Subject: Re: NTL - "deep copy semantics"?
Posted by nixnixnix on Wed, 05 Sep 2007 02:43:07 GMT
View Forum Message <> Reply to Message

Thanks Mirek - that all works fine.

I have another related question: How exactly does Array support polymorphism? In the documentation it says that you pass a pointer but this seems meaningless as all data structures are polymorphic if you use them to store pointers as a pointer can point to anything. In any case I tried this and then copied my object and found that the pointer values are copied but not the objects they point to. A simplified analogue of my problem is below...

```
class VPoint
{
  Pointf m_pt;
  ...
  ...
}

class PointLayer
{
  WithDeepCopy <Array <VPoint> > m_pts;
  ...
  .../* lots of functions for using and drawing and editing VPoints */
}

class House : VPoint
{
  double m_fNoise;
  ...
}

class HouseLayer : PointLayer
{
  ...
}
```

Now I want my HouseLayer to contain an array of House objects in place of the VPoint objects in its base class but to use all the functionality that I've written into PointLayer but using House objects rather than VPoint objects.

I thought that the polymorphism in Array would let me do that but it appears not. I don't appear to be able to store House objects in the PointLayer::m_pts array and if I store pointers in my Array then my House objects and VPoint objects don't get copied.

I don't know if this is a UPP question or just to do with my lack of knowledge of C++. Any pointers (excuse the pun) would be really appreciated though.

Been searching through Stroustrup's book but not finding anything useful. I know I can do what I want to do if I just split my House object into the VPoint part and the House part and have another Array in my HouseLayer object and then keep the two arrays in sync but I have this overwhelming feeling that C++ is not that messy and that a more elegant solution exists.
