## 实验4 Object-Oriented Programming: Inheritance

## 学号: 09017423 姓名：杨彬

### 实验目的

Objectives

This lab was designed for learning inheritance mechanism to support OO programming in C++:

To create classes by inheriting from exiting classes.

The notations of base classes and derived classes and the relationships between them.

The order in which objects were constructed and destructed in inheritance hierarchies.

The initial in heritance

The difference between public,protected and private member access specifier.

The difference between public,protected and private inheritance.

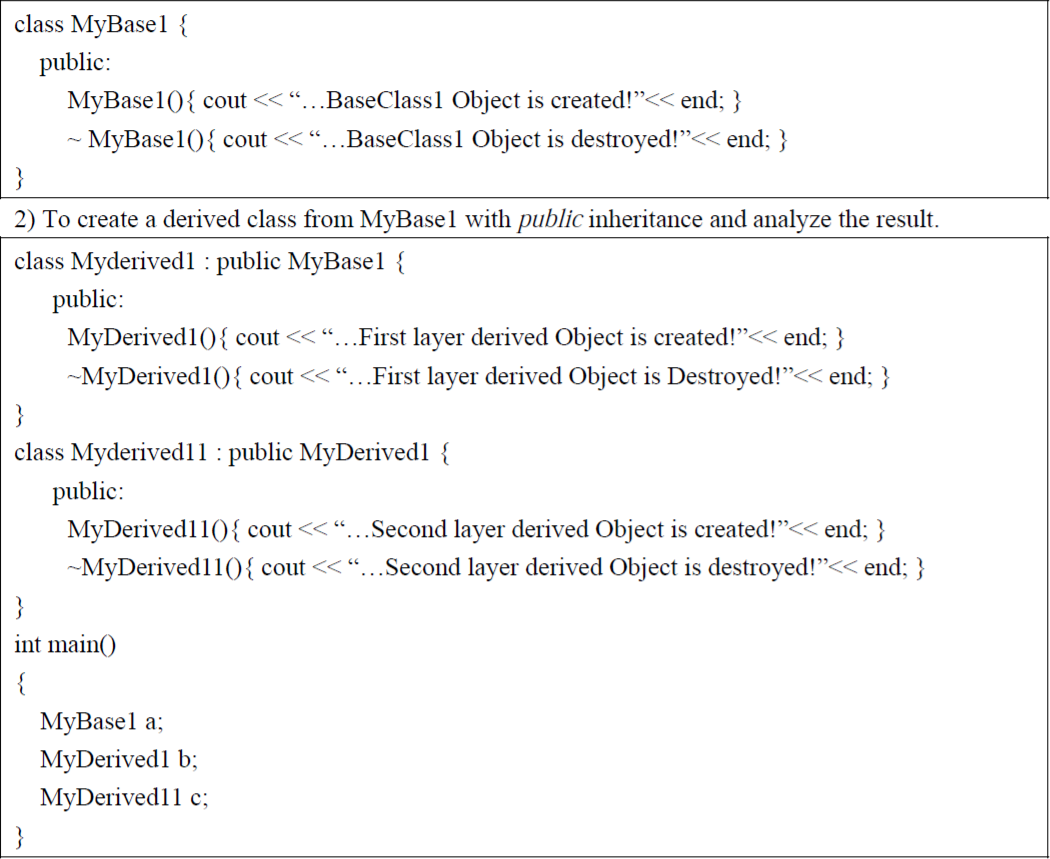
The inheritance,add and hide of class member functions.

The translation between base class and derived classs.

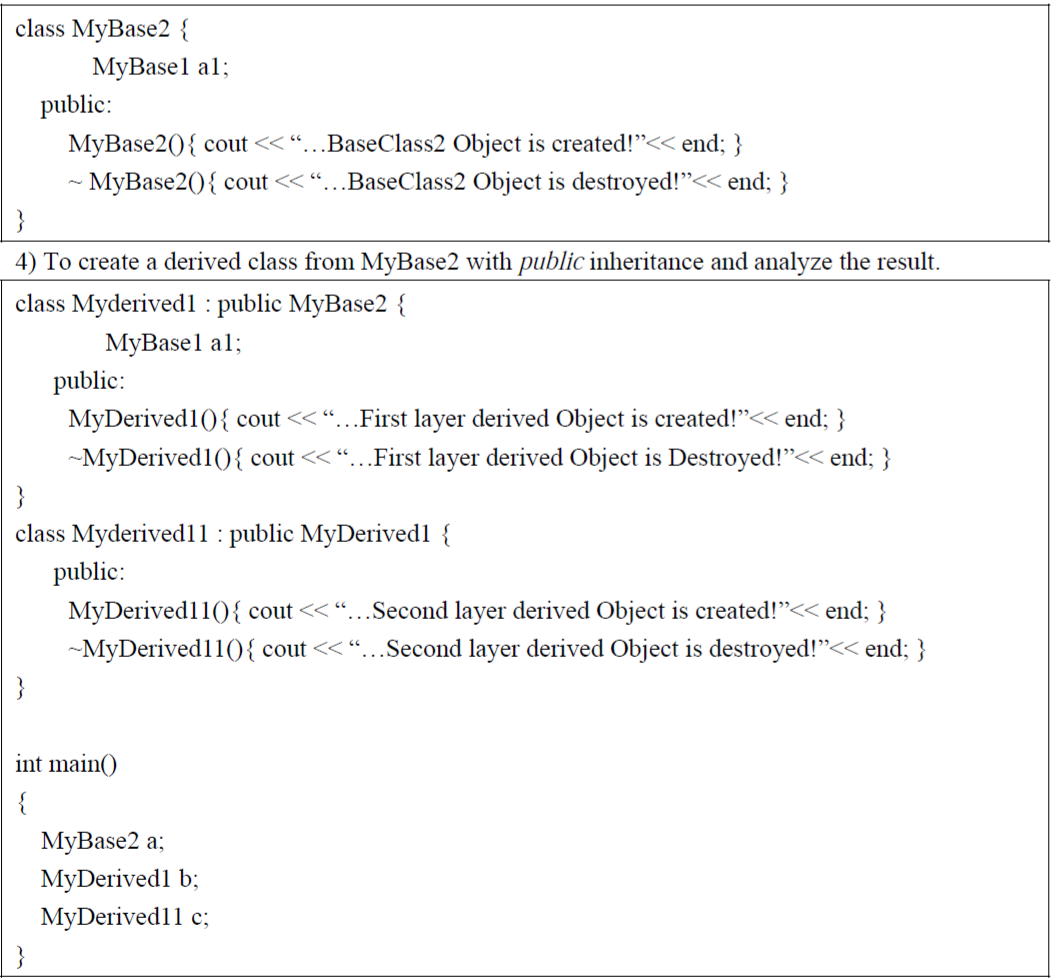
### 实验内容

Exp1: The construction and destroying of objects in heritance

1.1: To create a base class as following:

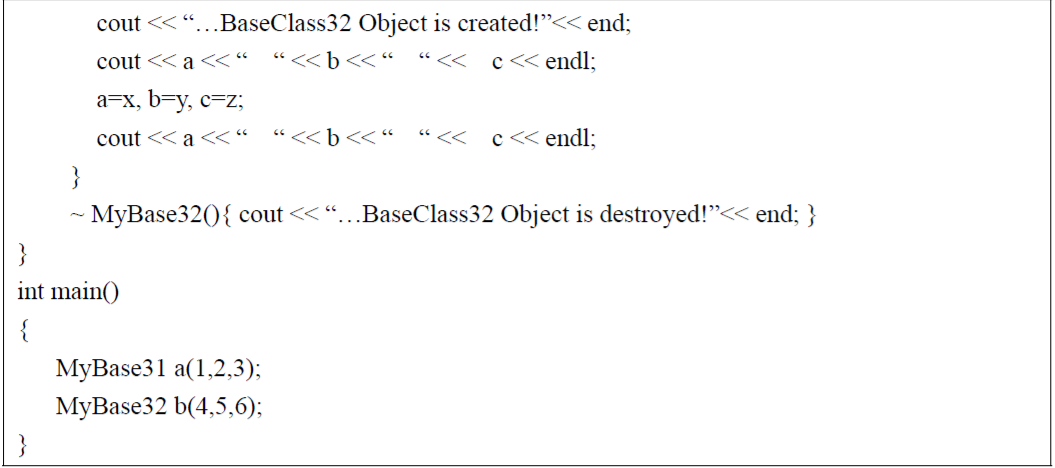
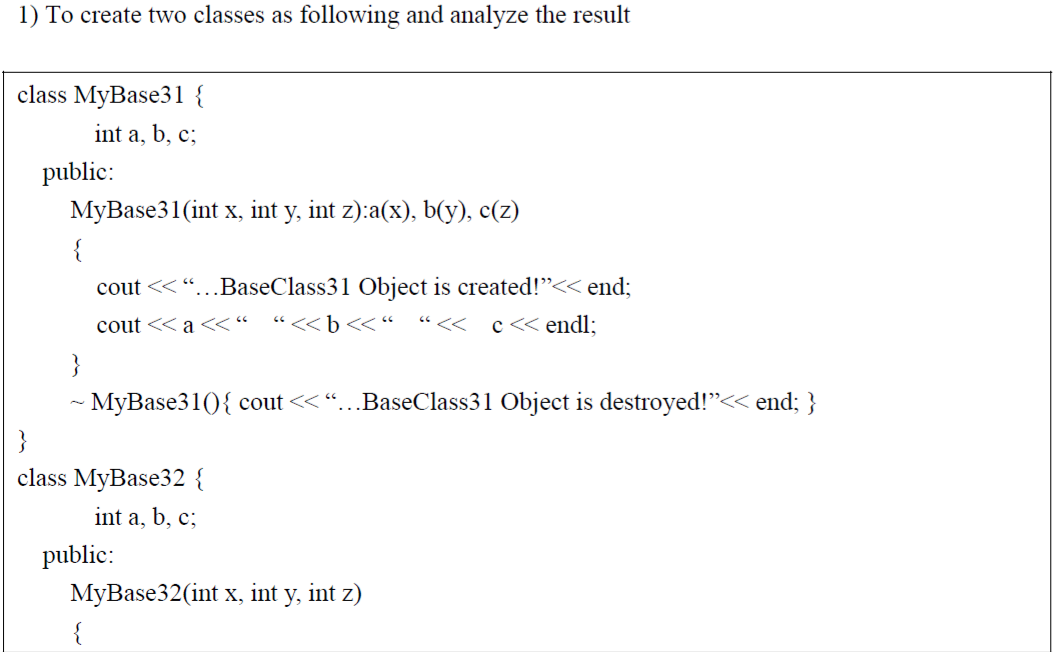


