOfHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFMatrixOfHProf**

  A self-filling, 2-dimensional collection of profile histograms that have the same axis.

7.148 SFO.h File Reference

7.148.1 Detailed Description

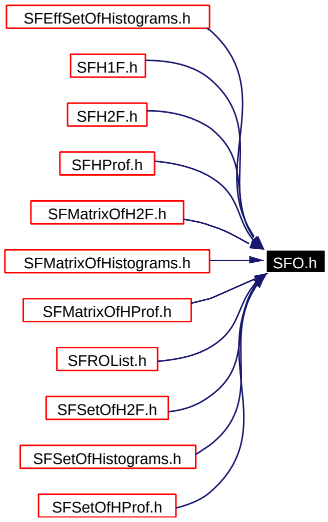
Declares class **SFO**.

Changelog:

- 10.6.04 BL: First version (adapted from SFH.h)

Definition in file **SFO.h**.

This graph shows which files directly or indirectly include this file:



Compounds

- class **SFO**

  Abstract mix-in base class for self-filling objects.

7.149 SFROList.C File Reference

7.149.1 Detailed Description

Implements class **SFROList**.

Changelog:

- 10.6.04 BL: First version, adapted from SFHList.C

CVS Log messages:

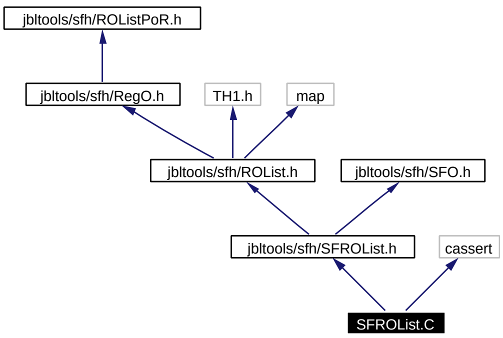
- **Log**
  SFROList.C,v
- Revision 1.6 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFROList.C**.

`#include "jbltools/sfh/SFROList.h"`

`#include <cassert>`

Include dependency graph for SFROList.C:



7.150 SFROList.h File Reference

7.150.1 Detailed Description

Declares class **SFROList**.

Changelog:

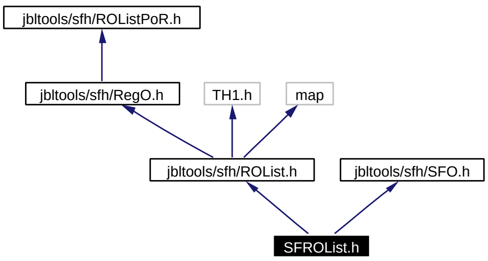
- 10.6.04 BL: First version, adapted from SFHList.h

Definition in file **SFROList.h**.

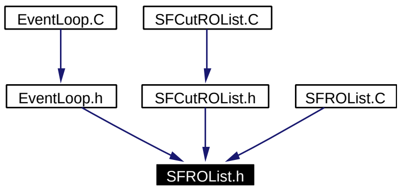
`#include "jbltools/sfh/ROList.h"`

`#include "jbltools/sfh/SFO.h"`

Include dependency graph for SFROList.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFROList**

  A list of histograms with a **Fill()** *method.*

7.151 SFSetOfH2F.C File Reference

7.151.1 Detailed Description

Implements class **SFSetOfH2F**.

Changelog:

- 10.6.05 BL: First version
- 11.6.05 BL: Changed Fill() method to apply also to non-exclusive binnings
- 23.8.05 BL: Use Float_FF for weight, x, and y in Fill()

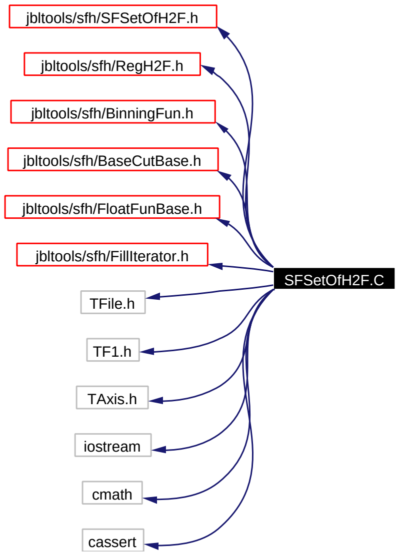
CVS Log messages:

- **Log**
  SFSetOfH2F.C,v
- Revision 1.4 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)
- Revision 1.3 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFSetOfH2F.C**.

```
#include "jbltools/sfh/SFSetOfH2F.h"
#include "jbltools/sfh/RegH2F.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/FloatFunBase.h"
#include "jbltools/sfh/FillIterator.h"
#include <TFile.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SFSetOfH2F.C:


jbltools/sfh/SFSetOfH2F.h
jbltools/sfh/RegH2F.h
jbltools/sfh/BinningFun.h
jbltools/sfh/BaseCutBase.h
jbltools/sfh/FloatFunBase.h
jbltools/sfh/FillIterator.h
SFSetOfH2F.C
TFile.h
TF1.h
TAxis.h
iostream
cmath
cassert

7.152 SFSetOfH2F.h File Reference

7.152.1 Detailed Description

Declares class **SFSetOfH2F**.

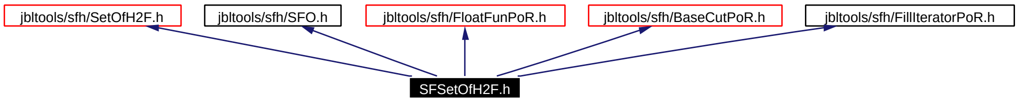
Changelog:

- 13.5.05 BL: First version

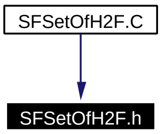
Definition in file **SFSetOfH2F.h**.

