Subject: If you really want the copy constructor
Posted by Volker Friese on Fri, 17 Feb 2012 08:42:08 GMT
View Forum Message <> Reply to Message

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.