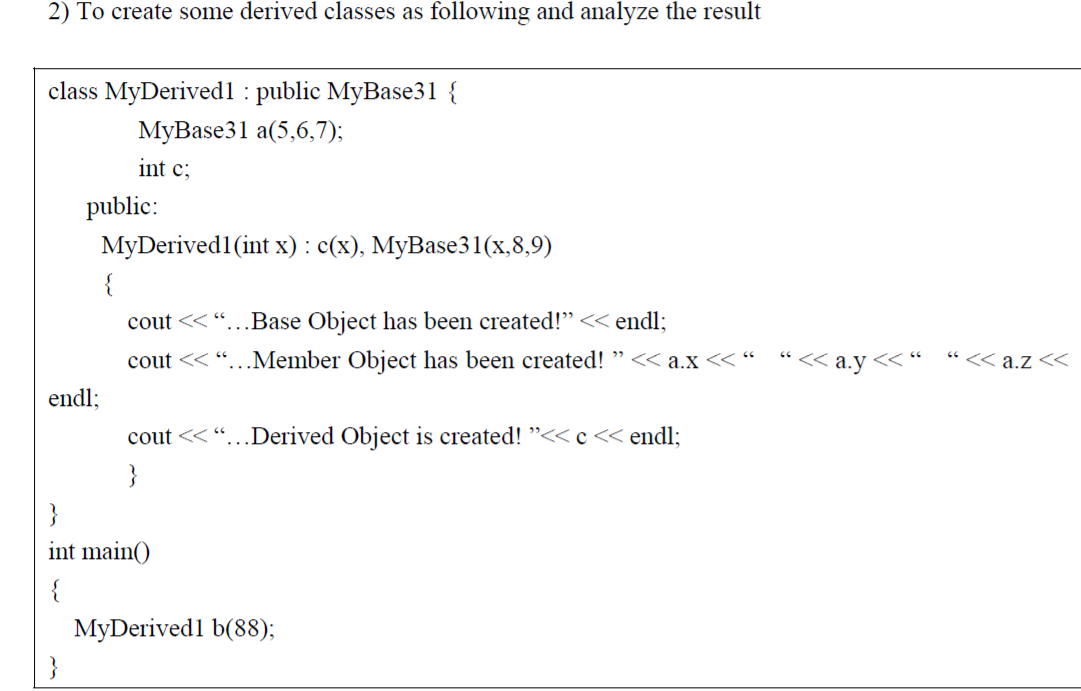
1.2: To create a base class as following



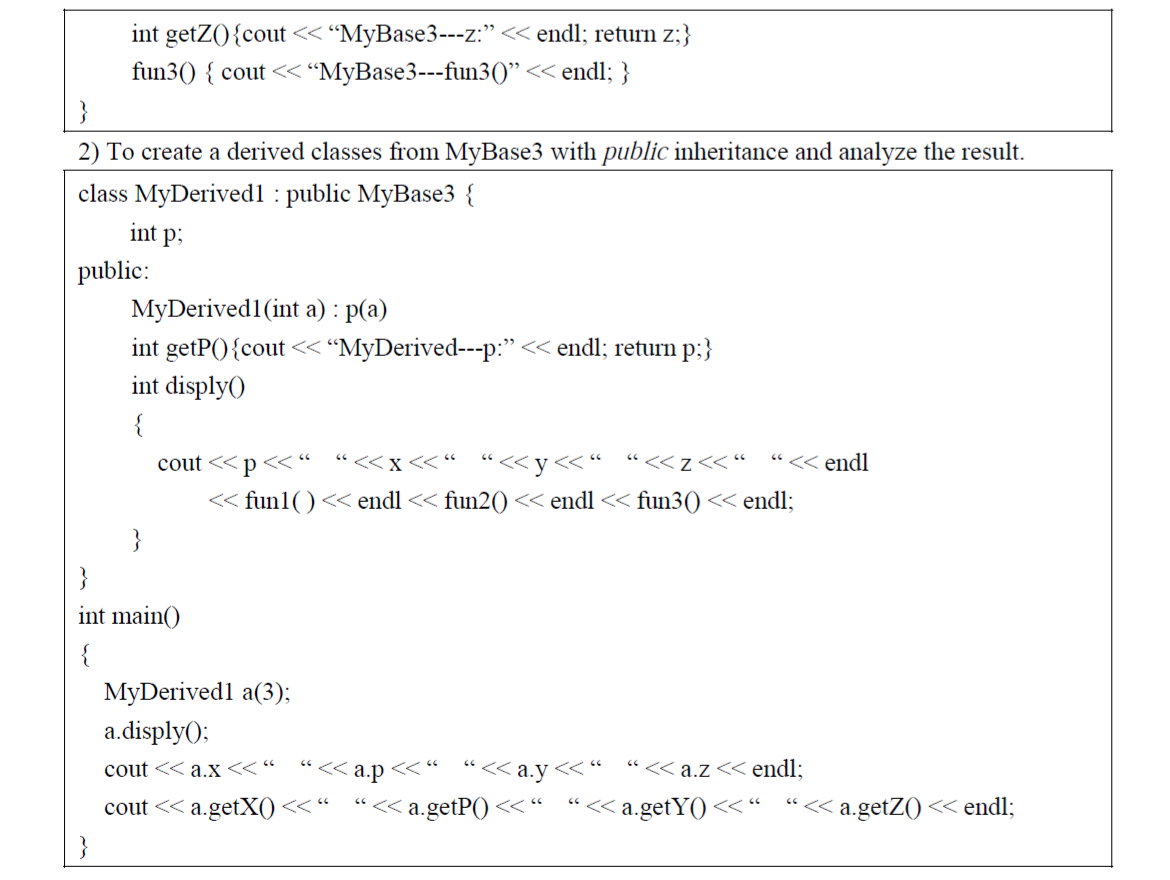
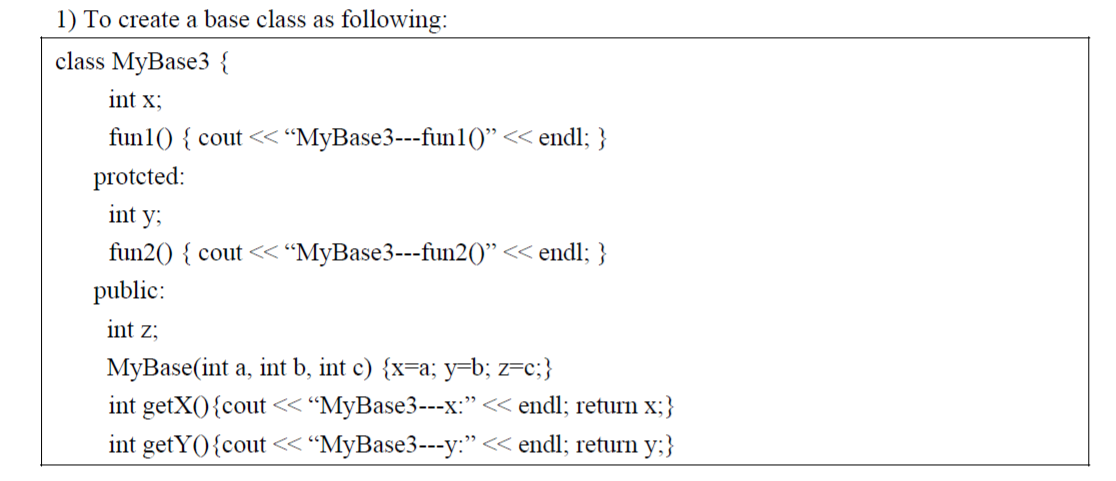
Exp2: The initial of objects in heritance

