

Re: Copy Constructors and Assignment Operator, When should I use?

Re: Copy Constructors and Assignment Operator, When should I use?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2008-03/msg00084.html>

- *From:* Ulrich Eckhardt <eckhardt@xxxxxxxxxxxxxxxx>
 - *Date:* Mon, 03 Mar 2008 15:02:06 +0100
-

rockdale wrote:

Sorry my code example has a lot of typos. I extract my current code and changed those classes names.

So from what you wrote:

No. It works for any datatype that defines its own copy constructor that you don't have to invoke explicitly, like any standard container, strings, any POD, any classes containing those (which is actually kind of recursive).

```
ItemSet set1;  
set1.load();
```

```
ItemSet set2;
```

I simply assign set1 to set2 like this

```
set2 = set1;
```

will this assignment will copy whatever in set1 to set2? (deep copy, even the vector items?),

well my coworker told me that I need to have a define my own copy constructor for ItemSet, ItemListA and ItemListB for it works properly.

in the copy constructor of ItemSet,

```
ItemSet(ItemSet& toCopy){
```

Re: Copy Constructors and Assignment Operator, When should I use?

```
//copy constructor
a = toCopy.a;
b = toCopy.b;
}
```

Two things here:

1. A copy constructor usually takes a reference to a const object, not a plain reference. If the book or the coworkers you are learning from have this wrong I would strongly suggest replacing/upgrading them. Getting this wrong means getting the very basics of C++ wrong and that is a really bad sign.
2. If ItemSet only contains the fields 'a' and 'b', this copy constructor almost does the same as the compiler-generated one would do (only that it's using assignment instead of initialisation), so there is no need to write it manually.

The compiler-generated one would look like this:

```
ItemSet(ItemSet const& toCopy):
a(toCopy.a),
b(toCopy.b)
{ }
```

I need to define copy constructor in my ItemListA and ItemListB classes, basically, to manually copy the vector from the toCopy, is this statement true?

Hard to say if this is true, I seriously don't understand what you mean here.

```
struct ItemA{
int aInt;
std::string aString;
} ;

struct ItemB{
int bInt;
std::string bString;
time_t bTime;
} ;
```

Both of these types are freely copyable and assignable and you don't have to define any copy constructors or assignment operators.

Re: Copy Constructors and Assignment Operator, When should I use?

```
void load(){  
a.load():  
b.load();  
}
```

One thing please: do not retype code but use cut'n'paste. The above is syntactically wrong and this makes it impossible to guess what code you are referring to.

```
std::vector <ItemA> m_vecA;
```

std::vector is also freely copyable and assignable, provided that is true for its elements. This last restriction e.g. applies to vectors of pointers, which are copyable but which only copy pointers and not the objects they point to.

[full quote]

Please refrain from top posting. The quoted text is for you to reference it, if you don't use it please delete it. Google for "Usenet netiquette".

Uli

—

C++ FAQ: <http://parashift.com/c++-faq-lite>

Sator Laser GmbH

Geschäftsführer: Michael Wöhrmann, Amtsgericht Hamburg HR B62 932

.