```
#include "jbltools/sfh/SetOfH2F.h"
#include "jbltools/sfh/SFO.h"
#include "jbltools/sfh/FloatFunPoR.h"
#include "jbltools/sfh/BaseCutPoR.h"
#include "jbltools/sfh/FillIteratorPoR.h"
```

Include dependency graph for SFSetOfH2F.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFSetOfH2F**

  A collection of profile histograms that have the same axis.

7.153 SFSetOfHistograms.C File Reference

7.153.1 Detailed Description

Implements class **SFSetOfHistograms**.

Changelog:

- 27.5.05 BL: Access private members of **SetOfH** through accessors
- 11.6.05 BL: Changed Fill() method to apply also to non-exclusive binnings
- 23.8.05 BL: Use Float_FF for weight and x in Fill()

CVS Log messages:

- **Log**
  SFSetOfHistograms.C,v
- Revision 1.17 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)

- Revision 1.16 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFSetOfHistograms.C**.

`#include "jbltools/sfh/SFSetOfHistograms.h"`

`#include "jbltools/sfh/BaseCutBase.h"`

`#include "jbltools/sfh/FloatFunBase.h"`

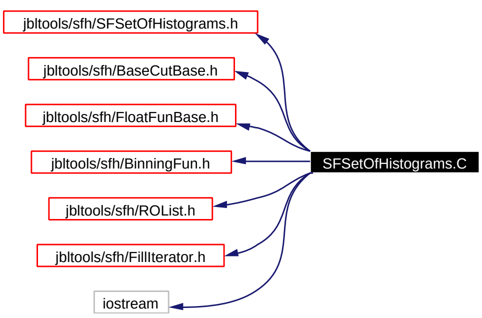
`#include "jbltools/sfh/BinningFun.h"`

`#include "jbltools/sfh/ROList.h"`

`#include "jbltools/sfh/FillIterator.h"`

`#include <iostream>`

Include dependency graph for SFSetOfHistograms.C:



7.154 SFSetOfHistograms.h File Reference

7.154.1 Detailed Description

Declares class **SFSetOfHistograms**.

Changelog:

- 22.6.04 BL: Added more documentation, use **ROListPoR**, **BinningFunPoR**, **FloatFunPoR**, **BaseCutPoR**, **FillIteratorPoR**, provide constructor with Binning object for axis binning

Definition in file **SFSetOfHistograms.h**.

`#include "jbltools/sfh/SetOfHistograms.h"`

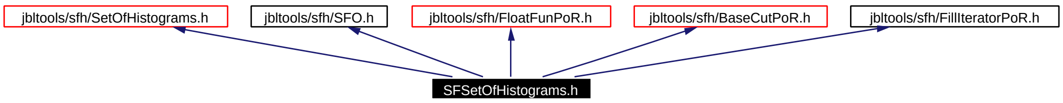
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

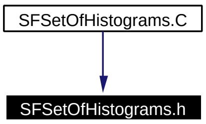
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFSetOfHistograms.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFSetOfHistograms**

  A self-filling collection of 1-dimensional histograms that have the same axis.

7.155 SFSetOfHProf.C File Reference

7.155.1 Detailed Description

Implements class **SFSetOfHProf**.

Changelog:

- 11.6.05 BL: Changed Fill() method to apply also to non-exclusive binnings
- 23.8.05 BL: Use Float_FF for weight, x, and y in Fill()

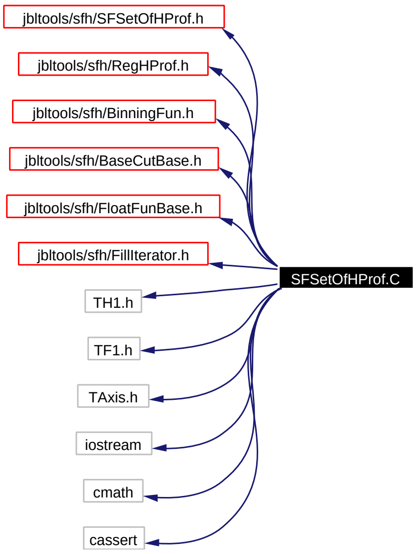
CVS Log messages:

- **Log**
  SFSetOfHProf.C,v
- Revision 1.6 2005/08/23 12:05:44 blist
- Data members of Binning are now private, bin edges are of type Axis_t
- New constructor for Binning from TAxis
- Self-Filling histograms etc. now use Float_FF instead of float (or double)
- Revision 1.5 2005/07/08 15:01:35 blist
- Added identity string to all .C files

Definition in file **SFSetOfHProf.C**.

```
#include "jbltools/sfh/SFSetOfHProf.h"
#include "jbltools/sfh/RegHProf.h"
#include "jbltools/sfh/BinningFun.h"
#include "jbltools/sfh/BaseCutBase.h"
#include "jbltools/sfh/FloatFunBase.h"
#include "jbltools/sfh/FillIterator.h"
#include <TH1.h>
#include <TF1.h>
#include <TAxis.h>
#include <iostream>
#include <cmath>
#include <cassert>
```

Include dependency graph for SFSetOfHProf.C:


jbltools/sfh/SFSetOfHProf.h
jbltools/sfh/RegHProf.h
jbltools/sfh/BinningFun.h
jbltools/sfh/BaseCutBase.h
jbltools/sfh/FloatFunBase.h
jbltools/sfh/FillIterator.h
SFSetOfHProf.C
TH1.h
TF1.h
TAxis.h
iostream
cmath
cassert

7.156 SFSetOfHProf.h File Reference

7.156.1 Detailed Description

Declares class **SFSetOfHProf**.

Changelog:

- 13.5.05 BL: First version

Definition in file **SFSetOfHProf.h**.

`#include "jbltools/sfh/SetOfHProf.h"`

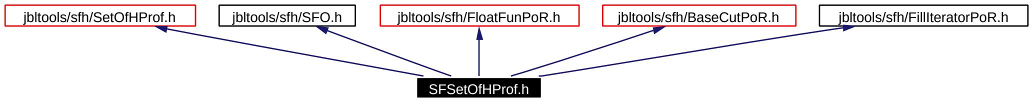
`#include "jbltools/sfh/SFO.h"`

`#include "jbltools/sfh/FloatFunPoR.h"`

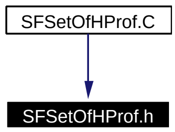
`#include "jbltools/sfh/BaseCutPoR.h"`

`#include "jbltools/sfh/FillIteratorPoR.h"`

Include dependency graph for SFSetOfHProf.h:



This graph shows which files directly or indirectly include this file:



Compounds

- class **SFSetOfHProf**

  A self-filling collection of profile histograms that have the same axis.

7.157 VectorIterator.h File Reference

7.157.1 Detailed Description

Declares class **VectorIterator**.

Author: Benno List, Jenny Böhme

Date

2006/09/12 09:24:48

Author

zimmerto

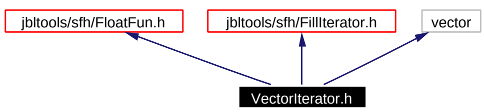
Changelog

- 19.8.04 BL, JB: Change template argument names T->Row, TM->Member, A->Arg

Definition in file **VectorIterator.h**.

```
#include "jbltools/sfh/FloatFun.h"
#include "jbltools/sfh/FillIterator.h"
#include <vector>
```

Include dependency graph for VectorIterator.h:



Compounds

- class **VectorFloatFun**

  Returns a data member of a vector's row in a **FloatFun**.

- class **VectorFloatFunFun**

  Returns the result of a member method of a vector's row in a **FloatFun**.

- class **VectorFloatFunFunArg**

  Returns the result of a member method of a vector's row that takes an argument in a **FloatFun**.

- class **VectorIntFun**

  Returns a data member of a vector's row in a **IntFun**.

- class **VectorIntFunFun**

  Returns the result of a member method of a vector's row in a **IntFun**.

- class **VectorIntFunFunArg**

  Returns the result of a member method of a vector's row that takes an argument in a **IntFun**.

- class **VectorBaseCut**

  Returns a data member of a vector's row in a **BaseCut**.

- class **VectorBaseCutFun**

  Returns the result of a member method of a vector's row in a baseCut.

- class **VectorBaseCutFunArg**

  Returns the result of a member method of a vector's row that takes an argument in a baseCut.

- class **VectorIterator**

  An iterator over a vector with row type Row.

Chapter 8

SFH Page Documentation

8.1 Basic Abstractions used in SFH

If one looks at the `graphical class hierarchy` of this package, one notes a number of base classes that form the roots of the class hierarchy. These base abstractions have generally very simple interfaces, and will be discussed in the following.

8.1.1 Class SFO: Self-Filling Object

Class **SFO** is the heart of the self-filling histograms. Any class derived from **SFO** must implement a method Fill(), by which the corresponding object is filled, without the need to pass abscissa values or weights to the object.

The implementation of this Fill() method will generally use objects of classes **BaseCut**, **FloatFun** and possibly **FillIterator** and **BinningFun** to know what the abscissa value and weights are, and into which histogram to fill a value.

Particular subclasses of **SFO** are **SFH1F**, **SFH2F**, and **SFHProf**, which correspond to the RooT classes TH1F, TH2F, and THProf (and are actually derived from them).

Additionally, also collections of histograms can be self-filling, and are therefore derived from **SFO**. One such collection id **SFROList**, which is simply a collection of self-filling Objects. Its Fill() method is implemented by looping over all objects in its collection and calling their respective Fill() method.

By making **SFROList** an **SFO** itself, we allow the creation of trees of such object lists. This is an application of the "Composite" pattern of Gamma, Helm, Johnson and Vlissides (the famous "Gang of Four", "GoF" in short).

A particular **SFROList** subclass is **EventLoop**, which is the base class for the classes where the user books and outputs her histohrams.

Another type of self-filling histogram collections are **SFSetOfHistograms** and **SFMatrixOfHistograms**. The collections also have a Fill() method and administer several histograms. However, as opposed to **SFROList**, the Fill() method will fill only one histogram at a time. Objects of type **BinningFun** are checked into which histogram one should make an entry, and then the corresponding histogram gets an entry. These collections are used if we have a number of similar histograms (say, mass histograms) which should be filled under mutually exclusive conditions. These conditions can be file type (histogram 0 for data, 1 for MC type 1, 2 for MC type 2, etc), or some kinematic quantity, if we want to make a measurement differential in t, W, pt, or some other variable.

8.1.2 Class RegO: Registered Object

A registered object is an object that registers itself in a list (of class **ROList**) when it is created. Consequently, a **RegO** object can only be created when it is given a pointer to a **ROList**. (However, the pointer may be NULL.)

Initially, registered objects where only histograms in our framework (at which time we called the class RegH, for "registered histogram"). The main purpose was to build lists, or collections, of self-filling histograms.

Meanwhile we have extended the use of **RegO**. Now, also cached Objects (class **CachedO**) are derived from **RegO**, and we use **ROList** objects of cached objects to invalidate all caches of cached objects.

In future releases, we will probably use **RegO** to implement a reference-counting mechanism, so that purists will not have to complain about memory leaks anymore.

8.1.3 Class ROList: Registered Object List

As already discussed, **RegO** objects register themselves in **ROList** objects.

Class **ROList** is again derived from **RegO**, another case of the "Composite" pattern of GoF.

8.1.4 Class FloatFun: float-valued Function objects

Our framework makes heavy use of function objects.

A function object is symply an object that overloads operator(). One can think of it a a class that has mainly one purpose, and therefore one main method. This method could be called "doit()", or "foo()", or "doWhatThisClassIsMeantToDo()", or simply have "no name", namely be "operator()".

The nice (though at first puzzling) thing with function objects is that if you have declared such an object, lets say