2.1

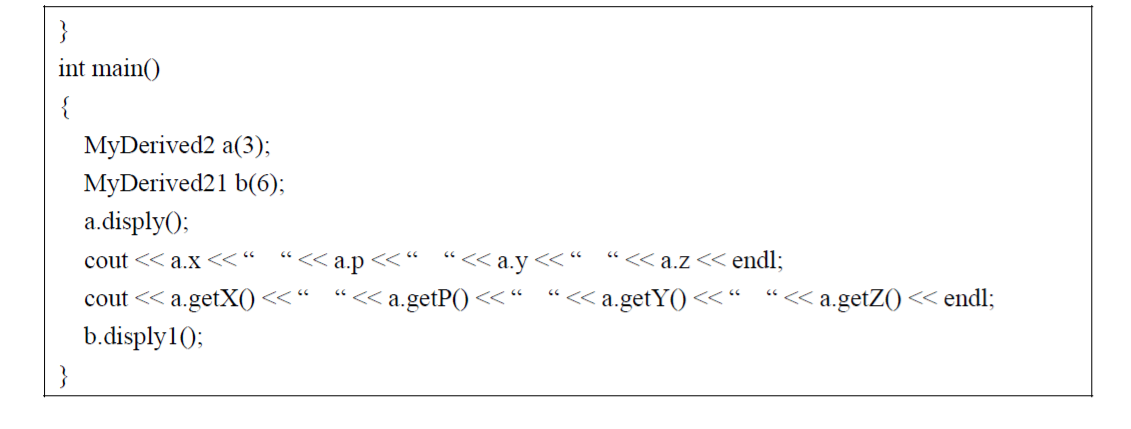
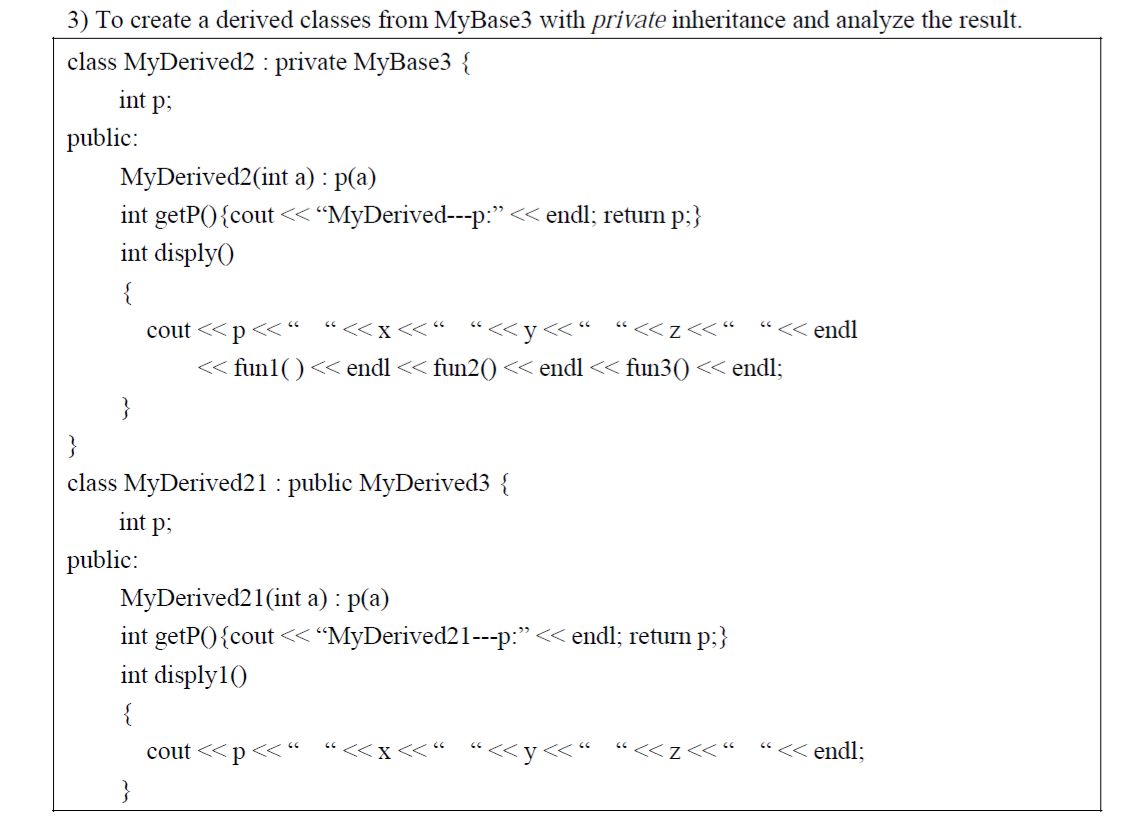


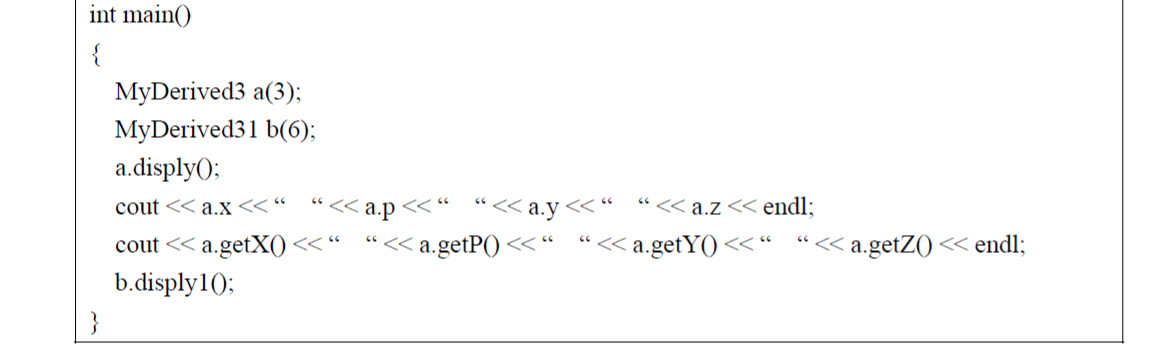
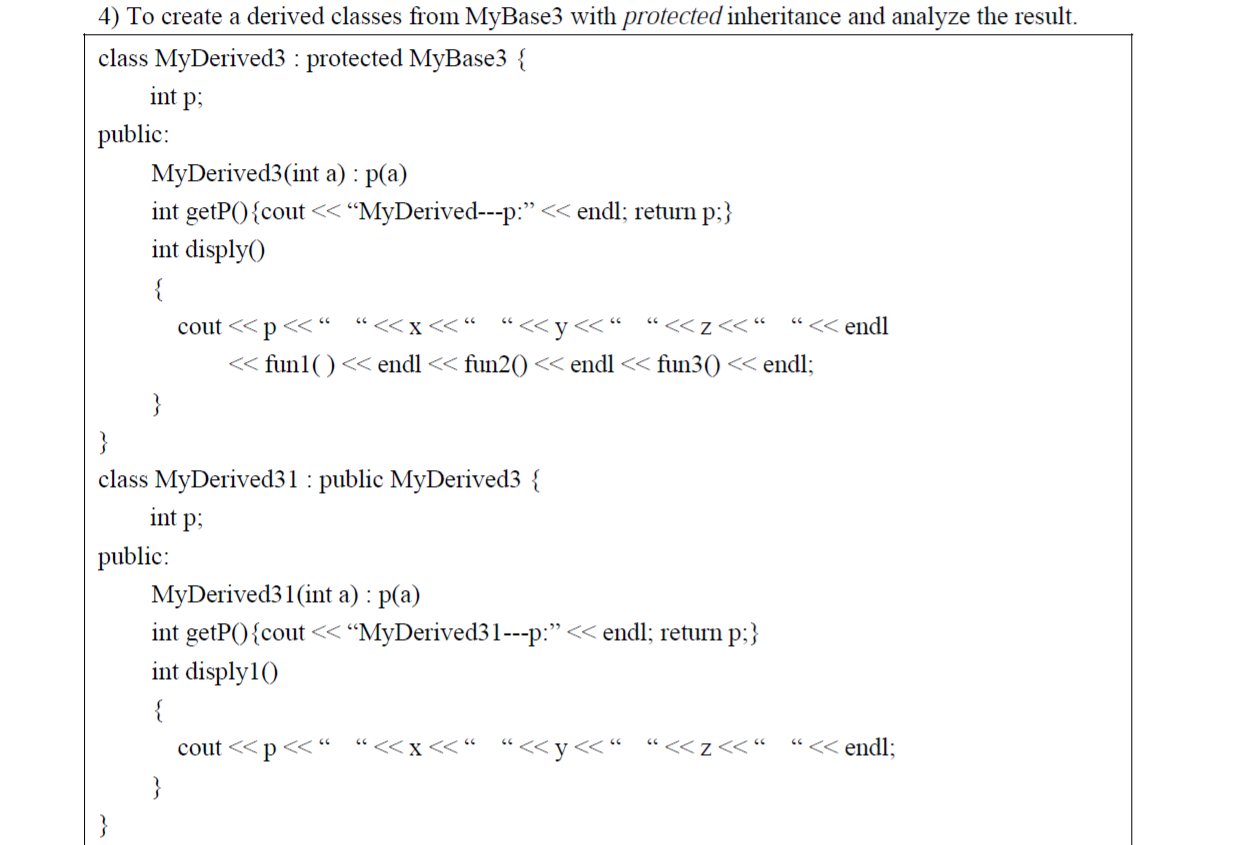
2.2

Exp3: The access properties in inheritance

3.1: 

