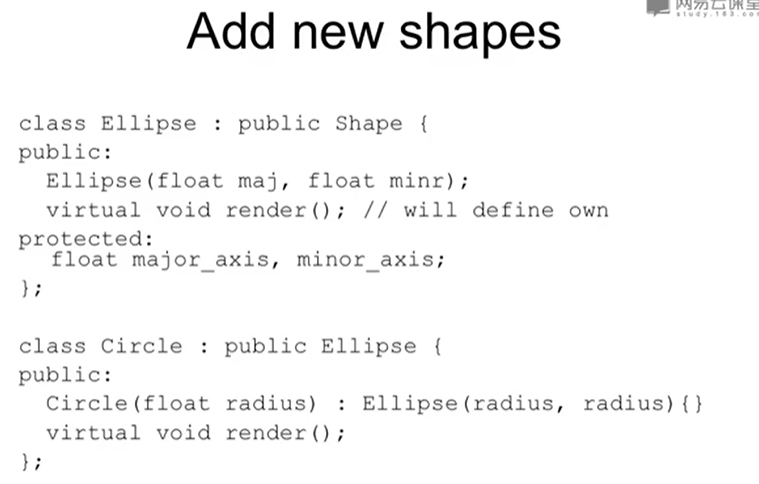
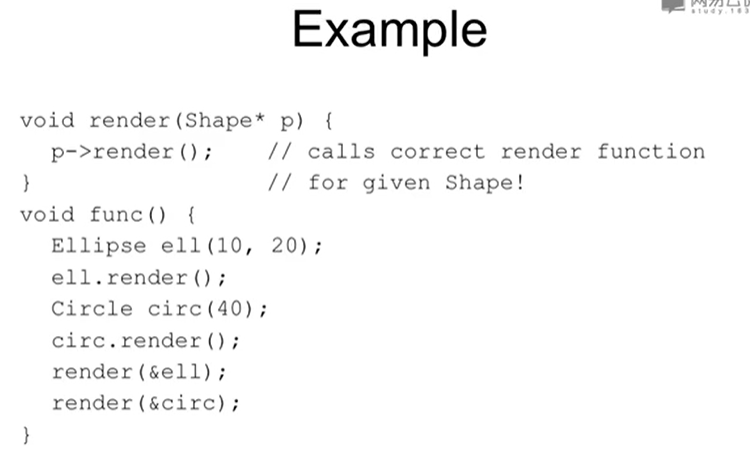


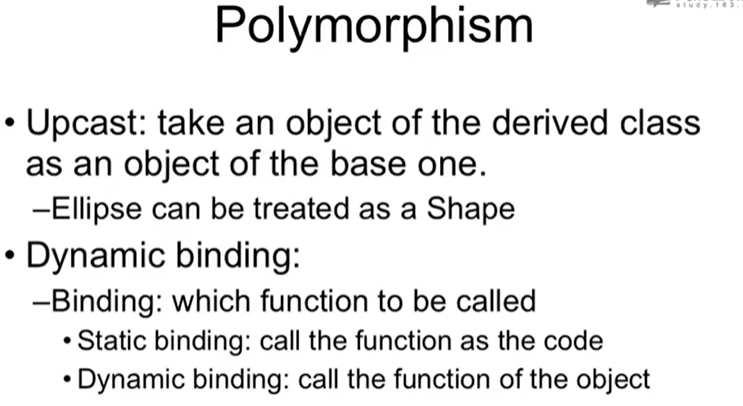
virtual所做的事是让**子类与父类同名函数有联系，因为常规函数 在CPP中子类与父类同名函数** 无联系。





last 2nd line, 调用的时ellipse的render函数，而非shape的render

多态(Poly-morphism)



**upcast** & **dynamic binding**

**upcast:将子类的对象视为基类的对象**

# # 24 多态的实现

void render(Shape\* p){

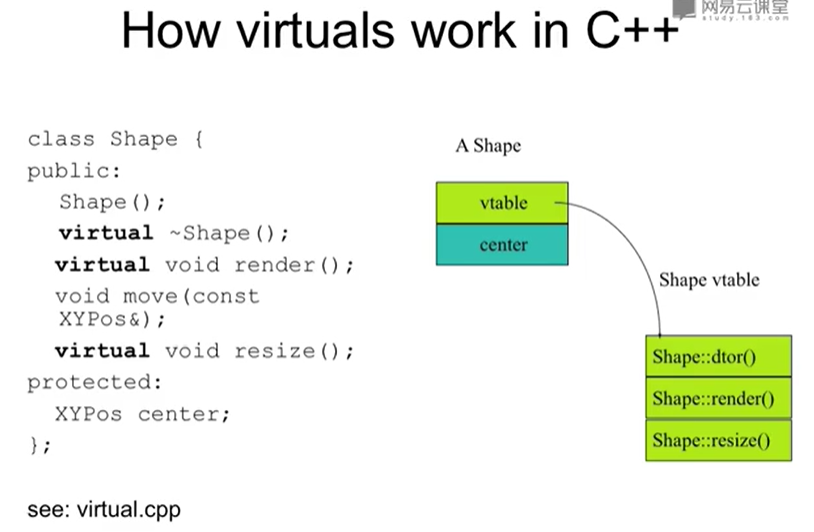
p->render();

