Subject: Deepcopying One container
Posted by dolik.rce on Mon, 02 Nov 2009 08:37:44 GMT
View Forum Message <> Reply to Message

Hi,

I've met following problem with polymorphic classes in One containers. Consider following code:#include <Core/Core.h>
using namespace Upp;

```
class A{
public:
 virtual void DoSmthng(){Cout()<<"I'm A.\n";};
};
class B:public A{
public:
 virtual void DoSmthng(){Cout()<<"I'm B.\n";};
};

CONSOLE_APP_MAIN{
 One<A> a=new A;
 One<A> b=new B;
 One<A> c;

 c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng();
```

The output of this is I'm A.
I'm A.It surprised me at first, but after looking in the implementation of operator<<=, I understood that this is to be expected (that is not a bug).

The question is: Is there some workaround to make a copy of One without loosing the information about the type it stores? I mean to make it work same way as if you do

```
 c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng();
```

but without a and b beeing picked. Is that even posible?

Thanks for any responses.
Regards,
Honza

Subject: Re: Deepcopying One container
Posted by cbpporter on Mon, 02 Nov 2009 09:21:22 GMT
View Forum Message <> Reply to Message

Well using references works:

```
{
 One<A> &c=a;
 c->DoSmthng();
}
{
 One<A> &c=b;
 c->DoSmthng();
}
```

But since One is an encapsulation of a pointer, I believe it should not inherit <<= from DeepCopyOption, but rather do a low level copy of the pointer so that the placement new is avoided.

Subject: Re: Deepcopying One container
Posted by mirek on Mon, 02 Nov 2009 10:14:28 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Mon, 02 November 2009 03:37Hi,

I've met following problem with polymorphic classes in One containers. Consider following code:#include <Core/Core.h>
using namespace Upp;

```
class A{
public:
 virtual void DoSmthng(){Cout()<<"I'm A.\n";};
};
class B:public A{
public:
 virtual void DoSmthng(){Cout()<<"I'm B.\n";};
};

CONSOLE_APP_MAIN{
 One<A> a=new A;
 One<A> b=new B;
 One<A> c;

 c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng();The output of this is I'm A.
```

I'm A.It surprised me at first, but after looking in the implementation of operator<<=, I understood that this is to be expected (that is not a bug).

The question is: Is there some workaround to make a copy of One without loosing the information about the type it stores? I mean to make it work same way as if you do c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng(); but without a and b beeing picked. Is that even posible?

Thanks for any responses.
Regards,
Honza

Surprisingly, yes, we can provide polymorphic copies - by overloading DeepCopyNew.

That can be simplified by using PolyDeepCopyNew, using virtual
Copy method.

In reality, I have never really used polymorphic deep copy, it looks a little bit tricky to me. What is your usage scenario?

Mirek

Subject: Re: Deepcopying One container
Posted by dolik.rce on Mon, 02 Nov 2009 16:57:50 GMT
View Forum Message <> Reply to Message

Thanks for good news, Mirek!
Quote:What is your usage scenario?
I was afraid you will ask Now I have to show everyone the ugly scheme I've come up with

I use descendants of my class PlotSymbol to provide a methods used to paint different marks in plot. Then I have a VectorMap<String,One<Plotsymbol> >. Every new symbol class is "registered" by adding into this VectorMap, so that user can be presented with a list of available symbols and choose which one to use. Everytime user changes his mind and selects new symbol, I would like just to copy the One<PlotSymbol> into local variable (and setup some additional parameters, e.g. colors or size).

So basicaly what I need is to make a copy of the underlaying PlotSymbol, without knowing what type is it. The information in manual made me believe it is possible, but then I've found that it only works when the source is picked.

I'll try the solution you suggested...

To cbporter: Thanks for your reply too. Unfortunately I need a full copy of the pointed object so I can change some additional properties without affecting the original...

Subject: Re: Deepcopying One container
Posted by dolik.rce on Tue, 03 Nov 2009 13:21:12 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 02 November 2009 11:14
Surprisingly, yes, we can provide polymorphic copies - by overloading DeepCopyNew.

That can be simplified by using PolyDeepCopyNew, using virtual
Copy method.

In reality, I have never really used polymorphic deep copy, it looks a little bit tricky to me. What is your usage scenario?

Mirek

PolyDeepCopyNew works like a charm! Thank you very much Mirek, the solution was trully genial in its simplicity.

Just for future reference, the solution is something like this:#include <Core/Core.h>
using namespace Upp;

```
class A: public PolyDeepCopyNew<A>{
public:
 virtual void DoSmthng(){Cout()<<"I'm A.\n";};
 virtual A* Copy()const{return new A;}
};
class B:public A{
public:
 virtual void DoSmthng(){Cout()<<"I'm B.\n";};
 virtual B* Copy()const{return new B;}
};

CONSOLE_APP_MAIN{
 One<A> a=new A;
 One<A> b=new B;
 One<A> c;

 c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng();
}
```

Honza

Subject: Re: Deepcopying One container

Posted by mirek on Tue, 03 Nov 2009 18:45:39 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Tue, 03 November 2009 08:21luzr wrote on Mon, 02 November 2009 11:14
Surprisingly, yes, we can provide polymorphic copies - by overloading DeepCopyNew.

That can be simplified by using PolyDeepCopyNew, using virtual
Copy method.

In reality, I have never really used polymorphic deep copy, it looks a little bit tricky to me. What is your usage scenario?

Mirek

PolyDeepCopyNew works like a charm! Thank you very much Mirek, the solution was trully genial in its simplicity.

Just for future reference, the solution is something like this:#include <Core/Core.h>
using namespace Upp;

```
class A: public PolyDeepCopyNew<A>{
public:
 virtual void DoSmthng(){Cout()<<"I'm A.\n";};
 virtual A* Copy()const{return new A;}
};
class B:public A{
public:
 virtual void DoSmthng(){Cout()<<"I'm B.\n";};
 virtual B* Copy()const{return new B;}
};

CONSOLE_APP_MAIN{
 One<A> a=new A;
 One<A> b=new B;
 One<A> c;

 c<<=a;
 c->DoSmthng();

 c<<=b;
 c->DoSmthng();
}
```

Honza

IMO, the example is a very little bit misleading - Copy should deepcopy some content...

Mirek

Subject: Re: Deepcopying One container
Posted by dolik.rce on Wed, 04 Nov 2009 07:40:48 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 03 November 2009 19:45
IMO, the example is a very little bit misleading - Copy should deepcopy some content...

Mirek

Well, the example is oversimplified. In most cases it probably should deepcopy the content - but not necessarily every time. For example in my case, all I needed was to preserve the information about type to keep the virtual methods accessible. (And I overwrite the content anyway.)

Honza