Subject: Who is doing the coping? [implicit-copy | deep-copy | pick constructor]
Posted by navi on Sat, 15 Dec 2012 04:24:53 GMT

View Forum Message <> Reply to Message

```cpp
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

class myobject: public Moveable<myobject>{
 public:
 String a;
 String b;
 String c;
};

GUI_APP_MAIN
{
 // gui code

 TopWindow w;
 LineEdit e;

 e.HSizePos().VSizePos();
 w.SetRect(0,0,400,600);
 w<<e;

 // codes for test example

 Vector<myobject> v;
 myobject tmp;
 String s;

 for(int i=0; i<5; i++){
 tmp.a=Format("a value=%d",i);
 tmp.b=Format("b value=%d",i);
 tmp.c=Format("c value=%d",i);

 v.Add(tmp);

 s<<"\nObject:"<<i<<"\n";
 s<<tmp.a<<",\n"<<tmp.b<<",\n"<<tmp.c<<"\n";

 }

 e.Set(s);

 w.Run();
```

```
}
```

http://www.ultimatepp.org/src$Core$Vector$en-us.html
class Vector : public MoveableAndDeepCopyOption< Vector<T> >

has 3 Add method,

1. T& Add(): Adds new default constructed element to Vector.

2. T& Add(const T& x) : Requires T to have deep copy constructor.

3. T& AddPick(pick_ T& x) : Requires T to have pick constructor.

In above code 'myobject' does not have any pick constructor, or deep copy constructor. As on the example code, if 'tmp' object is added to a vector using Add(T& x) method, then 'tmp' gets nicely copied into the new vector element. without nullifying or destructing the 'tmp' object. since I did not write a deep copy constructor, so:

* who/what is doing the coping?
* which contractor is getting called? (if any!?)

I have tried reading the followings but still can not understand why is the above code working?

 http://www.ultimatepp.org/srcdoc$Core$pick_$en-us.html
 http://www.ultimatepp.org/srcdoc$Core$PickTypes$en-us.html
 http://www.ultimatepp.org/srcdoc$Core$Moveable$en-us.html

however, after reading this: http://www.cplusplus.com/articles/y8hv0pDG/

I concluded that C++ is writing a implicit copy constructor which is doing a member-wise copy of 'myobject' and neither U++ deep copy nor pick is being used. since myobject only has 3 String and String class has operator= overloaded, so implicit copy constructor is able to carryout its member-wise copy. Is this correct?