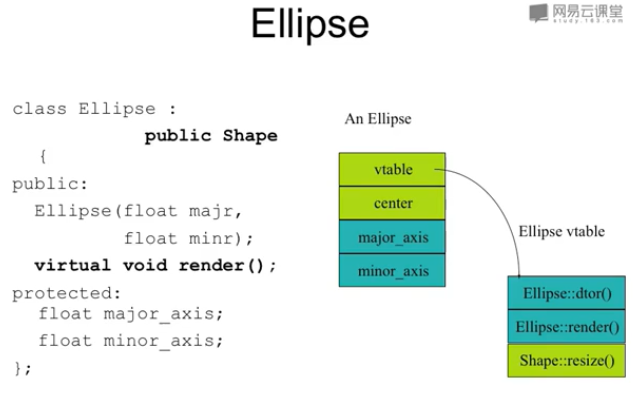
}

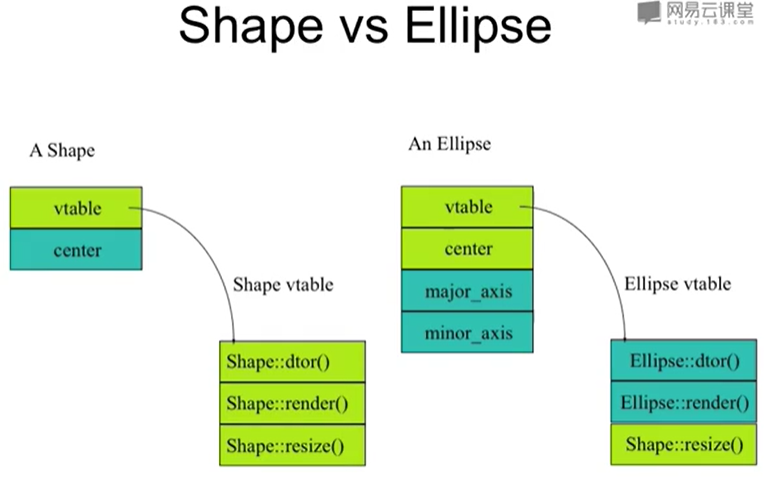
这里的p具有多态性（多态对象）。

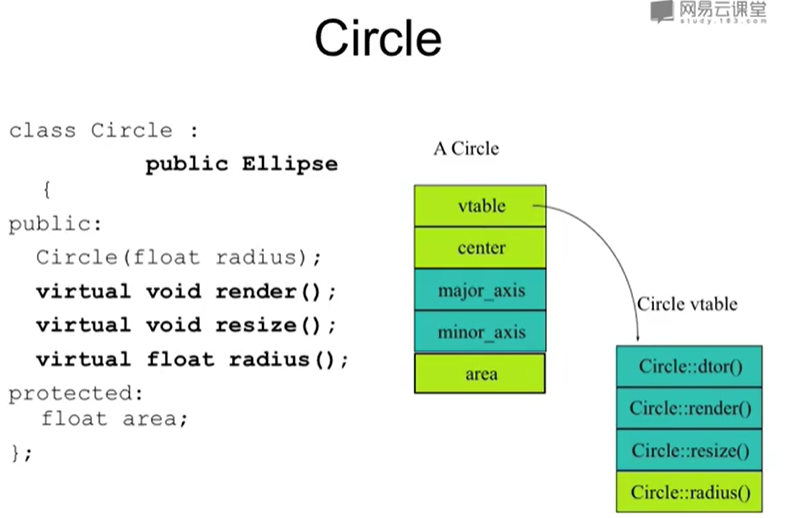
任何一个类，只要有虚函数，就会比正常的类对象大一点！所有**有virtual的**类 首部都会加一个指针，用以指向虚函数表。

