Subject: Re: |SOLVED] Vector of object: cast to inherited class
Posted by Novo on Mon, 16 Sep 2019 21:47:50 GMT
View Forum Message <> Reply to Message

Xemuth wrote on Mon, 16 September 2019 14:25Define Hello() as virtual on A allow us to redefine Hello() on Children but since B destructor will call A destructor at is exit and A don't know if child have possible redefinition of Hello() he will call is own Hello() declaration. :d
Still wrong. In case of "a->Hello();" a doesn't know about possible redefinition of Hello() as well, but it still prints "Hello from B".

And destructor of B doesn't call destructor of A. It works in a different way, although the order of calls is the same.

And you do not have to declare Hello() in A as virtual if you want to redefine it in B. Code below will compile.

```
struct A {
 virtual ~A() {
  Hello();
 }
 void Hello() const {
  Cout() << "Hello from A" << EOL;
 }
};

struct B : A {
 void Hello() const {
  Cout() << "Hello from B" << EOL;
 }
};
```