Subject: Questions about static casting Polymorphic Array Elements, iterator, Ptr and Pte

Posted by navi on Mon, 24 Dec 2012 00:58:02 GMT

View Forum Message <> Reply to Message

```
struct shape{
 int type;
}

struct circle : shape{
 circle(){ type=1; }
 int radius;
 int x, y;
}

struct triangle : shape{
 triangle(){ type=2; }
 int x[3], y[3];
}

struct rectangle : shape{
 rectangle(){ type=3; }
 int x[4],y[4];
}

Array<shape> a;

a.Add(new circle);
a.Add(new triangle);
a.Add(new rectangle);

if(a[1].type==2){

 triangle *m = static_cast<triangle *> (&a[1]);
}
```

in the above example, is this the correct Syntax & correct way to static casting Polymorphic Array Elements?triangle *m = static_cast<triangle *> (&a[1]);

How do I create an iterator and static cast the iterator instead? Is this a better way then the previous method?

What is the correct way to static cast an Element from Polymorphic Array?

As suggested in NTL Tutorial - Point.6 Polymorpic Array is great for the purpose of storing Polymorphic objects. no problem of object slicing. No need for memory management. So, the below question is purely for the curiosity sake.

Are there smart pointers in U++? "Stack Overflow" and other forums people suggesting to use STL vector of smart pointers to keep Polymorphic Elements. Is there any way to achieve that using U++ NTL? I have seen Ptr and Pte in the Manuel but confuse about what do they actually do? do they only assign null when object is destroyed or do they actually destroy pointing object (i.e. manages the object De-Allocation?) when they go out of scope?

Thanks & Regards
Navi