在赋值操作“=”中，虚函数的指针不会被赋值（被改变）。



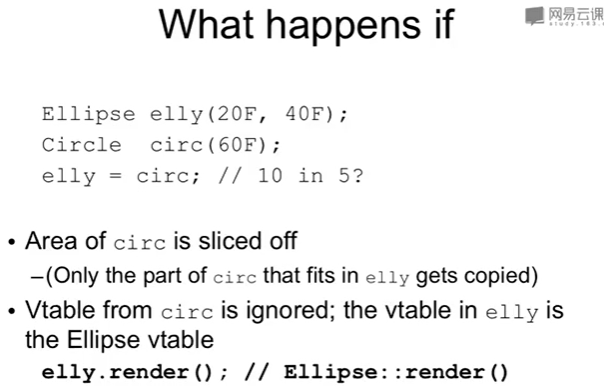


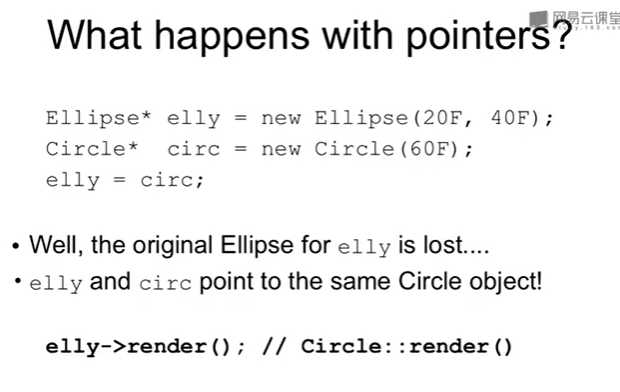




能够把**子类对象赋值给父类对象**吗？

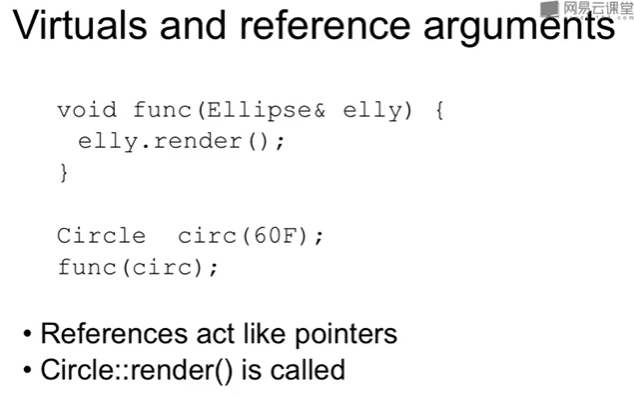
vtable这个指针在赋值过程中不会被赋值，也即等号左侧的vtable不会改变。

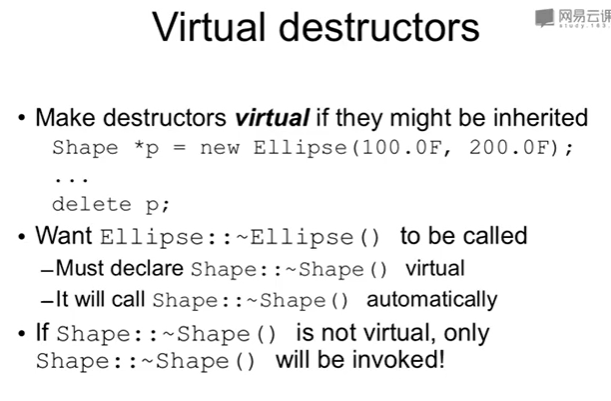




这是什么意思？子类指针赋给基类指针。base\_ptr = dereived\_ptr; base\_ptr->func 则会调用子类的func而非基类的func函数。【表现出多态？】

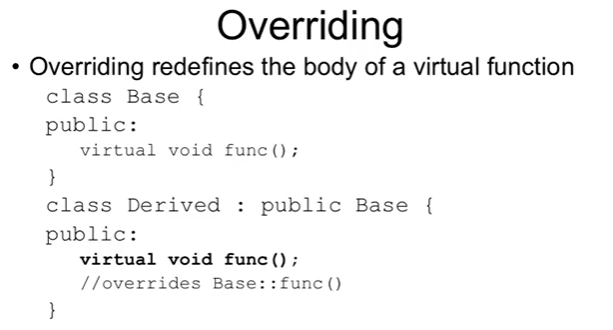
**但是如果直接把子类对象赋值给基类对象，那么指向vtable的指针不会进行赋值（被忽略）。因此，以基类对象调用函数，那么还是会调用基类对应的函数。**

