

---

Subject: Derivating from Vector<>

Posted by [victorb](#) on Sat, 03 Mar 2007 19:12:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

I am trying to derivate a class from Vector<...> but I can't get it to work. Any help from the Upp community would be welcome.

Here is some sample code:

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

```
using namespace Upp;
```

```
class IntVector : public DeepCopyOption<IntVector, Vector<int> >
```

```
{
```

```
public:
```

```
IntVector(){Cout() << "IntVector\n"; }
```

```
virtual ~IntVector(){Cout() << "~IntVector\n";}
```

```
IntVector(const IntVector &src, int) {
```

```
  ::new IntVector;
```

```
  Vector<int>(src, 0);
```

```
  name = src.name;
```

```
  Cout() << "DCC\n";
```

```
}
```

```
String name;
```

```
String ToString(void) {
```

```
  String dump;
```

```
  dump << name << " ";
```

```
  if (IsPicked()) return dump << "Picked";
```

```
  for (int i = 0; i < GetCount(); i++) {
```

```
    dump << At(i) << " ";
```

```
  }
```

```
  return dump;
```

```
}
```

```
};
```

```
CONSOLE_APP_MAIN
```

```
{
```

```
  Cout() << "iv\n";
```

```
  IntVector iv;
```

```
iv.name = "IV";

iv.Add(5);
iv.Add(6);

Cout() << "iv2\n";
IntVector iv2(iv, 0);

Cout() << iv.ToString() << "\n";
Cout() << iv2.ToString() << "\n";

}
```

I expect iv2 to be equal to iv at the end. There is probably something wrong with the deep copy constructor but I really can't figure it out.

Victor

---

---

Subject: Re: Deriving from Vector<>  
Posted by [victorb](#) on Sat, 03 Mar 2007 19:37:46 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

I have found something working

```
IntVector(const IntVector &src, int) {
  ::new IntVector;
  for (int i = 0; i < src.GetCount(); i++)
    At(i) = src[i];
  name = src.name;
  Cout() << "DCC\n";
}
```

I would need to add some check in order to make sure that src is not picked...  
But really there should be a nicer solution.

---

---

Subject: Re: Deriving from Vector<>  
Posted by [victorb](#) on Sat, 03 Mar 2007 19:48:25 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Another working solution:

```
IntVector(const IntVector &src, int) {
  ::new IntVector;
```

```
__DeepCopy(src);
name = src.name;
Cout() << "DCC\n";
}
```

But this require to change \_\_DeepCopy access to protected in Vcont.h

Anything better ?

---

---

Subject: Re: Deriving from Vector<>  
Posted by [victorb](#) on Sat, 03 Mar 2007 21:17:08 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Actually latest does not work (not recursive)

---

---

Subject: Re: Deriving from Vector<>  
Posted by [victorb](#) on Tue, 06 Mar 2007 23:54:48 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

I think that the solution is:

```
class IntVector : public Vector<int>
{
public:
    IntVector(){Cout() << "IntVector\n"; }

    virtual ~IntVector(){Cout() << "~IntVector\n";}

    IntVector(const IntVector &src, int) : Vector<int>(src, 0)
    {
        name = src.name;
    }

    String name;

};
```

---

---

Subject: Re: Deriving from Vector<>  
Posted by [victorb](#) on Wed, 07 Mar 2007 00:14:52 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

not so sure but I'll find...

---

---

Subject: Re: Derivating from Vector<>  
Posted by [victorb](#) on Wed, 07 Mar 2007 00:30:59 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

This seems to do the trick:

```
class IntVector : public DeepCopyOption<IntVector>, public Vector<int>
{
public:
    IntVector(){Cout() << "IntVector\n"; }

    virtual ~IntVector(){Cout() << "~IntVector\n";}

    IntVector(const IntVector &src, int) : Vector<int>(src, 0)
    {
        name = src.name;
        Cout() << "DCC\n";
    }

};
```

---

Subject: Re: Derivating from Vector<>  
Posted by [victorb](#) on Wed, 07 Mar 2007 16:05:16 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Simpler, without having to use multiple inheritance:

```
class IntVector : public DeepCopyOption<IntVector, Vector<int> >
{
public:
    IntVector(){Cout() << "IntVector\n"; }

    virtual ~IntVector(){Cout() << "~IntVector\n";}

    IntVector(const IntVector &src, int)
    {
        Append(src);
        name = src.name;
    }

    String name;

};
```

Should be the ultimate solution

---

---

Subject: Re: Derivating from Vector<>  
Posted by [mirek](#) on Thu, 08 Mar 2007 12:55:34 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Well, derivating from container is possible, but generally not quite a good idea. This is true both for U++Core and STL...

Mirek

---

---

Subject: Re: Derivating from Vector<>  
Posted by [victorb](#) on Thu, 08 Mar 2007 15:30:44 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

The main reasons because derivating from containers seem to be:

- 1- the lack of virtual destructor,
- 2- member function are not virtual then you can override them.

However in my case I am just adding a few properties to Vector<> and I don't want to have to rewrite the Add()/Remove()/... then I'll stick with inheritance. I agree that composition should be the preferred way in more complex cases.

Thanks,  
Victor

---