```
  FloatFun& pt = *new PtFun (ntuple);
```

you can write

```
  float thePT = pt();
```

which looks like a function call (in fact, it is a function call, it calls PtFun::operator()!) and gives the impression that pt itself is simply a function, declared as

```
float pt() {
  return something;
}
```

rather that the function object that it really is.

It is proably best to think of a function object as a function with a memory. The object's state is the "memory", it can hold values (e.g. the normalization, mean and sigma of a Gaussian, or a pointer to an ntuple variable), the operator() is the main way to use the object.

Now, a **FloatFun** in our framework is a function object where operator() takes no arguments and returns a float, leaving the object itself unaltered:

```
class FloatFun {
  public:
    virtual float operator() () const = 0;
    virtual void destroy() { delete this; }
    virtual const FillIterator *getIterator() const { return 0; }
  protected:
    virtual ~FloatFun() {};
};
```

The idea is that for any variable that we want to plot, we have a function object of some class derived from **FloatFun** that will return the value of the variable via operator().

The self-filling objects will then get pointers to **FloatFun** objects when they are created, and thereby "know" what to plot when their Fill() method is called.

A nice percolate of function objects is that we can easily define classes that hold pointers to other function objects and thereby can implement arithmetic operations like product or sum, or represent functions of function objects like sine or square root.

In our framework, **FloatFun** objects are used to define abscissa values and weights for self-filling histograms.

We can also express cuts by writing things like

```
FloatFun& pt = *new PtFun(ntuple);     // returns pt of an event
BaseCut& ptcut1 = (pt > 3);            // A cut on pt
BaseCut& ptcut2 = (2 <= pt < 3);       // Another ptcut, equivalent to
                                       // (2 <= pt) && (pt < 3);
SFH1F *h = new SFH1F ("pt",            // Histo name
                      "Event pt > 3",  // Histo title
                      ptbinning,       // The binning
                      pt,              // What is plotted: pt
                      pt > 3,          // A cut
                      sqrt (pt));      // A weight
```

One last note: **Why is the destructor of FloatFun virtual?** Actually, for **FloatFun** itself this is a bit pointless, because **FloatFun** is a pure abstract class, so no **FloatFun** objects can be instantiated anyway. However, it should remind you, the user, also to make the destructor of any subclass of **FloatFun** protected, or even private (in which case no further subclasses can be derived from that class!).

The protected constructor makes code such as this illegal:

```
// constructor of DataLoop (derived from EventLoop):
// Book all histograms
DataLoop::DataLoop (Ntuple& ntuple) {
  PtFun pt(ntuple);               // pt is now an instance of PtFun
  // h shall plot the event pt; it will hold a pointer to the
  // pt object!
  h = new SFH1F ("pt", "Event pt", ptbinning, pt); /

  // pt goes out of scope, ~PtFun(0 is called,
  // => object pt does no longer exist, the pointer held by
  // the self-filling hstogram h is now invalid!
}
```

If PtFun has a protected destructor, this code will not compile.

Since **FloatFun** objects are referenced through pointers held by self-filling objects like **SFH1F** objects, and since the pointers are used in the **SFH1F::Fill()** method, the **FloatFun** objects must live at least until the last Fill() call has been made, i.e. basically until the end of the program. By forcing the user to write

```
// constructor of DataLoop (derived from EventLoop):
// Book all histograms
DataLoop::DataLoop (Ntuple& ntuple) {
  PtFun& pt = *new PtFun(ntuple);  // pt now points to an instance of PtFun
  // h shall plot the event pt; it will hold a pointer to the
  // pt object!
  h = new SFH1F ("pt", "Event pt", ptbinning, pt); /

  // pt goes out of scope, but the (anonymous) objects
  // it points to lives on! => All is well that ends well...
}
```

we can ensure that the **FloatFun** objects live as long as necessary. See section **Why Do I Have to Create FloatFun, BaseCut, andFillIterator** Objects with 'new'?" for further discussion on this subject.

8.1.5 Class BaseCut: Base Class for Cuts

If class **FloatFun** returns a float, then Class **BaseCut** could also have been named BoolFun, which would maybe have been more logical. Anyway, an object of some subtype of **BaseCut** has to return a bool when its operator() is called:

```
class BaseCut {
  public:
    virtual bool operator() () const = 0;
    virtual void destroy() { delete this; }
    virtual const FillIterator *getIterator() const { return 0; }
  protected:
    virtual ~BaseCut() {};
};
```

So, a **BaseCut** object looks at an event (or some part of an event, such as a jet, if a **FillIterator** is used), decides whetehr the event passes a cut or not, and returns this decision as result of its operator().

8.1.6 Class IntFun: Integer valued Function objects

Class **IntFun** is equivalent to **FloatFun** in that it returns an integer when operator() is called:

```
class IntFun {
  public:
    virtual int operator() () const = 0;
    inline virtual FloatIntFun& Float () const {
      return *new FloatIntFun(*this);
    }
    virtual void destroy() { delete this; }
    virtual const FillIterator *getIterator() const { return 0; }
  protected:
    virtual ~IntFun() {};
};
```

An **IntFun** is very useful to express cuts on integer values, such as the number of tracks:

```
  IntFun&  ntrack = NTIntFun<Ntuple> (ntuple, "ntracks");
  BaseCut& trackcut = (ntrack == 2);
```

If, however, we want to plot an integer value, like the number of tracks in an event, we need a **FloatFun**, not an **IntFun**.

For this, we have method **Float()**, which returns an object of type FloatIntFun, which is derived from **FloatFun** and stores a reference to an **IntFun**. It calls operator() of the **IntFun**, converts the result to a float and returns it as result of its own operator():

```
class FloatIntFun: public FloatFun {
  public:
    FloatIntFun (const IntFun& intFun_)
      : intFun (intFun_) {}
    virtual float operator() () const {
      return intFun();
    }
  protected:
    virtual ~FloatIntFun() {};
    const IntFun& intFun;
};
```

8.1.7 Class FillIterator: Filling Iterators

The **FillIterator** idea came relatively late into our framework, which is why **FillIterator** objects sometimes appear at strange places in argument lists. Anyway, what is a **FillIterator** for?

A **FillIterator** helps to adress the question "how can we plot more that one value per event (= ntuple row)?". Typical applications occur when we want to plot the pt (transverse momentum) of all tracks in an event, or the distance of all secondary vertices, or the masses of all D∗ candidates.

The interface of this class is a bit more complicated than the interfaces we saw so far:

```
class FillIterator: public IntFun {
  public:
    virtual int operator() () const=0;
    virtual bool next()=0;
    virtual bool reset()=0;
    virtual const FillIterator *getIterator() const { return this; }
  protected:
    virtual ~FillIterator() {};
};
```

Typically, a **FillIterator** will access some integer value in an ntuple that contains the number of tracks, jets, D∗ candidates, or similar, and have an internal variable (like "index") that contains the value of the index.

To facilitate the writing of such a class, we have defined a subclass SimpleFillIterator, where the user just has to provide (in a subclass, of course) an implementation of method getRange(), and the rest is already in place.

A **FillIterator** object (if there is one) is used by the Fill() method of a self-filling histogram to step through all objects (jets, tracks, D∗ candidates) that are to be plotted. Then, the **FloatFun** objects are called and asked about their values.

Consequently, such **FloatFun** objects have to know about the iterator, i.e. they have to hold a pointer to the iterator and in their operator() method have to ask the iterator about the number of the track, jet or whatever whose pt or other value should be plotted.

Such a **FloatFun** might look like this:

```
class PtFun : public FloatFun {
  public:
```

```
    PtFun(const Ntuple& nt_, const FillIterator& iter_)
      : nt(nt_), iter(iter_)
      {};
    virtual float operator() () const {
      assert (trackIter() >= 0);
      return nt->IdPt[trackIter()];
    };
  protected:
    ~PtFun() {};
    const Ntuple& nt;
    const FillIterator* iter;
};
```

Common to all **FloatFun** objects that collaborate with a **FillIterator** is that they have to store a pointer (or reference) to a **FillIterator** object, because this object has to tell them the object number whose properties they should return.

It is of paramount importance that the **FloatFun** object gets the same **FillIterator** object that is also passed to the self-filling histogram, because the histogram's Fill() method is responsible for incrementing the **FillIterator** object, which should be reflected in the result of the **FloatFun**.

8.1.7.1 Relation between FillIterator and IntFun

As can be seen from the definition of **FillIterator**, a **FillIterator** is also an **IntFun**. This is resonable, because an **IntFun** asically has only to provide an operator() that returns an int, which is what a **FillIterator** does.

A **FillIterator** has sort of two faces:

- One face it presents to other **FloatFun**, **IntFun**, or **BaseCut** objects that normally will only use operator(), i.e. ask the **FillIterator** about its index value. For those clients the **FillIterator** is actually just as good as any other **IntFun**.
- The other face it presents to self-filling objects, which normally don't care about operator(), but use reset() and next() to iterate through the different values.

An advantage of a **FillIterator** being an **IntFun** is that we can use **FillIterator** also as value for an axis, e.g. in a situation where we want to plot how often a subtrigger has fired.

8.1.7.2 The Purpose of Method getIterator()

We have so far not discussed the purpose of method getIterator(), which is a method defined for classes **FloatFun**, **IntFun**, and **BaseCut**.

If any object of these classes uses a **FillIterator** object, it should return the pointer to this object in method getIterator(). This is used by the self-filling histograms to ensure that all function objects depend on the same iterator, and furthermore allows the self-filling histograms to deduce which iterator to use, so that the **FillIterator** needs not to be given explicitly.

8.1.8 Class CachedO: Cached Objects

Sometimes the calculation of event properties can be rather time-consuming, for instance when a jet finder has to be run over a set of tracks and clusters before jet properties like pt are plotted.

In such cases it can be prohibitively slow to recompute these properties every time operator() of a **FloatFun** is called. This problem is adressed by cached objects.

The interface of the abstract base class **CachedO** is again very simple:

```
class CachedO: public RegO {
  public:
    CachedO (const ROListPoR& rol);
    virtual ~CachedO ();
    virtual void invalidateCache() = 0;
  private:
    CachedO (const CachedO& rhs);
    CachedO& operator= (const CachedO& rhs);
};
```

The main property of a **CachedO** is a method invalidateCache(), which tells the object that its cache has become invalid, which will be the case after a new row of an ntuple has been read.

Tobe able to efficiently invalidate all caches, we made **CachedO** a registered object (derived from **RegO**). In class **EventLoop**, we have a **ROList** named cachedObjects, which is used to collect all cached objects and call invalidateCache() for all of them in the loop() method.

To facilitate the use of cached objects, we have defined four derived classes that should serve as base classes for cached **FloatFun** and **BaseCut** classes. We have defined two versions, one using iterators and one without iterators:

- SimpleCachedFloatFun: A **FloatFun** whose value is cached
- IteratorCachedFloatFun: A **FloatFun** with a **FillIterator** whose values are cached
- SimpleCachedIntFun: A **IntFun** whose value is cached
- IteratorCachedIntFun: A **IntFun** with a **FillIterator** whose values are cached
- SimpleCachedCut: A **BaseCut** whose value is cached
- IteratorCachedCut: A **BaseCut** with a **FillIterator** whose values are cached

8.1.9 A Word on const and mutable

An important caveat in using **CachedO** subclasses concerns the use of const. The classes **FloatFun** and **BaseCut** have declared their respecive operator() methods to be "const", i.e. these methods must not change any data members of these classe.

Howeve, if we want to recalculate the value of a **FloatFun** object only when operator() is actually called, we face the problem that we need to store (i.e., cache) the result in a data member, which is not allowed in a "const" method. Now what?

C++ defines the notion of "logical const-ness", i.e. allows to have data members that can be changed even for const objects. This is signalled by the keyword "mutable". Look at our implementation of SimpleCachedFloatFun:

```
class SimpleCachedFloatFun: public FloatFun, public CachedO {
  public:
    SimpleCachedFloatFun (const ROListPoR& rol)
      : CachedO (rol), cacheValid (false), cachedValue (0)
      {}

    virtual float operator() () const {
      if (!cacheValid) {
        recalculate();
        cacheValid = true;
      }
```

```
      return cachedValue;
    }

    virtual void invalidateCache() {
      cacheValid = false;
    }

    virtual void recalculate() const = 0;

  protected:
    mutable bool cacheValid;
    mutable float cachedValue;
};
```

Here, cacheValid and cachedValue are declared "mutable" and hence may be altered for const objects, i.e. in member functions that are marked "const", such as operator(). Observe also that recalculate() has been declared "const", because otherwise it could not be called by operator().

Of course, another way out would be to recalculate the values immediately in the invalidateCache() method, which is not "const":

```
class SimpleCachedFloatFun: public FloatFun, public CachedO {
  public:
    SimpleCachedFloatFun (const ROListPoR& rol)
      : CachedO (rol), cachedValue (0)
      {}
    virtual float operator() () const {
      return cachedValue;
    }
    virtual void invalidateCache() = 0;
  protected:
    float cachedValue;
};
```

We have decided against this model, because presumably the recalculation is a costly process (otherwise we wouldn't go through all the trouble), and it is quite possible that for many events the recalculation is unnecessary, because cuts are made that reject most events (consider, for example, an application where we run a jet finder on an LHC event, but only for events which have at least two muon candidates). In such a case, a calculation of the **FloatFun** result for every event could degrade performance more than not using the cache mechanism at all.

8.1.10 How to apply cached objects

We see basically two situations where cached objects are beneficial:

- We have relatively simple, straighforward calculations like making complicated cuts that access a lot of n-tuple variables, or cases where simple but slow calculations are made, such as taking an arc tangent of a quantity. In these cases we recommend to derive a user class from SimpleCachedFloatFun, IteratorCachedFloatFun, SimpleCachedCut, or Iterator-CachedCut,whatever applies, and implement the recalculate method.
- We have long, involved calculations like running jet finders or vertex finders, and want to use the result in several ways (e.g., we want to plot pt, rapidity, and mass of jets). In that case we recommend to derive a class from **CachedO** that does all the calculations (say, the jet finder), and then write additional **FloatFun** or **BaseCut** objects that access the data of the cached class.

8.1.11 HVisitor: Histogram Visitor

In handling collections of histograms we often encounter the situation that we want to apply a certain operation to all histograms in the collection. Such operations can be writing histograms to a file, drawing histograms on a canvas, fitting them with some fit function, or setting attributes like line colors.

Now we have two possibilities:

We could write (ugly) code like this:

```
void setLineColors (const ROList& rol) {
  for (unsigned int i = 0; i < rol.getEntries(); i++) {
    RegO *ro = rol.getEntry(i);
    // Make sure our registered object is really a RooT histogram:
    if (TH1 *h = dynamic_cast<TH1 *>(ro)) {
      // Set line color to red
      h->->SetLineColor (2);
    }
  }
}
```

Then we apply this method to the various collections (e.g., all **SFSetOfHistograms**, all **SFMatrixOfHistograms** objects).

Somewhat better would be to derive our own collections:

```
class MySetOfHistograms: public SetOfHistograms {
  void setLineColors () {
    for (unsigned int i = 0; i < this->getEntries(); i++) {
      RegO *ro = this->getEntry(i);
      // Make sure our registered object is really a RooT histogram:
      if (TH1 *h = dynamic_cast<TH1 *>(ro)) {
        // Set line color to red
        h->->SetLineColor (2);
      }
    }
  }
};
```

But then we have to provide a lot of constructors, we have to repeat the exercise for SSetOf-Histograms, **MatrixOfHistograms**, **SFMatrixOfHistograms**, and possibly other classes. Not nice.

Or is there a third possibility? The Visitor pattern from GoF comes to the rescue:

Write a class like this:

```
class LineColorSetter: public HVisitor {
  public:
    virtual void visit (RegO& ro) {
      if (Th1 *h = dynamic_cast<TAttLine *>(&ro)) {
        h->SetLineColor (h);
      }
    }
};
```

Now, all we need is some code in class **ROList** that calls the visit() method for every object in its list:

```
ROList& ROList::visit (HVisitor& v) {
```

```
  for (unsigned int i = 0; i < entries; i++) {
    if (theList[i]) v.visit((*theList[i]));
  }
  return *this;
}
```

(the true code is a bit more involved, but that's not the point here).

Now in our user code we can write code like this:

```
  LineColorSetter lcs; // a LineColorSetter object
  // sfset is a SFSetOfHistograms
  sfset.visit(lcs);    // Set all histogram's line colors to red
```

In file **HVisitors.h** (plural!), we have defined a number of handy **HVisitor** subclasses that perform common tasks:

- **Writer**: writes out all TObject objects into a (previously opened) RooT file.
- **Normalizer**: Normalizes the contents of all histograms to a constant value (nice for shape comparisons).
- **BinWidthNormalizer**: For histograms with a non-equidistant binning, the bin contents are divided by the bin with, so that one can make the transition from a histogram that contains the number of events per bin to a differential cross-section.
- **ErrorEraser**: Sets the errors of the bins in all histograms to zero.
- **MinMaxSetter**: Sets the minimum and maximum of all histograms to some values.
- **AttMarkerSetter**: Sets the marker attributes color, size, and style of all histograms.
- **AttLineSetter**: Sets the line attributes color, style, and width of all histograms.
- **AttFillSetter**: Sets the fill attributes color and style of all histograms.
- AxisAttribSetter: Sets the axis attributes size, name, color, font, and number of divisions of all histograms.
- **AxisSetter**: Sets the x axis of all histograms to a given binning.

We have also predefined some objects that can be used directly to set some attributes, with (as we think) selfexplaining names:

```
static AttLineSetter blackline (1);
static AttLineSetter redline (2);
static AttLineSetter greenline (3);
static AttLineSetter blueline (4);
static AttLineSetter yellowline (5);
static AttLineSetter magentaline (6);
static AttLineSetter cyanline (7);
static AttFillSetter blackfill (1);
static AttFillSetter redfill (2);
static AttFillSetter greenfill (3);
static AttFillSetter bluefill (4);
static AttFillSetter yellowfill (5);
static AttFillSetter magentafill (6);
static AttFillSetter cyanfill (7);
static AttFillSetter hollowfill (-1, 0);
static AttFillSetter solidfill (-1, 1001);
static AttMarkerSetter blackmarker (1);
```

```
static AttMarkerSetter redmarker (2);
static AttMarkerSetter greenmarker (3);
static AttMarkerSetter bluemarker (4);
static AttMarkerSetter yellowmarker (5);
static AttMarkerSetter magentamarker (6);
static AttMarkerSetter cyanmarker (7);
```

A note on the class name: **HVisitor** should really be called RegOVisitor, but for historical reasons its name is what it is.

8.1.11.1 How to use HVisitor and ROList efficiently

As we have seen, **HVisitor** objects are an efficient way to perform operations on collections of histograms.

Often we'll want to perform some operation, let's say a mass fit, only on some subset of all histograms. In such cases it makes sense to define our own **ROList** objects in our DataLoop class:

```
class DataLoop: public EventLoop {
public:
  // the usual stuff here...
private:
  ROList masshistos;
  ROList otherhistos;
};

// constructor: book histos
DataLoop::DataLoop() {
  // Define binnings, FloatFuns etc here

  // book a mass histo, put it into list masshistos:
  new SFH1F ("mass", "Some Mass", massbinning,
             masshistos, massfun);
}

// Plot histograms
DataLoop::output (const char* rootfile, const char* psfile) {
  // MassFitter is a HVistor that fits a mass histogram
  MassFitter theMassFitter;
  // Fit only mass histograms, not the others
  masshistos.visit (theMassFitter);
  // Continue here with plotting etc...
}
```

Another possibility would be that a **HVisitor** checks e.g. the name of a histogram and fits only histograms that contain the string "mass".

8.1.12 Binning: Histogram Binning

Class Binning is our way to have objects that represent the binning of a histogram axis. We think that RooT itself should have such a class.

Often we have to book histograms which all have the same binning. Now, instead of

```
  h1 = new TH1F ("h1", "hist 1", 100, 0., 1.);
  h2 = new TH1F ("h2", "hist 2", 100, 0., 1.);
  h3 = new TH1F ("h3", "hist 3", 100, 0., 1.);
  h4 = new TH1F ("h4", "hist 4", 100, 0., 1.);
  // and so on, ad infinitum...
```

we rather want to be able to write

```
Binning hbinning (100, 0., 1.);
h1 = new RegH1F ("h1", "hist 1", hbinning, this);
h2 = new RegH1F ("h2", "hist 2", hbinning, this);
h3 = new RegH1F ("h3", "hist 3", hbinning, this);
h4 = new RegH1F ("h4", "hist 4", hbinning, this);
// and so on, ad infinitum...
```

which allows us to change the binning of all histograms in one single place.

Binning objects allow us to do exactly that. We have constructors for Binning objects that exactly mimic the corresponding part in the RooT histogram constructors:

```
class Binning {
  public:
    Binning ();
    Binning (int nbins_, double xlow, double xhigh);
    Binning (int nbins_, const float binedges_[]);
    Binning (int nbins_, const double binedges_[]);
    Binning (const Binning& rhs);
    virtual ~Binning();

    virtual int getBin (double x) const;
    virtual int getNBins() const;
    virtual double getLowerBinEdge(int i) const;
    virtual double getUpperBinEdge(int i) const;
    virtual double getLowerEdge() const;
    virtual double getUpperEdge() const;
    virtual const double *getEdges() const;
    virtual bool isEquidistant() const;

  protected:
    int nbins;
    double *binedges;
    bool equidistant;

  private:
    Binning& operator= (const Binning&);
};
```

We have defined constructors for all our histogram classes that take Binning objects instead of the parameters nbins, xlow, xhigh.

8.1.13 BinningFun: Histogram Binning Function objects

Derived from Binning is class **BinningFun**.

Compared to a Binning, it has the same constructors, and only a few additional methods that have to be implemented by a derived class: class **BinningFun**: public Binning, public **IntFun** { public: **BinningFun** (); **BinningFun** (int nbins_, float xlow, float xhigh); **BinningFun** (int nbins_, const float binedges_[]); **BinningFun** (int nbins_, const double binedges_[]); **BinningFun** (const Binning& binning_);

virtual int operator() () const = 0; virtual const char ∗getBinName(int i) const = 0; virtual const char ∗getBinTitle(int i) const = 0; protected: virtual ∼BinningFun() {}; };

A **BinningFun** returns a bin number (-1 for "no bin") in operator(), insofar it is also an **IntFun**. **BinningFun** objects are used by classes **SFSetOfHistograms** and **SFMatrixOfHistograms** to decide into which histogram a certain entry should be made.

For instance, if we have a flag "filetype" in our Ntuple, where 0 means "data", 1 means "Signal Monte Carlo", and 2 means "Background Monte Carlo", we could have a **BinningFun** that returns 0, 1, or 2, and then the entry goes into the corresponding histogram in a **SFSetOfHistograms**.

In addition to operator(), a **BinningFun** has two other methods that have to be implemented by a derived class, namely getBinName(int i) and getBinTitle (int i). These methods should return strings (allocated with operator new[]) with a name or a title for a given bin i. The name should be short (like "0", "1", "2" or "data", "sigMC") and contain no spaces, the title could be nicer like "Data", "Signal Monte Carlo" or "0.1 < t < 0.2". These methods are used by **SetOfHistograms** and **MatrixOfHistograms** and their subclasses to generate histogram names and titles during the histogram booking stage.

To make life easier, we have defined a class **FloatFunBinning**, which takes a **FloatFun** that defines a variable according to which the binning is done, a Binning, and a string that should be the variable name (like "t" or "pt" or "W"), from which bin names like "007" and bin titles like "0.1 < t < 0.2" are generated.

8.2 Frequently Asked Questions

8.2.1 What are these strange FloatFunPoR, BaseCutPoR, and FillIteratorPoR classes?

Self-filling histograms take pointers to **FloatFun**, **BaseCut**, and **FillIterator** objects, because generally it makes sense to pass no such objects: If no **BaseCut** is passed, no cut is applied, and if no **FillIterator** is passed, we have exactly one entry per ntuple row.

On the other hand, to define logical or arithmetic operators between these objects, we have to work with references: We cannot define

```
inline AndOfTwoCuts& operator&& (const BaseCut *lhs_,
                                 const BaseCut *rhs_) // does not work!
{
  return *new AndOfTwoCuts (*lhs_, *rhs_);
}
```

because we cannot overload an operator that takes only pointers.

So, if we want not to have to write

```
// If SFH1F takes only pointers
SFH1F *h1 =
  new SFH1F ("JPsiElasticTT",
             "#muODS: m_{J/#Psi}: elastic production, (#mu-#mu)",
             40, 2., 4.,
             *this,
             jpsiMass,
             &(*jpsiElastic&&*jpsiTrackTrack&&*jpsiNoCosmic&&*jpsi2mu));
```

we would have to use references everywhere, but then we cannot pass NULL pointers anymore, at least not easily (a possible way out would be to use an ugly, dirty, mean trick like `*(static_cast<BaseCut *>(0))`).

The logical way out would be to provide constructors with all posible combinations of pointers and references, which would mean increasing the number of constructors by something like a factor 16.

A better possibility is to introduce a class **FloatFunPoR** (which stands for "FloatFun Pointer Or Reference"). This class has two constructors, taking a pointer or a reference to a **FloatFun**, and store the pointer (in member **FloatFunPoR**:pff, meaning "Pointer to FitFun"). These constructors constitute automatic conversion operators, so that if we now call the **SFH1F** constructor with either a pointer or a reference to a **FloatFun** object (or to a subclass of it), this is automatically converted to an **FloatFunPoR** object, which is then used by the **SFH1F** constructor to extract the pointer to a **FloatFun**.

Clever, isn't it?

(We haven't seen this trick anywhere else. Maybe we should have it patented? If amazon can get 1-click shopping patented...)

8.2.2 We Cannot Delete FloatFun, BaseCut, and FillIterator Objects. Isn't that a Memory Leak?

Short answer: Yes. Why do you care? How many of these objects do you plan to create?

Long answer: Needs to be written...

8.2.3 Why Do I Have to Create FloatFun, BaseCut, and FillIterator Objects with "new"?

To most people, it would be much more attrctive to write

```
IDTrackIterator trackIter (ntuple);
IpzFun          ipzfun (ntuple);
PtFun           ptfun (ntuple, &trackIter);
```

instead of

```
IDTrackIterator *trackIter   = new IDTrackIterator (ntuple);
IpzFun          *ipzfun      = new IpzFun (ntuple);
PtFun           *ptfunPtFun  = new PtFun (ntuple, trackIter);
```

or (our recommendation):

```
IDTrackIterator& trackIter   = *new IDTrackIterator (ntuple);
IpzFun&          ipzfun      = *new IpzFun (ntuple);
PtFun&           ptfunPtFun  = *new PtFun (ntuple, &trackIter);
```

The problem with the first solution, using automatic variables, is this: Typically we (seem to) need all these objects only within one function, e.g. in the constructor of an **EventLoop**, where we book all histograms.

Automatic variables would be destroyed at the end of this function. However, the self-filling histograms store only pointers to these objects instead of copying them, so these pointers would become invalid.

But the objects are needed later to fill the histograms, so they must not be deleted too early. Allowing the use of automatic variables puts this burden on the shoulder of the user, who would have to make sure the objects live long enough, e.g. by making them data members of her **EventLoop** subclass.

This is burdensome and error-prone.

If, however, the objects are created with "new", they live until they are explicitly destroyed, which normally happens only when the program terminates.

Another solution would be that the self-filling histograms keep a copy of the objects. This needs an additional "clone()" method, because we need a sort of polymorphic copying. The resulting classes would look like this:

```
class IpzFun : public FloatFun {
  public:
    IpzFun(Ntuple* ntuple) : nt(ntuple) {};
    virtual float operator() () const {return nt->ipz;}
    virtual IpzFun& clone() () const {return new IpzFun(*this);}
  protected:
    Ntuple* nt;
};
```

The clone() method calls the automatically generated copy constructor, which in most cases is fine.

The disadvantage of this method is threefold:

- We generate a lot more objects.

- The user has to write the clone() method for each subclass. The method cannot be inherited, because it needs to use the specific copy constructor of the class at hand. This makes it more tiresome to write a new **FloatFun** subclass.
- Since now all histograms that plot one quantity, say, the J/psi mass, hold a copy of the original object, operator() is called only once per event and **FloatFun** object. This makes any sort of caching mechanism impossible. If we have only one jpsimass object, we could (if necessary for perfomance reasons) implement a mechanism that calculates the J/psi mass once every time a new ntuple row is loaded, and then always returns the cached value.

Our first design actually used the clone approach, but these disadvantages convinced us to go the other way.

8.2.4 Writing all the Function Object Classes is Tiresome. What's the Advantage?

Object Oriented Programming is not primarily about writing less code, although we think that as soon as you use the visitors and sets or matrices of histograms, you'll find you will write less code.

In our opinion, OOP is about maintainability of code. We think that our toolbox allows you to write code that is better structured, has smaller programming units, and is therefore better to understand and modify.

When you start an analysis, you rarely know where you'll end up. The code tends to grow with time, gets more and more complicated as more and more cuts and control plots are added. This leads to software entropy, which you'll often not notice until it's too late (typically 2 weeks before you have to present your result at a conference, when you are asked to change the order of your cuts or introduce a new cut right at the beginning of your cut chain).

Reordering cuts is a real pain if you nest cuts in if-then-else structures. We find that this is much easier in our framework.

Once you start to bin your J/psi events in 10 t bins times 5 W bins, and want to plots the mass peaks for 5 different cut stages and 5 datasets (data plus 4 Monte Carlos), you'll find that managing these 6250 (count them!) histograms can be much easier by using a two-dimensional array of MatrixOfHistogram objects than hassling with 4-dimansional histogram arrays, 5 levels of if-then-else indentations and all that, 2 levels of loops and all that.

Also, plotting and fitting these histograms can be done quite efficiently in our framework.

8.2.5 Why Can't I Express Cuts with Strings, as in root?

We don't like string parsing at run time.

First, it's a pain to write the parser, while the compiler already has one.

Second, using objects instead of strings makes it possible to detect many errors already at compile time rather than run time. This is a general idea behind the type-safe system of C++, and one of the main advantages over other languages like Smalltalk.

Since by using C++ we already have all the disadvantages of C++, like clumsy notation, the necessity to care about variable definitions and memory leaks, we think we should also take advantage of C++'s benefits. Otherwise, we'd be much better off using another language in the first place.

8.2.6 Loooking at Your Class Hierarchy, You Use Multiple Inheritance a Lot. Isn't that Dangerous?

In principle, we are no big fans of multiple inheritance. If we use it, we try to confine it to mix-in classes that only add behaviour, not data to other classes, similar to "interfaces" in java. However, we do not everywhere achieve that.

Anyway, since multiple inheritance is used a lot in the root source code (by modern standards probably overused), we figured we didn't have to restrain ourselves too much.

8.2.7 In the Constructors of Classes SFSetOfHistograms and SFMatrixOfHistograms, the Order of Arguments is Somewhat Suboptimal. Why?

Short answer: For historical reasons.

Long answer:

In general we tried to adhere to a number of guidelines when we defined constructors:

For classes that are derived from root classes, provide constructors that take the same arguments as in root, put additional arguments (if any) behind the arguments of the root classes. Similarly, for derived classes in general we try to put additional arguments for conistructors at the end.

For **SFSetOfHistograms** and **SFMatrixOfHistograms**, we wanted to provide the same constructors as for class **SFH1F**, plus the BiningFun objects.

At that time, we always passed references to **FloatFun** and **BaseCut** objects, so we couldn't have default arguments, and at that time we didn't have **FillIterator** objects. If we did it again, we'd do it differently, but changing the order of the arguments would break all user code, and we don't think that's worth it.

In short: We did it wrong, but to correct it would mean a lot of work for our users, who are too dear to us to demand that.

8.3 History

Changes 8.7.04

- New mechanism for generic bank access on ODS
- New H1 examples

Changes 5./6.7.04

- New documentation page on Basic abstractions
- New class **IntFun**, with comparison operators to implement cuts on integer value quantities like number of tracks.
- Completed doxygen documentation of basically all classes

Changes 29./30.6.04

- New templates **NTFloatFun**, **NTFillIterator** to allow easy access to ntuple variables.
- Additional functions like sin, cos, sqrt of **FloatFun** objects are available
- Cuts can now be defined using comparison operators between **FloatFun** objects and floats. Comparisons can be changed: 1 < x < 2 means the same as (1 < x) && (x < 2)
- New examples in ntexamples: DataLoop precisely reproduces the stuff in examples/DataLoop, but without any additional **FloatFun** classes. comp and CompLoop shows the power of the new comparison operators.
- The baseclass **FloatFun** is now defines in **FloatFunBase.h**, which is automatically included by **FloatFun.h**

Changes 21./22.6.04

New caching mechanism: Introduced abstract base class **CachedO** (derived from **RegO**), derived classes SimpleCachedFloatFun, IteratorCachedFloatFun, SimpleCachedCut, and IteratorCachedCut from it to provide simple bases for cached objects. Adapted **EventLoop** to invalidate all caches in the loop() method.

Changed classes **SetOfHistograms**, **SFSetOfHistograms**, **MatrixOfHistograms**, and **SFMatrixOfHistograms** to use "PoR" classes everywhere (like **SFH1F** etc), and to take Binning objects for the axis binning in the constructor.

Changes 10.6.04

Don't distinguish anymore between registered histograms and other registered entities. Class **SFO** now is the base class for everything that has a Fill() method, including **SFSetOfHistograms** and **SFMatrixOfHistograms**. This makes the Fill() method in **SFROList** much easier and decouples **SFROList** from **SFSetOfHistograms** and **SFMatrixOfHistograms**; future self-filling collections will thereby automatically be filled as well.

- class RegH replaces by class **RegO**
- class HList replaced by class **ROList**
- class SFH replaced by class **SFO**

- class SFHList replaced by class **SFROList**

New Files:

- **RegO.h**
- **RegO.C**
- **ROList.h**
- **ROList.C**
- **ROListPoR.h**
- **SFO.h**
- **SFROList.h**
- **SFROList.C**
- **history.txt**

Deleted files:

- RegH.h
- RegH.C
- HList.h
- HList.C
- HListPoR.h
- SFH.h
- SFHList.h
- SFHList.C

8.4 What is a SetOfHistograms?

8.5 H1 Specific Extensions of the SFH Package

H1's object oriented software has a very intricate data structure, based on a variant of RooT trees called H1Tree. H1 data is stored in 3 different layers, called HAT, mODS (micro ODS), and ODS (Object Data Store).

- HAT data is very small (order hundreds of bytes per event) and contains mostly "global" event variables such as total energy, total pt, etc.
- mODS data is small (tens of kilobytes per event) and contains information on selected tracks, clusters, and particles.
- ODS data is completely equivalent to DST (Data Summary Tape) data, and contains all reconstructed tracks, clusters, etc.

To allow easy access to the data stored on HAT and ODS, the SFH package provides additional classes derived from **FloatFun**, **IntFun**, and **FillIterator**, plus a set of predefined variables that can be used to immediately access large parts of the available data.

These classes are declared in jbltools/sfh/h1/H1Funs.h, the predefined variables for HAT access are in jbltools/sfh/h1/H1HATVariables.h, and the variables for mODS access are in jbltools/sfh/h1/H1ModsVariables.h.

8.5.1 HAT Access

File jbltools/sfh/h1/H1HATVariables.h declares a full set of variables that allow easy access to all HAT variables (currently of H1OO version 2.6.3). For a variable "Xxx" (e.g., "RunNumber", "Iclass", "Il1ac", "Epz", or "Q2da"), there exists a variable Hat_Xxx (e.g., Hat_RunNumber, Hat_Iclass, Hat_Il1ac, Hat_Epz, or Hat_Q2da).

Depending on whether the variable concerned is of an integer type (char/byte, short, int) or of floating point type (float or double), the corresponding variable is either a **IntFun** or a **FloatFun**.

Such variables can immediately be used like this:

```
#include "jbltools/sfh/h1/H1HatVariables.h"
DataLoop::DataLoop()
{
   SFH1F *hRunNumber = new SFH1F ("RunNumber", "RunNumber", 100, 300000., 400000., this,
                                  Hat_RunNumber.Float());
   SFH1F *hQ2da      = new SFH1F ("Q2da", "Q2da", 100, 0., 100., this,
                                  Hat_Q2da);
   SFH1F *hEpz       = new SFH1F ("Epz", "Epz", 100, 0., 100., this,
                                  Hat_Epz);
}
```

To be used as a **FloatFun**, a ".Float()" must be appended to the **IntFun** type objects.

For array type data, like subtrigger information in Hat_Il1ac, we have defined an operator[] that can be used like this:

```
#include "jbltools/sfh/FillIterators.h"
#include "jbltools/sfh/h1/H1HatVariables.h"
DataLoop::DataLoop()
{
  // An iterator that loops from 0 to 127
  FillIterator& st_iter = *new RangeIterator (0, 127);
  // A histogram that shows how often raw subtriggers 0 to 127 have fired
```

```
  SFH1F *h_rwst = new SFH1F ("l1rw",  "Raw Subtriggers", 128, -0.5, 127.5, this,
                             st_iter.Float(), Hat_Il1rw[st_iter]==1, 0, st_iter);
  // A histogram that shows Q2 for ST9 events
  SFH1F *h_q2   = new SFH1F ("Q2",  "Q2(DA), ST9", 100, 0., 100., this,
                             Hat_Q2da, Hat_Il1ac[9]==1);
}
```

The operator[] actually returns an object (of a subclass of **FloatFun** or **IntFun**), so we could also write

```
  IntFun&  Hat_Il1ac[st_iter];
  IntFun&  Hat_Il1ac[9];
  BaseCut& Hat_Il1ac[st_iter]==1;
  BaseCut& Hat_Il1ac[9]==1;
```

8.5.2 mODS Access

The access to mODS is a bit more complicated, because a large part of the mODS information is stored in H1ClonesArray objects, and because all the different particles or pseudo-particles belong to different classes.

However, to make a few simple things simple, we have defined a number of predefined variables in file jbltools/sfh/h1/H1ModsVariables.h.

For once, there exist objects that represent the number of particle candidates in the different arrays; these are called mods_NumParticles, mods_NumFNCPart, etc., and are basically of type **IntFun**.

These numbers are, however, not very interesing, especially, because equivalent HAT variables exist anyway.

Additionally, we have defined iterators over these arrays, with names like mods_Particles, mods_-FNCPart, etc.

These iterators are of a special type, so that they can return function objects that give you basic quantities like momentum P, transverse momentum Pt, Mass, etc. These methods have the same names as in class H1Part:

```
\code
#include "jbltools/sfh/h1/H1ModsJpsi.h"
DataLoop::DataLoop()
{
  // A histogram that shows the mass of all J/psi candidates
  SFH1F *h_mass = new SFH1F ("jpsimass",  "J/psi mass", 40, 2., 4., this,
                             mods_JPsi.GetMass());
  // A histogram that shows the pt distribution of all tracks
  // with 20 deg < theta < 160deg:
  SFH1F *h_pt   = new SFH1F ("pt",  "Track pt", 100, 0., 10., this,
                             mods_SelectedTracks.GetPt(),
                             20 < 57.2958*mods_SelectedTracks.GetTheta() < 160);
}
```

Additionally, one can use access methods that are specific to the H1Part subclass under consideration like this:

```
  SFH1F *h_t    = new SFH1F ("jpsit",  "J/psi t", 40, 0., 1., this,
                             mods_JPsi.Float(&H1PartJpsi::Get_t));
}
```

The predefined iterators have a method **Float()** that returns a **FloatFun** object which uses an access methyod (in our case H1PartJpsi::Get_t()).

More complicated things, like cuts on the properties of J/psi decay particles, will, however, involve the writing of your own cut classes, for example:

```
#include "jbltools/sfh/BaseCut.h"
#include "H1Mods/H1PartSelTrack.h"
#include "H1Mods/H1PartJPsi.h"
#include "H1Mods/H1PartJPsiArrayPtr.h"
#include "H1Mods/H1PartMuon.h"

static H1PartJPsiArrayPtr jpsi;
// Base class: Stores the iterator
class JpsiBaseCut: public BaseCut {
  public:
    JpsiBaseCut (FillIterator& iter_) : iter(iter_) {};
  protected:
    FillIterator& iter;
};

// Selects only elastic J/psi candidates
class JpsiElasticCut: public JpsiBaseCut {
  public:
    JpsiElasticCut (FillIterator& iter_) : JpsiBaseCut(iter_) {};
    virtual bool operator() () const {
      return (iter() > -1) && (jpsi[iter()]->IsElastic() == kTRUE);
    };
};

// Selects only J/psi candidates decaying to 2 identified muons
class Jpsi2MuCut: public JpsiBaseCut {
  public:
    Jpsi2MuCut (FillIterator& iter_) : JpsiBaseCut(iter_) {};
    virtual bool operator() () const {
      return (iter() > -1) &&
             (jpsi[iter()]->GetTrack1()->GetParticle()->IsMuon()) &&
             (jpsi[iter()]->GetTrack2()->GetParticle()->IsMuon());
    };
};
```

But this code is still easy enough to be completely contained in a single header file, without the need for a .C file, and the resulting analysis code will look very clean:

```
  BaseCut&  jpsielastic    = *new JpsiElasticCut (mods_JPsi);
  BaseCut&  jpsi2mu        = *new Jpsi2MuCut (mods_JPsi);

  FloatFun& jpsimass       = mods_JPsi.GetMass();

  Binning massbinning (40, 2., 4.);

  // Mass histogram for elastic J/psi mesons decaying to 2 muons:
  SFH1F *hJpsiElasticTT   = new SFH1F ("jpsi", "J/psi->(#mu-#mu)", massbinning, this,
                                       jpsimass,
                                       jpsielastic && jpsi2mu,
                                       0,
                                       mods_JPsi);
```

8.5.3 ODS Access: Generic Bank Access

The data on H1 ODS contains the full DST information. Part of that data is available through special tracks and cluster classes, part is only available through access to objects that correspond to the original DST banks.

The SFH package provides means for a generic access to bank objects through classes defined in jbltools/sfh/h1/H1BankFuns.h.

Here is some example code:

```
#include "jbltools/sfh/h1/H1BankFuns.h"
DataLoop::DataLoop()
{
  // An object that represents the dtra (DST track) bank:
  JBLH1Bank& dtra = *new JBLH1Bank ("DTRA");
  // An iterator that loops over all DTRA tracks
  JBLH1BankIterator& dtra_it = dtra.iter();
  // A histogram that the pt of all DTRA tracks with 20<theta<160 deg:
  // (column 0 of the bank contains 1/pt, signed,
  //  column 2 contains the theta value):
  SFH *h = new SFH1F ("dtrapt", "DTRA track pt", 100, 0., 10., this,
                      1/abs(dtra_it.GetFloat(0)),              // the track pt
                      20 < 57.2958*dtra_it.GetFloat(2) < 160,  // cut on theta
                      0,                                       // weight 1
                      dtra_it);                                // the track iterator
}
```

Objects of type JBLH1Bank represent an H1 bank. Upon creation, they create an H1BankPtr object that points to the bank.

An JBLH1Bank object has a method iter() that returns an JBLH1BankIterator object, i.e. a special **FillIterator**, that can be used to loop over the rows of the bank.

An JBLH1BankIterator object has methods GetFloat(i), GetFloatFromInt(i), and GetInt(i) which create function objects that return the value of column i of the row selected by the iterator. GetFloat and GetFloatFromInt return **FloatFun** objects for columns that are float- or int-valued, respectively, and GetInt returns an **IntFun** object for an int-valued column.

Therefore, one could also write code like this:

```
  JBLH1Bank& dtra = *new JBLH1Bank ("DTRA");
  JBLH1BankIterator& dtra_it = dtra.iter();

  // Azimuthal angle phi
  FloatFun& dtraPhi   = dtra_it.GetFloat(1);
  // Track transverse momentum
  FloatFun& dtraPt    = 1./abs(dtra_it.GetFloat(0));
  // Link to forward track, 0 for central tracks
  IntFun&   dtraForw  = dtra_it.GetInt(14);

  // Cut on theta:
  BaseCut& thetaCut = 20 < 57.2958*dtra_it.GetFloat(2) < 160;
  // Cut on central tracks
  BaseCut& centralCut = dtra_it.GetInt(14)==0;
```

Index