Subject: Effective C++
Posted by Volker Friese on Fri, 17 Feb 2012 08:09:14 GMT
View Forum Message <> Reply to Message

We want to remove, throughout the repository, all compiler warnings related to the warning level -Weffc++. These mostly concern member intialisation and assignment/copy constructors for classes with pointer members (see CBM Software Meeting, 16 February 2012).

A SVN ticket (#40) was created for this project. Please assign all corresponding commits to this ticket.

Subject: How to see the -Weffc++ warnings in a directory
Posted by Volker Friese on Fri, 17 Feb 2012 08:12:51 GMT
View Forum Message <> Reply to Message

Here now the description how to use CMake to publish the results on our DashBoard.

Add the following lines for the wanted directories, e.g.

"rich/"
"littrack/"
"KF/"
"L1/"

at the end of this part in the CTestCustom.cmake file after FairTSBufferFunctional.h and comment the lines with
directories you are not interested in.
If you want to avoid svn updates please comment the following line in CbmRoot_test.cmake

CTEST_UPDATE (SOURCE "${CTEST_SOURCE_DIRECTORY}")

Create a file (e.g. Dart_test.cfg) with the following content which defines the input for the Dart.sh script.

export LINUX_FLAVOUR=Test_Rich # change it to some meaningful name
export FAIRSOFT_VERSION=may11 # your version of the external packages
export SIMPATH=$HOME/software/SnowLeo/fair/fairsoft_may11 # your SIMPATH
export BUILDDIR=$HOME/software/SnowLeo/fair/cbmroot_dev/build # the build directory you want to use
export SOURCEDIR=$HOME/software/SnowLeo/fair/cbmroot_dev # your source directory

Run the shell script in the main source directory as follows (The last parameter is the file you have cerated in the last step)

./Dart.sh Nightly Dart_test.cfg

This will configure, build and test the code and will in the end send the results to the dashboard.

If you have any questions don't hesitate to ask me.

Ciao

Florian

Subject: Effective C++: Member initialisation
Posted by Volker Friese on Fri, 17 Feb 2012 08:14:33 GMT
View Forum Message <> Reply to Message

Hi

In the following I will explain how one get rid of one of the two most common warnings which show up when one switch on the effc++ warnings from gcc. The second warning is described in the
topic How to get rid of -Weffc++ warnings (part 2).

The first warning complaines about class members which are not initialized in the member initialization list. Please check example output below.

Quote:
warning: 'CbmFieldMap::fFileName' should be initialized in the member initialization list

Before we start with a real example from CBM code we have to do
some theory about C++ (taken from Scot Meyers Effective C++).

The problem is that one can't rely on that C++ will initialize Objects during declaration. C++ sometimes initialize the Objects during declaration, sometimes it doesn't do it.

int x;

In the above example x will be initialized (with 0) or not depending on the context. Due to this reason it is best practice to initialize the variables before usage. See example below.

int x = 0;
const char *text = "CBM is the best experiment ever."

When working with objects the initialization of all data members should be done in the constructor.

Coming back now to the CBM example. As example the class CbmFieldMap is taken.

The data members of this class are the following (taken from the header file)

```
  TString  fFileName;
  Double_t fScale;
  Double_t fPosX, fPosY, fPosZ;
  Double_t fXmin, fXmax, fXstep;
  Double_t fYmin, fYmax, fYstep;
  Double_t fZmin, fZmax, fZstep;
  Int_t    fNx, fNy, fNz;
  TArrayF* fBx;
  TArrayF* fBy;
  TArrayF* fBz;
  Double_t fHa[2][2][2];
  Double_t fHb[2][2];
  Double_t fHc[2];
```

On of the constructors looks like

```
CbmFieldMap::CbmFieldMap()
{
  fPosX  = fPosY  = fPosZ  = 0.;
  fXmin  = fYmin  = fZmin  = 0.;
  fXmax  = fYmax  = fZmax  = 0.;
  fXstep = fYstep = fZstep = 0.;
  fNx    = fNy    = fNz    = 0;
  fScale = 1.;
  fBx    = fBy    = fBz    = NULL;
  fPosX = fPosY = fPosZ = 0.;
  fName     = "";
  fFileName = "";
  fType = 1;
}
```

To all the data memebers values are assigned in the body of the constructor. To mention here is the fact that one assignes values to fPosX, fPosY and fPosZ two times which shouldn't hurt but is definetly not something which is intended. The other point
to mention is the assignment of values to members of the base class. This is necessary because there is no appropriate constructor of the base class.

