Subject: PolyDeepCopyNew: MSC / GCC differ in behaviour
Posted by kohait00 on Thu, 14 Oct 2010 11:59:22 GMT
View Forum Message <> Reply to Message

hi people

i've got some issue using the PolyDeepCopyNew feature, where GCC and MSC behave differently, while this time MSC seems to do it right, or i did sth wrong. MSC compiles well, GCC doesnt. I use MSC9 and TDMGCC.

the problemarising is:
GCC seems not to be able to properly recognize the friend DeepCopyNew from PolyDeepCopyNew as an alternative to the template DeepCopyNew.

any help on this one is really appriciated.
i've added a testcase down there.

```
#include <Core/Core.h>

using namespace Upp;

//without this one beeing in namespace upp it doesnt work eaither
//suppose because of friend DeepCopyNew not beeing placed / referenced to template definition
NAMESPACE_UPP

template<class T, class B = EmptyClass>
class Copyable : public B
{
public:
 virtual ~Copyable() {}
 virtual T* Copy() const                 { return PartialCopy(); }
 virtual T* PartialCopy() const            = 0; //{ NEVER(); return NULL; }
};

template<class C, class B = EmptyClass>
class CopyableC : public Copyable<CopyableC<C,B>, B>
{
public:
 virtual const C& GetC() const             = 0;
 virtual C& GetC()                       = 0;

 operator const C&() const                { return GetC(); }
 operator C&()                           { return GetC(); }
};

template<class B, class C, class CB = EmptyClass>
class PolyCopyableC : public B, public PolyDeepCopyNew<CopyableC<C, CB>, CopyableC<C,
```

```
CB> >
{
public:
 virtual const C& GetC() const             { return *this; }
 virtual C& GetC()                        { return *this; }
};

END_UPP_NAMESPACE

//OWN CLASSES

//this one doesnt help
//NAMESPACE_UPP

//a common base interface, that should be available after copying, (i.e Offer() )
//is meant as an extension interface on very bottom
//should be implemented at very top
class CBase
{
public:
 virtual void Offer(int a) = 0;
};

//a common base class, which should be accessible via GetC (could be i.e Ctrl)
class Base {};

//a Base version (could be any Ctrl derive, i.e StaticText)
//neither Base nor Derived are to be changed, they are foreign
//thats why so complicated
class Derived : public Base {};

//extension of the whole thing, to make everything cloneable..
class Master : public PolyCopyableC<Derived, Base, CBase>
{
public:
 Master* PartialCopy() const { return new Master(); }
 virtual void Offer(int a) {}
};

//END_UPP_NAMESPACE

//with this one both work, but i'd like to avoid that and to have it automatic in PolyCopyableC
#if 0
NAMESPACE_UPP
template<>
inline CopyableC<Base, CBase>* DeepCopyNew(const CopyableC<Base, CBase>& x) {
 return x.Copy();
}
```

```
END_UPP_NAMESPACE
#endif

CONSOLE_APP_MAIN
{
 Array<CopyableC<Base, CBase> > a1, a2;
 a1.Add(new Master());
 a1 <<= a2;
}
```

File Attachments

1) Blate.rar, downloaded 249 times