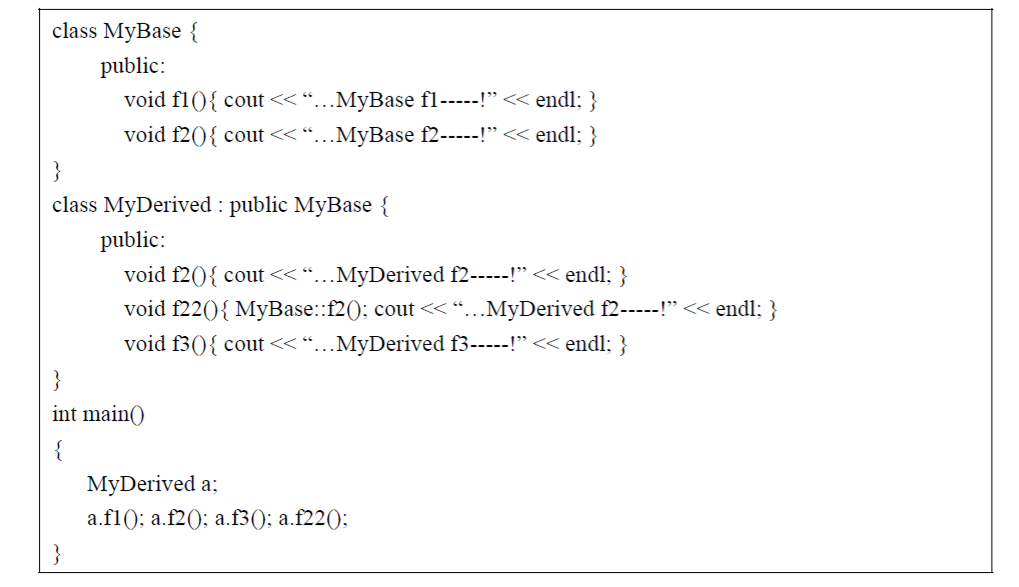
3.2



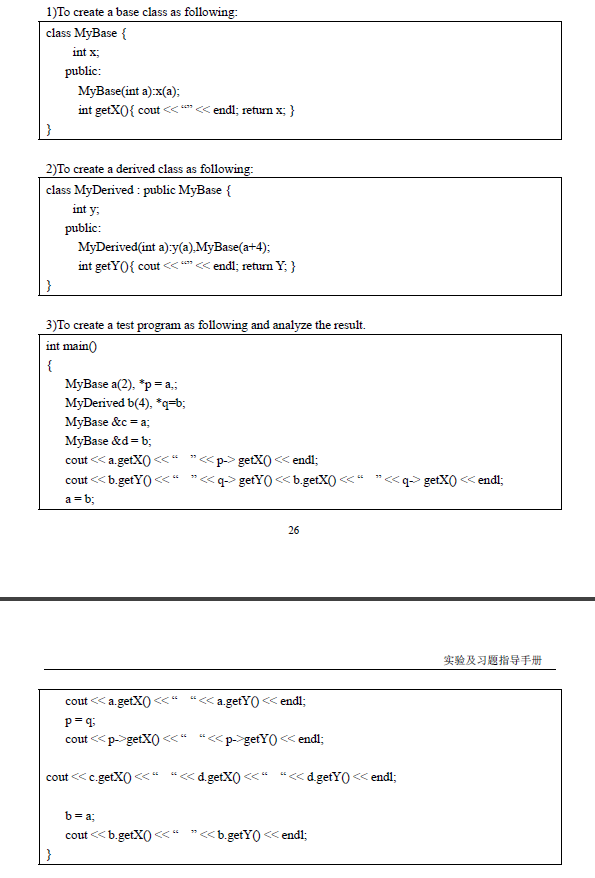
3.3

3.4

To analyze the result



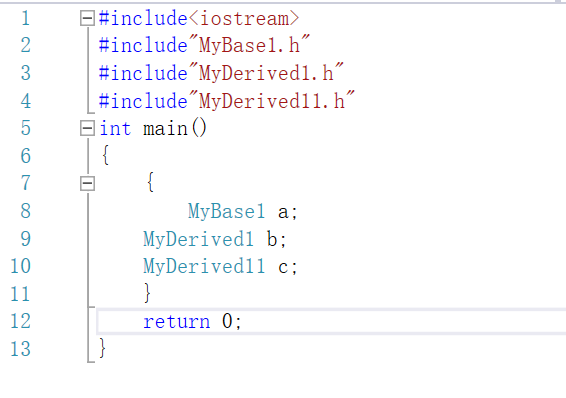
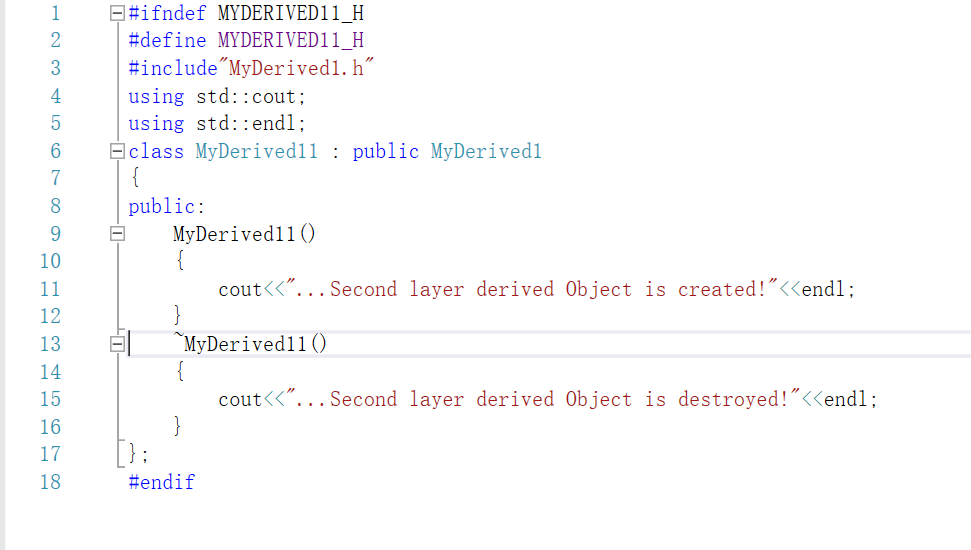
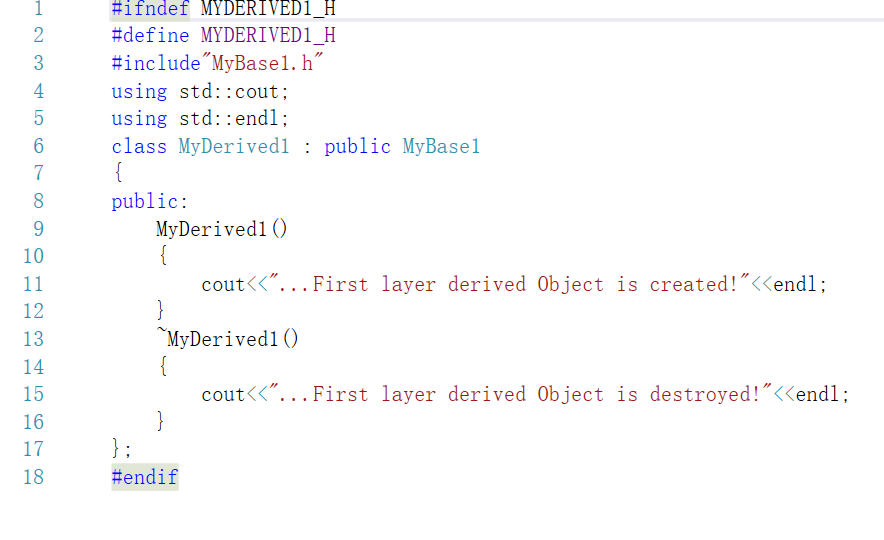
Exp4: The translation between base class and derived class.