The problem here is that the data members have in the end the expected values but it is not the best solution for the task.
The C++ rules define that alle data members should be initialized before the body of the constructor is entered. In the above example the values of the data members were not initialized but assigned. The initialization take place when the default constructors are called before entering the functions body. This is not true for all the integrated types (int, float ...), because here it is not guaranteed that the data members are initalized at all before entering the function body.
There is much more in the reference (Effective C++), but as a rule of thumb you should remeber to initialize all data members in in the member initialization list. The order of the data members should be the same as the order they are declared in the header file.

The corrected constructur now looks like the one below. Before initializing the data members of the class the default constructor of the base class is "called". Then the data members show up in the same order as they are declared in the header file. In the body of the constructor only the default values to all elements of the arrays are assigned. I don't know any way to initalize the elements of an array in the member initalization list.
Also here the values of data members of the base class are
assigned because there is no appropriate constructpr of the base class. If there would be such a constructor one could call this constructor in the member initilization list.

```
CbmFieldMap::CbmFieldMap()
  : FairField(),
    fFileName(""),
    fScale(1.),
    fPosX(0.),
    fPosY(0.),
    fPosZ(0.),
    fXmin(0.),
    fXmax(0.),
    fXstep(0.),
    fYmin(0.),
    fYmax(0.),
    fYstep(0.),
    fZmin(0.),
    fZmax(0.),
    fZstep(0.),
    fNx(0),
    fNy(0),
    fNz(0),
    fBx(NULL),
    fBy(NULL),
    fBz(NULL)
{
  // Initilization of arrays is to my knowledge not
  // possible in member initalization lists
  for (Int_t i=0; i < 2 ; i++) {
    fHc[i]=0;
    for (Int_t j=0; j < 2 ; j++) {
      fHb[i][j]=0;
      for (Int_t k=0; k < 2 ; k++) {
  fHa[i][j][k]=0;
      }
    }
  }
  // Assign values to data members of base classes
  // There is no appropriate constructor of the base
  // class.
  fName = "";
  fType = 1;
}
```

I hope this topic helps to understand the warning and how to correct the code. Remarks and corrections are very welcome

Ciao

Florian

Subject: Effective C++: Copy constructor for classes with pointer members
Posted by Volker Friese on Fri, 17 Feb 2012 08:16:34 GMT
View Forum Message <> Reply to Message

Hi

The second common warning is the following.

Quote:
warning: 'class CbmFieldMap' has pointer data members
warning: but does not override 'CbmFieldMap(const CbmFieldMap&)'
warning: or 'operator=(const CbmFieldMap&)'

The problem here is that both functions are automatically created by the compiler if the are not declared. If the class has pointer data members one should explicetly define what both functions should do with this pointers. If neither of the both functions are needed the solution of the problem is quite simple. If one needs the functions it get complicated (not discussed in this topic).

All of the automatically generated functions are public and can be accessed by everybody. If you declare both functions as private but don't implement them the problem is solved. The compiler will not generate the functions automatically and since the functions are private nobody can use them.
If now any of the two functions is called from another class there will be a compiletime error because the function is declared private.
If the functions are called from other functions of the same class or functions of a friend class there will be no compiletime but a linktime error. So in any case it will not be possible to use any of the two functions.

Add the following part to the header file.

```
private:
  CbmFieldMap(const CbmFieldMap&);
  CbmFieldMap operator=(const CbmFieldMap&);
```

Thats it.

Subject: If you really want the copy constructor

The copy constructor which is created by default by the compiler if you do not do it yourself will just copy each data member by value. If your class just contains normal type members, this is usually want you want.

If your class, however, contains a pointer member, things are different, because the default copy constructor will not copy the object the pointer points to, but just the pointer value. So, the pointer member of the new object will point to the same location as that of the original object. If the original object is removed, then its destructor will delete the object its pointer member points to (if properly implemented). So, the pointer member of the new, copied object, will point to an invalid address. The best you can get is then a segmentation fault.

If for such a class you do not need the copy or assignment constructor, follow the instructions given in the mother posting. If you do need it, you have to implement it yourself. In most cases that means to manually copy the normal members one by one. In addition, for the pointer member you have to instantiate an object it points to, and use the copy constructor of this object to create it from the one of the original class.

Example:

```
class Example {

double firstMember;
FairMCPoint* pointerMember;

// Assignment operator
Example& operator = (Example const & source) {
firstMember = source.FirstMember;
pointerMember = new FairMCPoint(*(source.pointerMember));
}

// Copy constructor, using the assignment operator
Example(Example const & source) { *this = source; }

}
```

This is just an example, there is no general solution. You have to decide on your own what in your particular case the assignment operator and the copy constructor are supoosed to do.