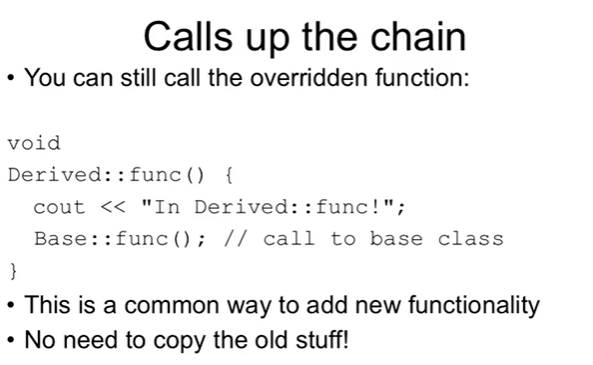




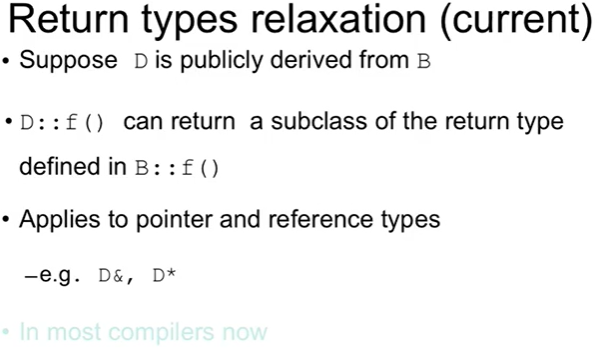
类里面有一个函数是虚构的，则其**析构函数就必须是虚构**的，否则就会出现问题。CPP默认是静态绑定，其他OOP是动态绑定。