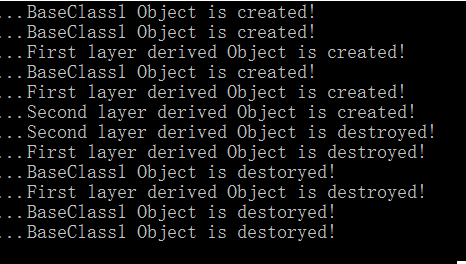


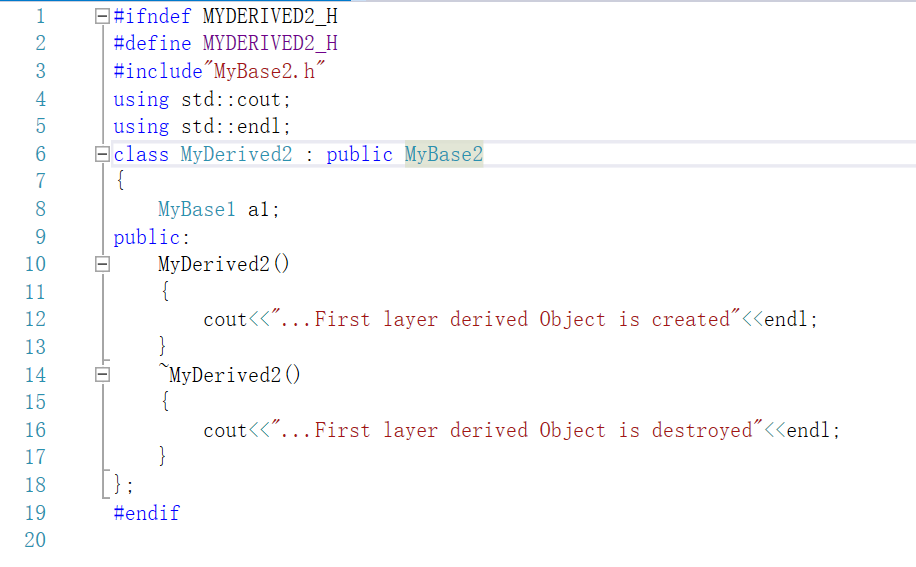
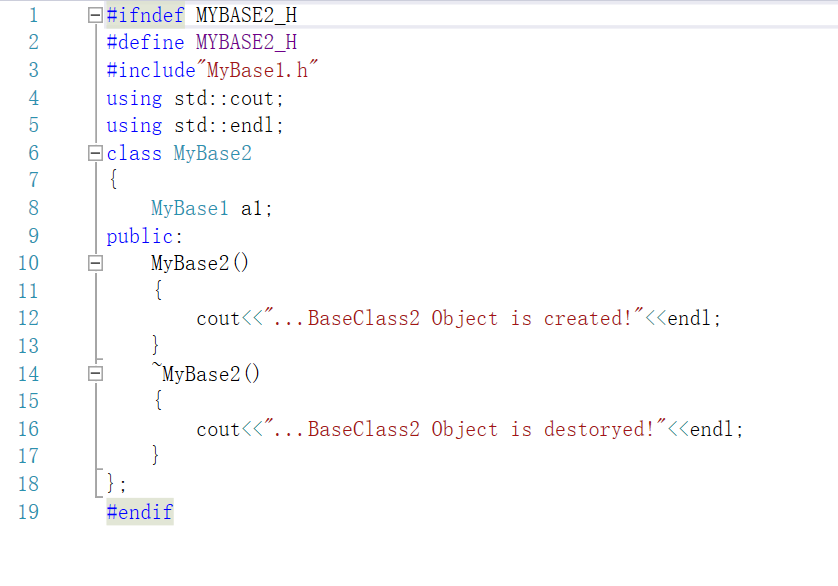
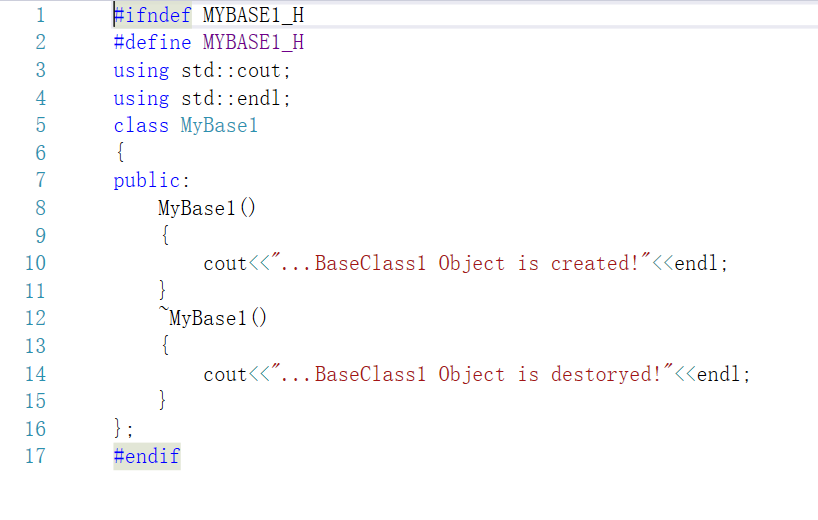
### 实验代码及结果：

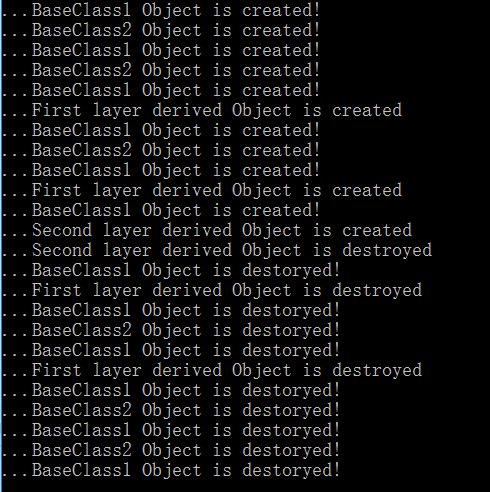
Exp1:

1.1:



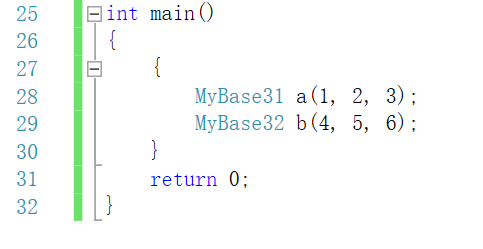
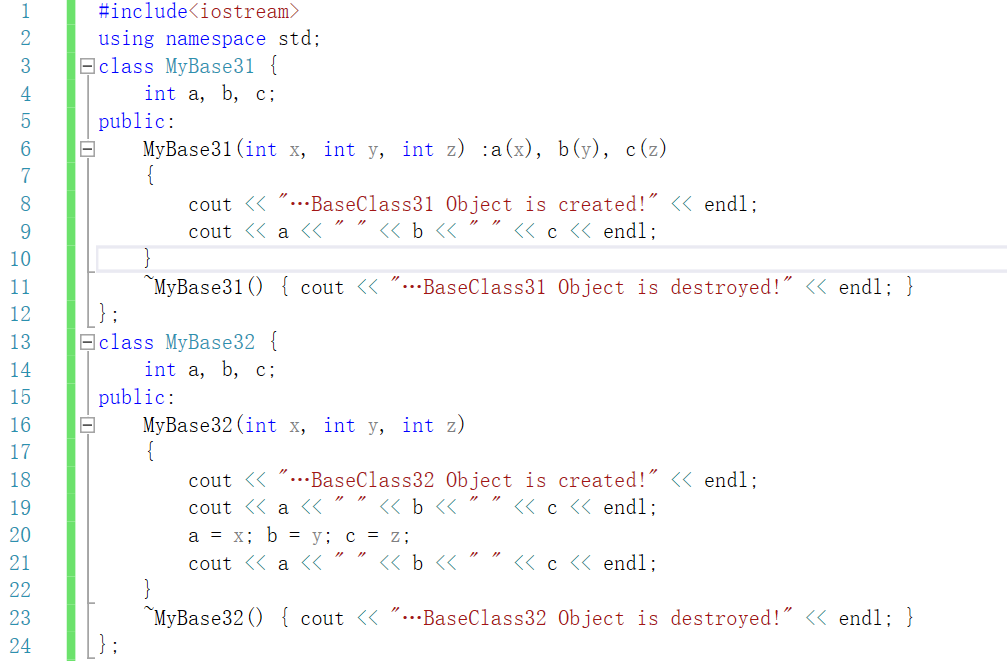
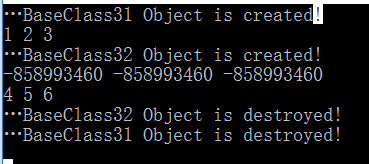


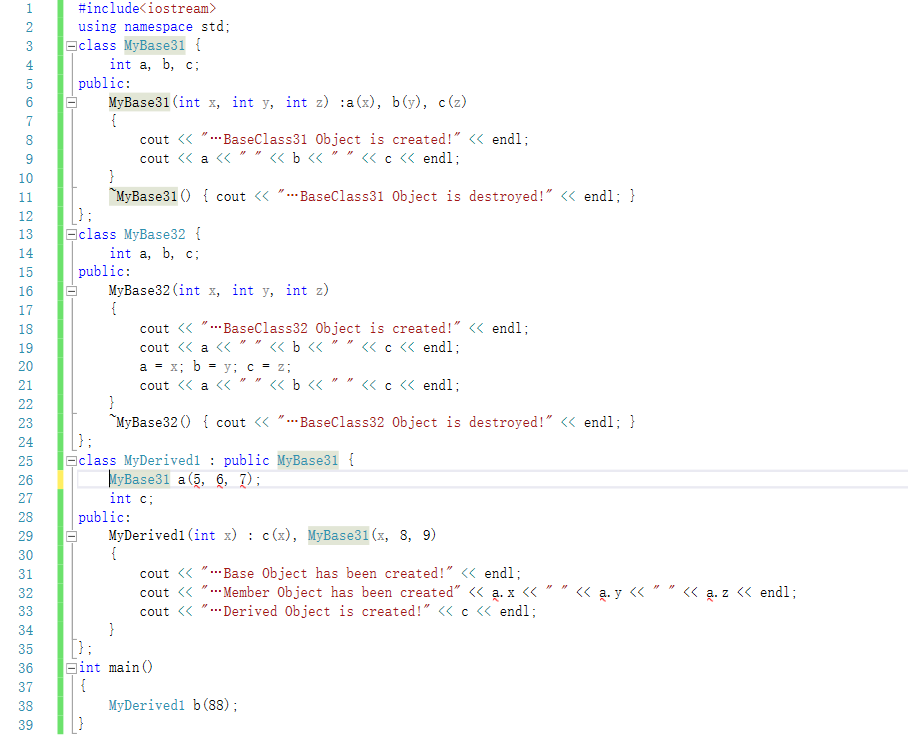
1.2: 