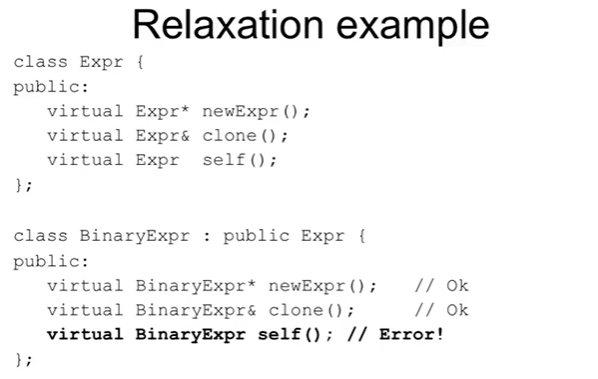
#include <iostream>



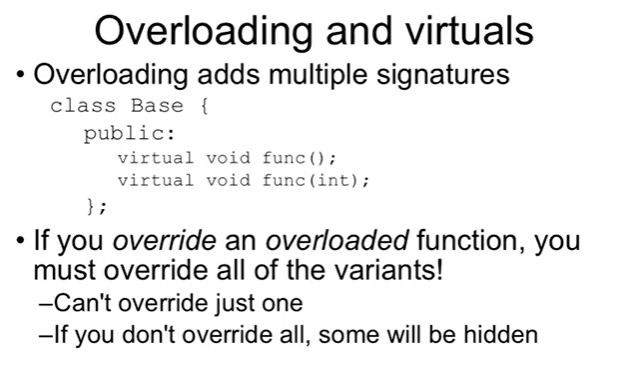
****

**返回值类型的放松relaxation**

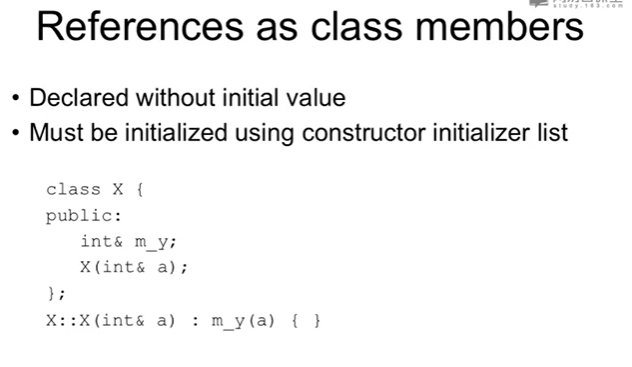




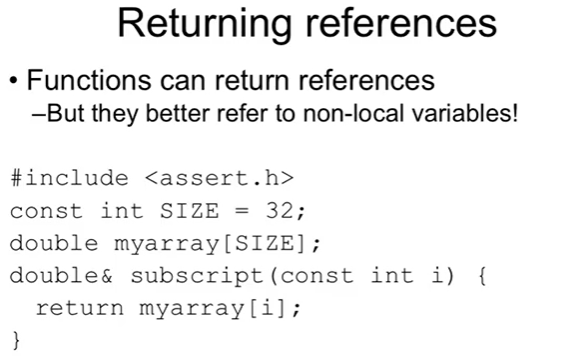
只有通过**指针／引用**才能构成Upcast关系．



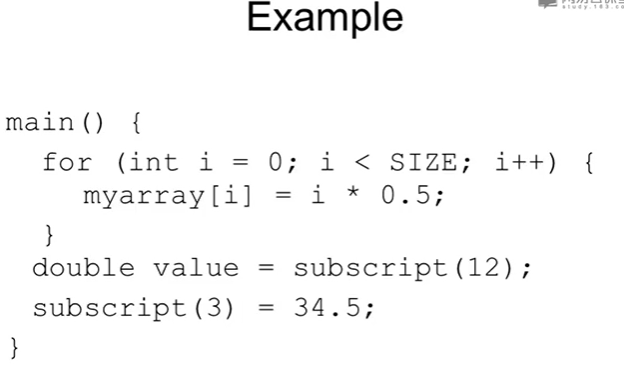
# # 25引用再研究

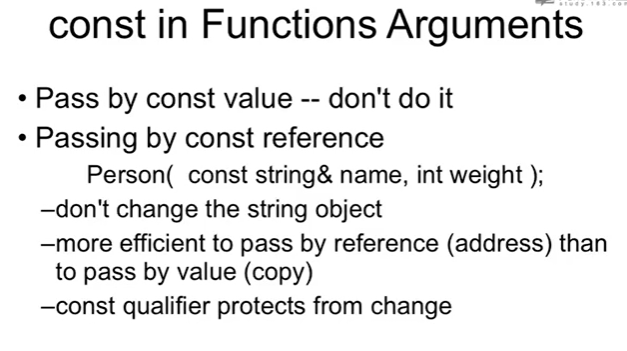


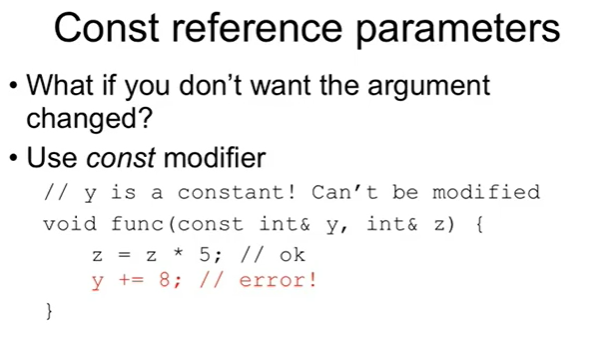
**必须在初始化列表中来对类中数据引用进行初始化**

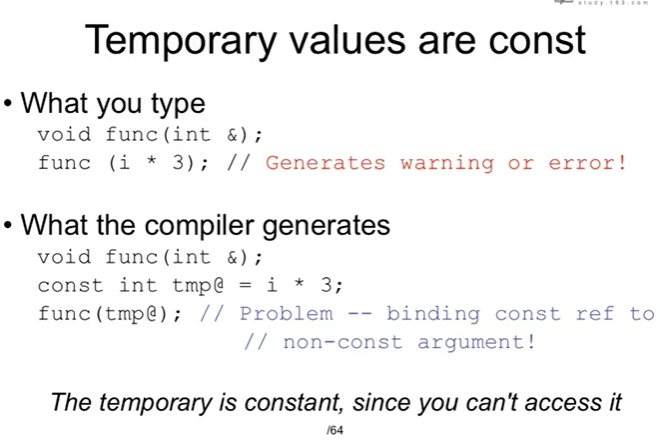


**函数不能返回局部变量的 地址(指针和引用).返回的是一个引用reference**



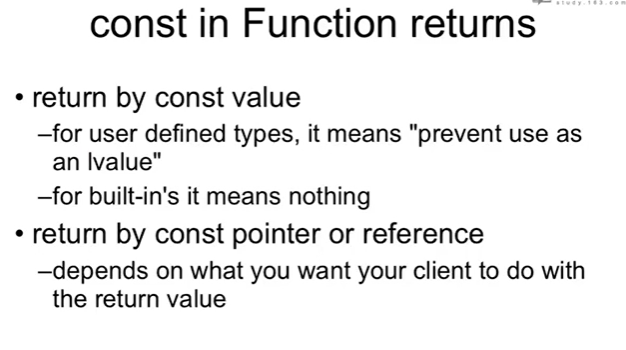






void func(const int &)； // this ok

临时变量是**常**的



const value v.s. const pointer(reference)

* 返回常量const value
  + 对于用户定义的类型，则不能够作为左值
  + 内建类型，则没啥用meaningless
* 返回常指针（引用）