Exp2:

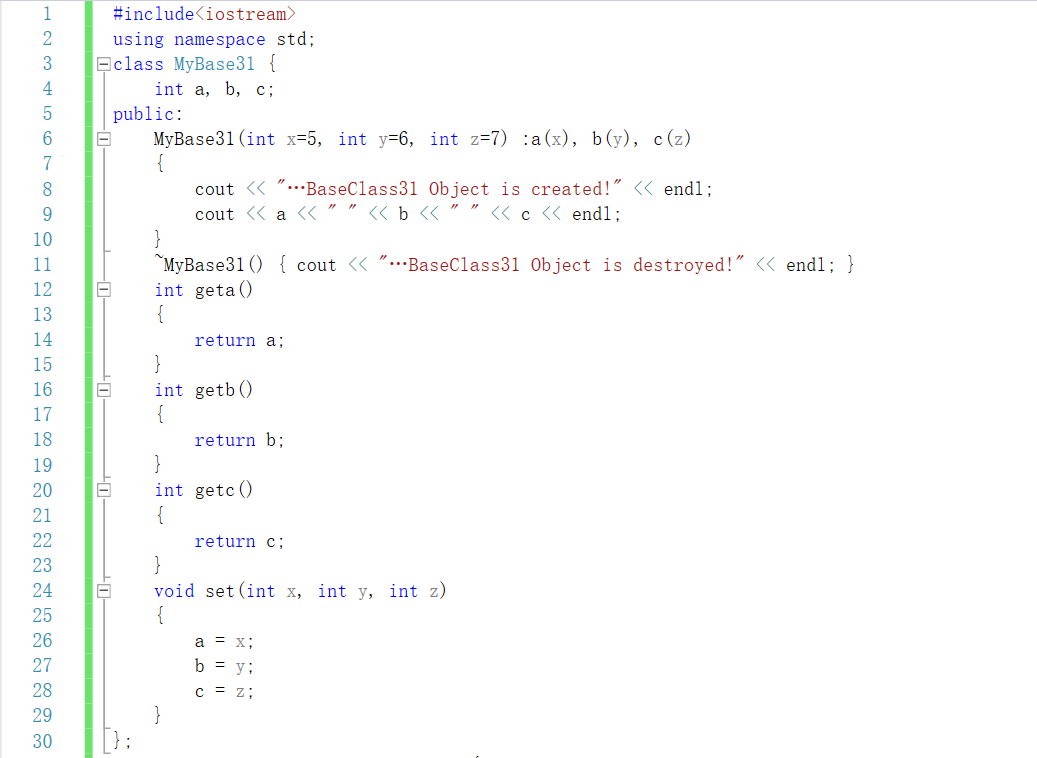
2.1

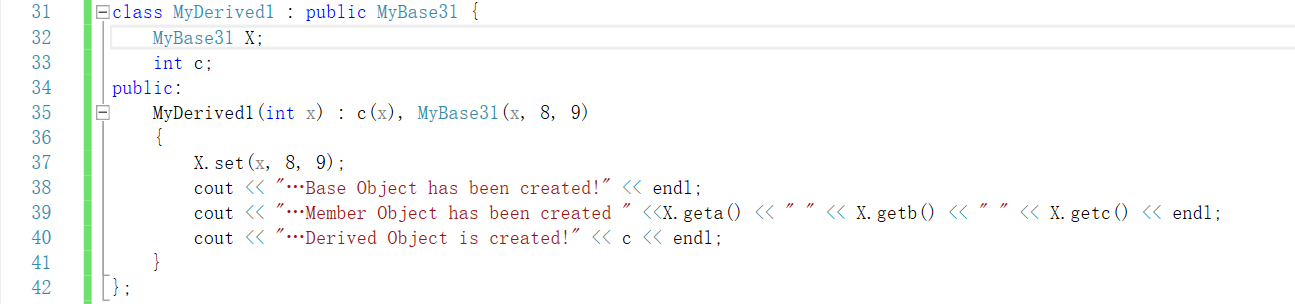


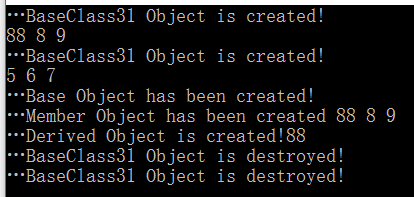
2．2 原始代码

代码明显有错 1类成员中除了静态常量，其它的成员不能在类中进行初始化2类的私有成员不能直接访问

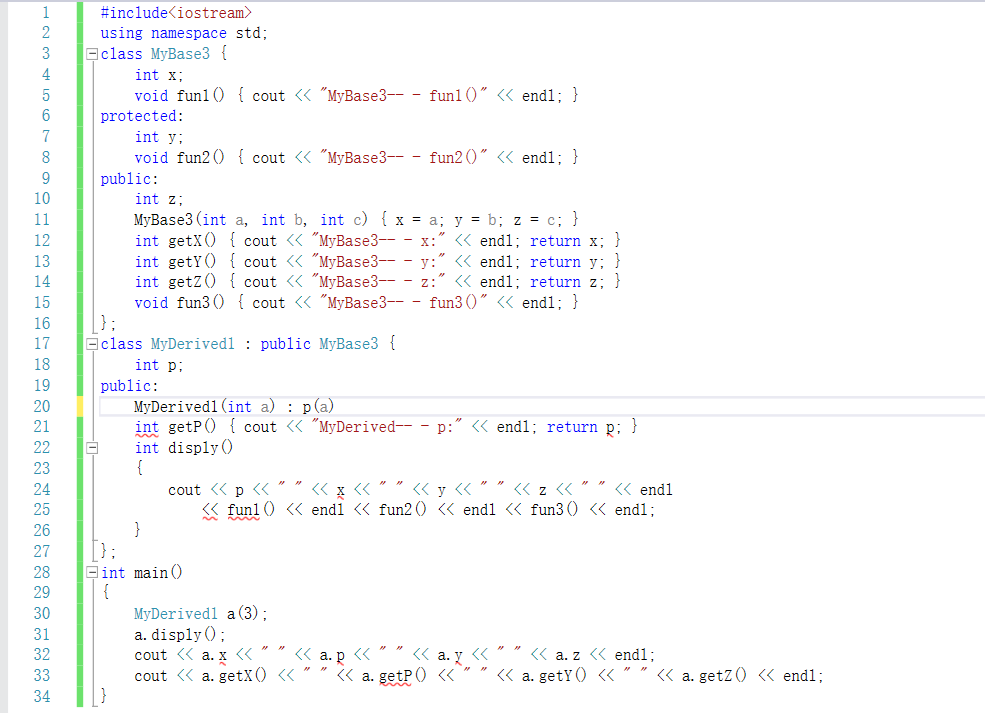
故更改为





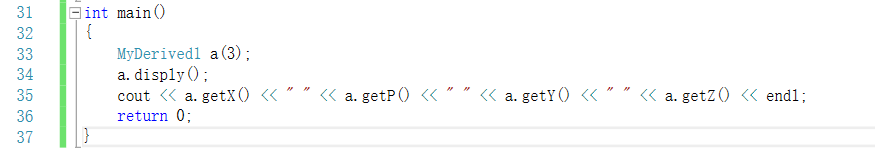
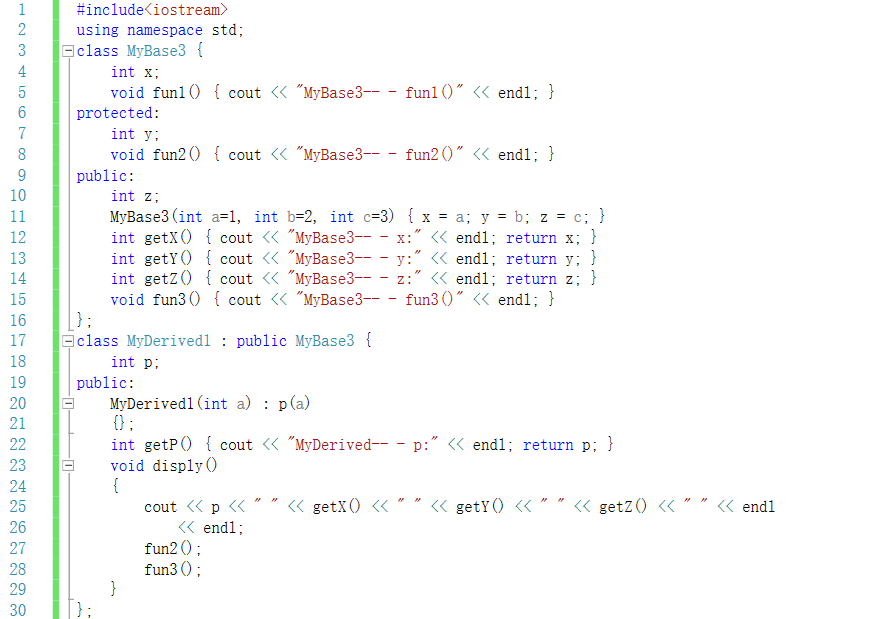


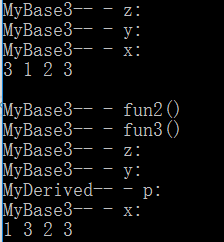
Exp3

3.1

这段原始代码存在很多严重错误，比如直接访问父类的私有成员，比如初始化问题，无法运行

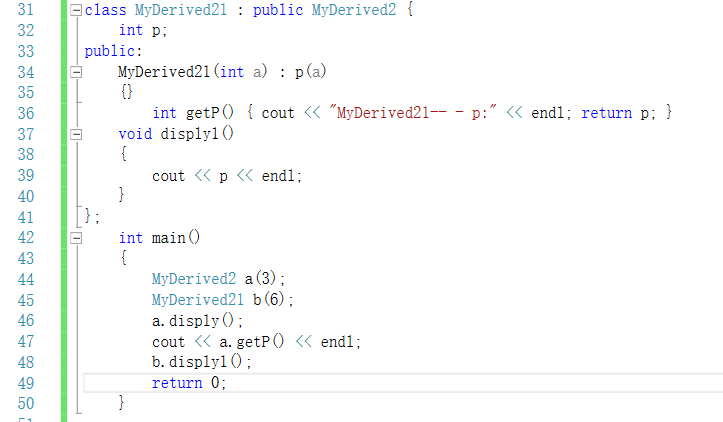
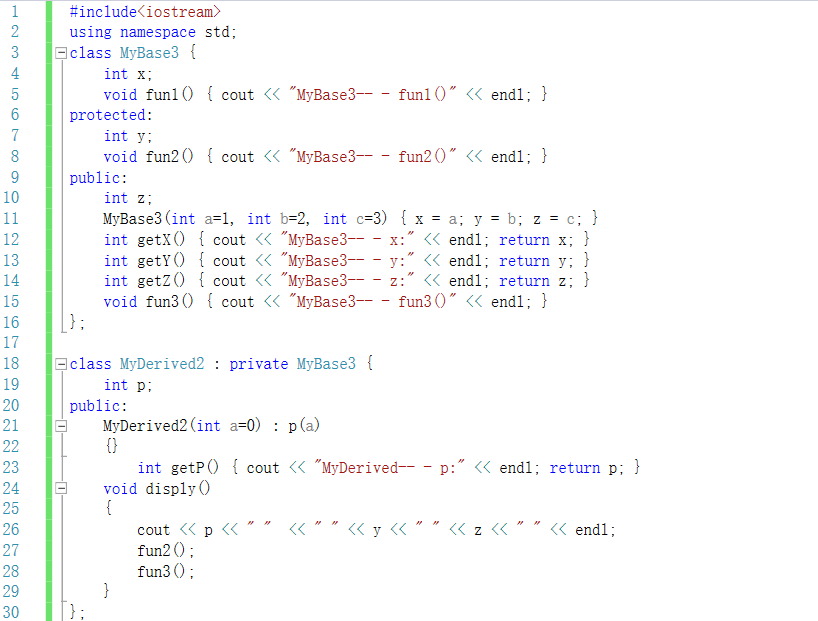
故将代码改为



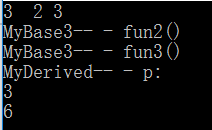


3.2原实验代码存在严重错误

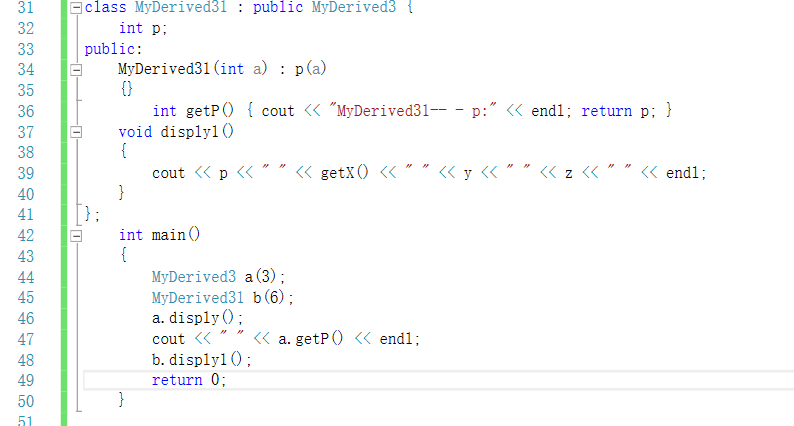
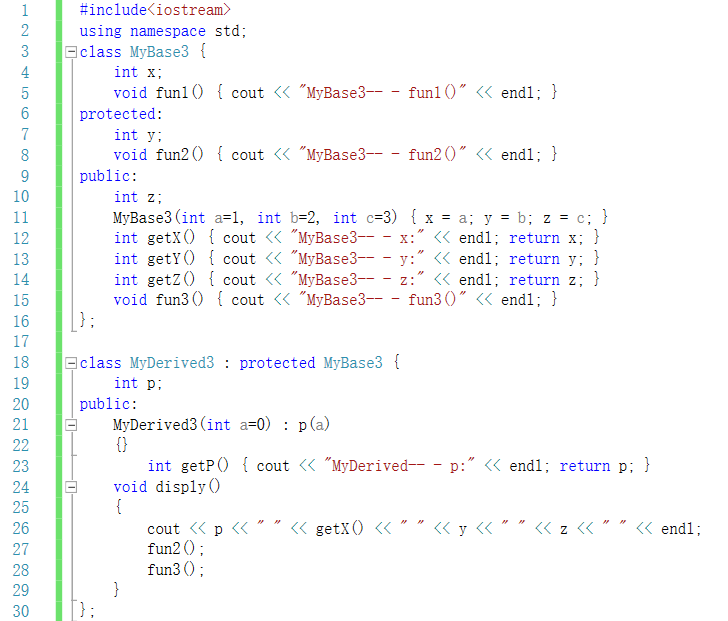
更改为

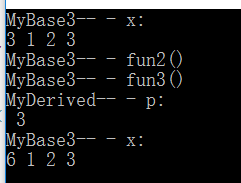


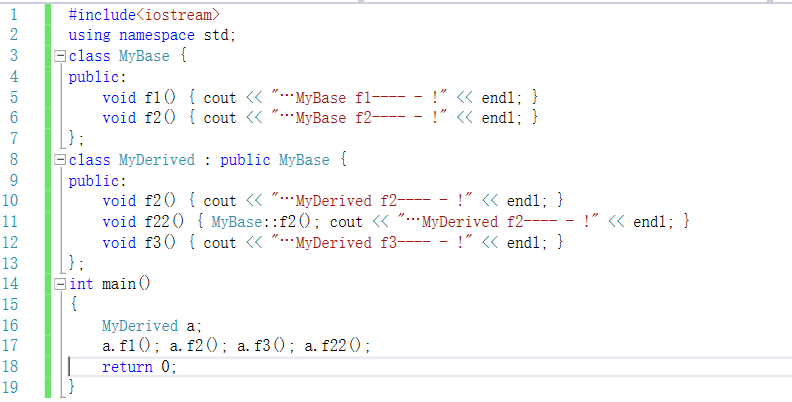
因为是私有继承，所以不能直接访问fun1 x ,getY(),getX(),getZ()

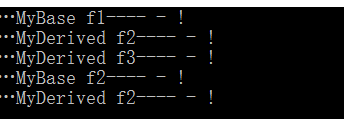


3.3: 因为在继承过程中存在的访问权限的问题，源代码存在很多错误

更改为

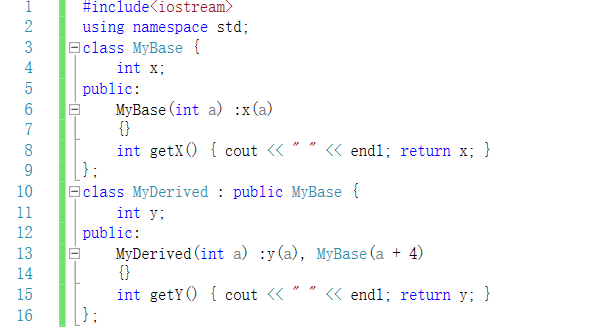
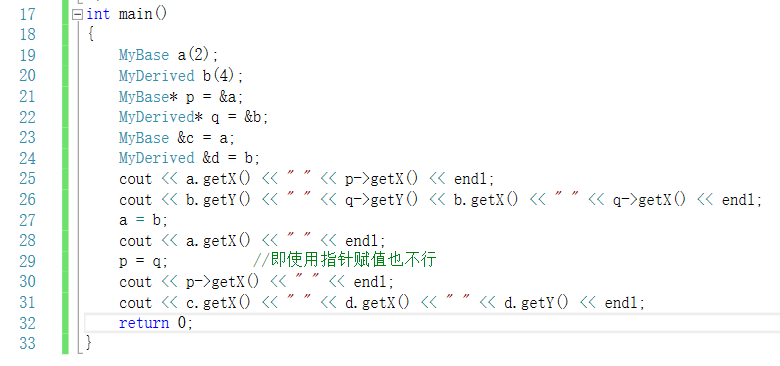


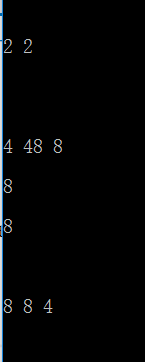
3.4



Exp4:

本实验的源代码存在错误更改为



### 遇到的问题及解决过程

Exp1:nothing

Exp2:nothing

Exp3:nothing

Exp4:nothing

### 实验体会

Exp1:这个实验充分地说明了在类继承和类包含当中的生和死的问题1类的生的问题是尊老爱幼，并且是先尊老、后爱幼。具体来说当既有继承的类、又有包含的类的时候先调用继承的类的构造函数、再调用包含的类的构造函数。通过这个实验将类中生和死的问题总结如下

1 类继承和类包含中的尊老爱幼原则

2 局部变量（在离它最近的花括号内、只要有花括号）遇到了调用构造函数，一离开作用域马上调用析构函数。

3 全局变量，一进入main函数调用其构造函数，离开main函数（程序结束之后）调用其析构函数

4static变量，遇到的时候调用其构造函数，程序结束（离开main）函数的时候调用其析构函数

5函数按值传递是的参数（没做一次按值传递，需要调用其复制构造函数，复制结束后然后调用其析构函数）

6函数中的返回值同5； 从5 和6来看一个类往往需要复制构造函数（特别是涉及到深复制的时候）没有复制构造函数，其在传递参数以及返回值的时候会出现严重的错误）

Exp2:

这个实验首先在构造函数中用两种不同的方法对成员变量进行初始化，1初始化参数列表2在函数体中进行初始化，实验结果证明先执行初始化参数列表然后在执行函数体内的语句。

在初始化子类的时候，还应该显式地调用父类的构造函数这样才能进行完整的初始化。并且用初始化参数列表是一个很好的选择。

Exp3:

总结一下，在类中

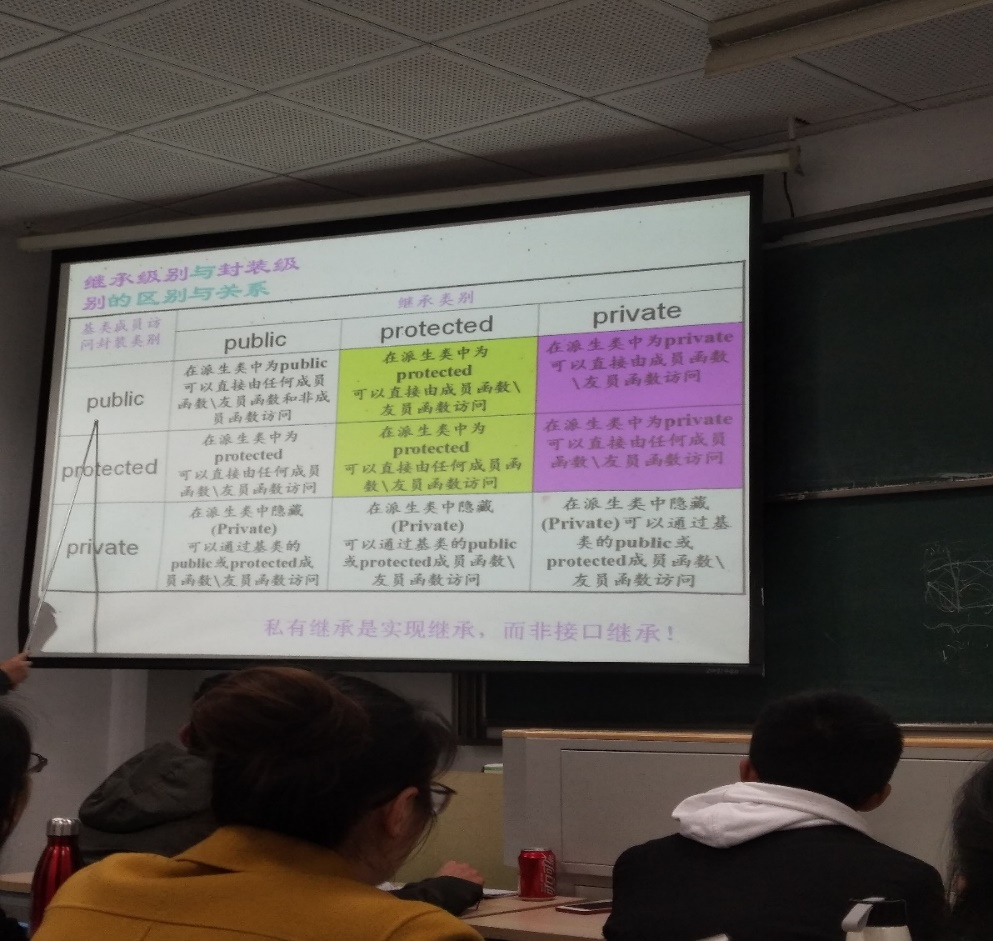
protected只能由类自身访问，可以被子类访问

基类中的private： 只能由类自身访问，不能被子类直接访问

派生类中的private:只能由类自身访问。

public:可以由外部直接访问也可以被子类直接访问

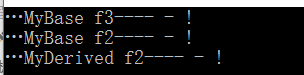
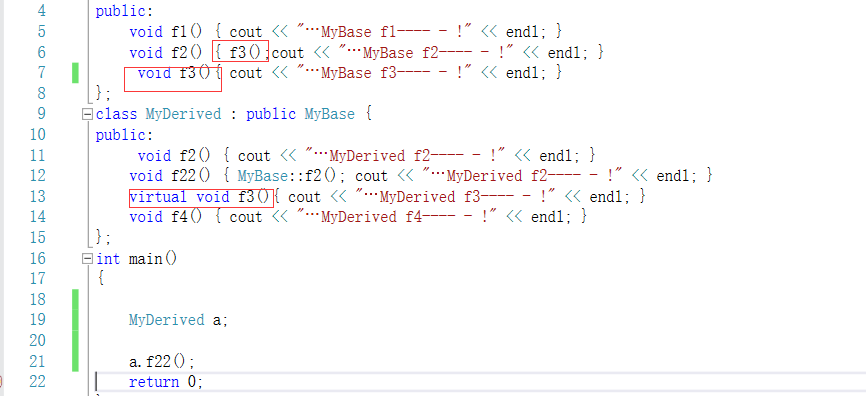
类的继承的类型对在派生类中封装类型的影响

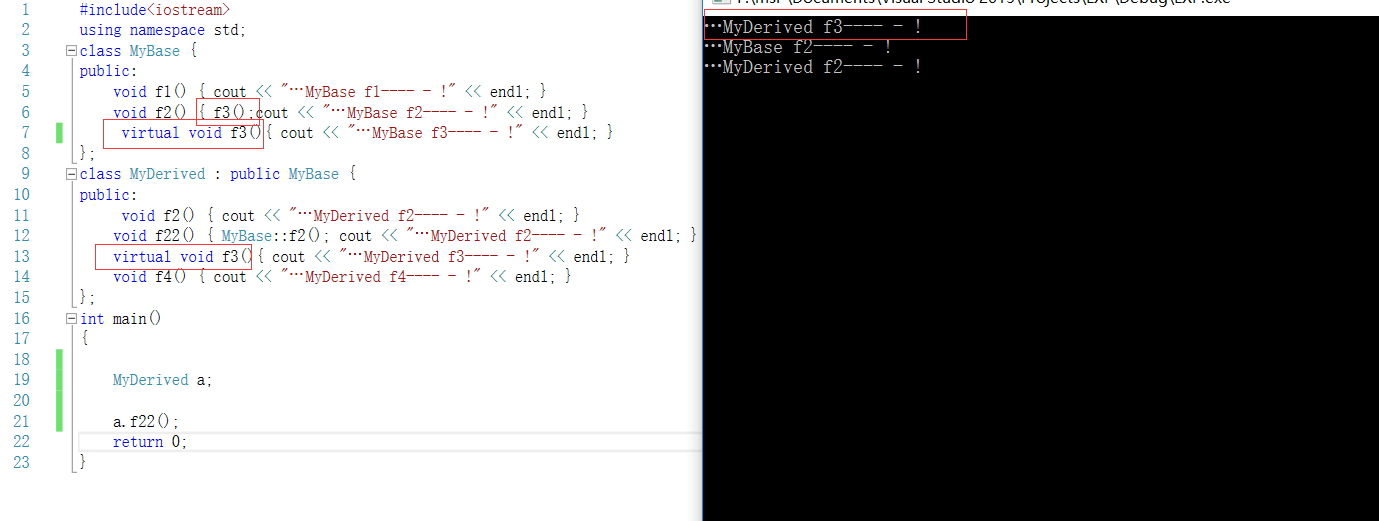


所以可以用protected 和private 的继承方式来控制“要继承几代”

而3.4这个实验说明当父类和子类有相同的函数名称的时候，子类的函数会盖过父类的函数，如果要使用父类的函数应该要加上类作用符号。（同时还应该注意虚函数的使用）

同时值得注意的是，在函数覆盖中除了虚函数之外，如果存在嵌套现象是不覆盖的，（而且虚函数必须满足子类和父类的函数都是虚函数）

比如



Exp4: 本实验说明了基类与子类之间相互赋值的特点

1基类可以用子类赋值，但是子类不能用基类赋值；

2将子类赋值给基类之后，子类中的继承部分的数据赋值过去，但是赋值之后的基类不能访问子类的数据（即使是指针也如此）换句话说将子类赋值给基类的时候会损失子类的数据。

但是在多态中，采用指针或者是引用的形式访问的是虚函数。