# EX1

**main1.cpp**

#include <iostream>  
class MyBase1  
{  
public:  
 MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is created!" << std::endl;  
 }  
 ~MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is destroyed!" << std::endl;  
 }  
};  
  
class MyDerived1 : public MyBase1  
{  
public:  
 MyDerived1()  
 {  
 std::cout << "…First layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived1()  
 {  
 std::cout << "…First layer derived Object is Destroyed!" << std::endl;  
 }  
};  
class MyDerived11 : public MyDerived1  
{  
public:  
 MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase1 a;  
 MyDerived1 b;  
 MyDerived11 c;  
}



**mian2.cpp**

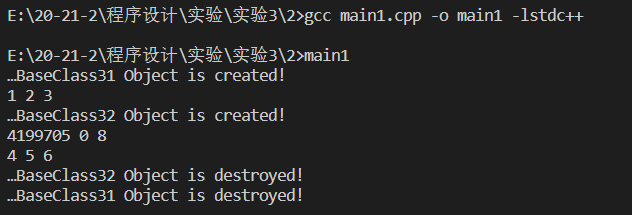
#include <iostream>  
class MyBase1  
{  
public:  
 MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is created!" << std::endl;  
 }  
 ~MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase2  
{  
 MyBase1 a1;  
  
public:  
 MyBase2()  
 {  
 std::cout << "…BaseClass2 Object is created!" << std::endl;  
 }  
 ~MyBase2()  
 {  
 std::cout << "…BaseClass2 Object is destroyed!" << std::endl;  
 }  
};  
class MyDerived1 : public MyBase2  
{  
 MyBase1 a1;  
  
public:  
 MyDerived1()  
 {  
 std::cout << "…First layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived1()  
 {  
 std::cout << "…First layer derived Object is Destroyed!" << std::endl;  
 }  
};  
class MyDerived11 : public MyDerived1  
{  
public:  
 MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase2 a;  
 MyDerived1 b;  
 MyDerived11 c;  
}



# EX2

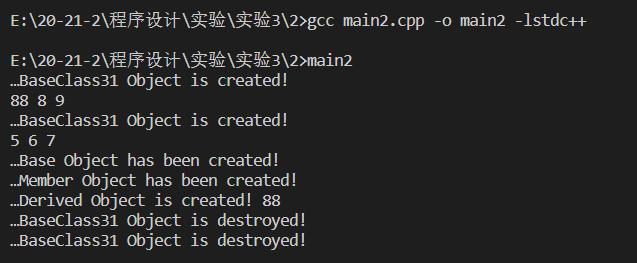
**main1.cpp**

#include <iostream>  
class MyBase31  
{  
 int a, b, c;  
  
public:  
 MyBase31(int x, int y, int z) : a(x), b(y), c(z)  
 {  
 std::cout << "…BaseClass31 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase31()  
 {  
 std::cout << "…BaseClass31 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase32  
{  
 int a, b, c;  
  
public:  
 MyBase32(int x, int y, int z)  
 {  
 std::cout << "…BaseClass32 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 a = x, b = y, c = z;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase32()  
 {  
 std::cout << "…BaseClass32 Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase31 a(1, 2, 3);  
 MyBase32 b(4, 5, 6);  
}



**main2.cpp**(实验手册程序有误，作少量改动)

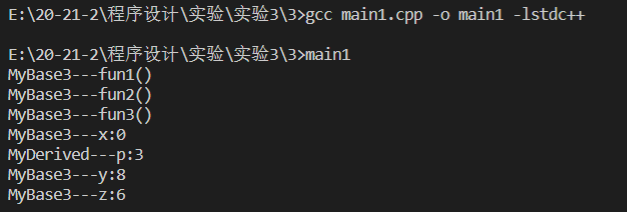
#include <iostream>  
class MyBase31  
{  
 int a, b, c;  
  
public:  
 MyBase31(int x, int y, int z) : a(x), b(y), c(z)  
 {  
 std::cout << "…BaseClass31 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase31()  
 {  
 std::cout << "…BaseClass31 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase32  
{  
 int a, b, c;  
  
public:  
 MyBase32(int x, int y, int z)  
 {  
 std::cout << "…BaseClass32 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 a = x, b = y, c = z;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase32()  
 {  
 std::cout << "…BaseClass32 Object is destroyed!" << std::endl;  
 }  
};  
class MyDerived1 : public MyBase31  
{  
 MyBase31 a = MyBase31(5, 6, 7);  
 int c;  
  
public:  
 MyDerived1(int x) : c(x), MyBase31(x, 8, 9)  
 {  
 std::cout << "…Base Object has been created!" << std::endl;  
 std::cout << "…Member Object has been created! " << std::endl;  
 std::cout << "…Derived Object is created! " << c << std::endl;  
 }  
};  
int main()  
{  
 MyDerived1 b(88);  
}



# EX3

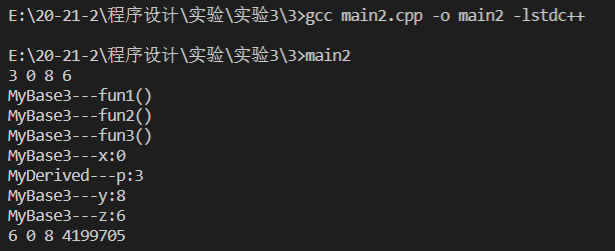
**main1.cpp**

#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase3  
{  
  
protected:  
 int y;  
 void fun2()  
 {  
 cout << "MyBase3---fun2()" << endl;  
 }  
  
public:  
 int x;  
 void fun1()  
 {  
 cout << "MyBase3---fun1()" << endl;  
 }  
 int z;  
 void MyBase(int a, int b, int c)  
 {  
 x = a;  
 y = b;  
 z = c;  
 }  
  
 int getX()  
 {  
 cout << "MyBase3---x:";  
 return x;  
 }  
  
 int getY()  
 {  
 cout << "MyBase3---y:";  
 return y;  
 }  
  
 int getZ()  
 {  
 cout << "MyBase3---z:";  
 return z;  
 }  
  
 void fun3()  
 {  
 cout << "MyBase3---fun3()" << endl;  
 }  
};  
  
class MyDerived1 : public MyBase3  
{  
public:  
 int p;  
 MyDerived1(int a) : p(a) {}  
  
 int getP()  
 {  
 cout << "MyDerived---p:";  
 return p;  
 }  
  
 void display()  
 {  
 //cout << p << "" << x << "" << y << "" << z << "" << endl;  
 fun1();  
 fun2();  
 fun3();  
 }  
};  
  
int main()  
{  
 MyDerived1 a(3);  
 a.display();  
 //cout << a.x << "" << a.p << "" << a.y << "" << a.z << endl;  
 cout << a.getX() << "\n"  
 << a.getP() << "\n"  
 << a.getY() << "\n"  
 << a.getZ() << endl;  
}



**main2.cpp**

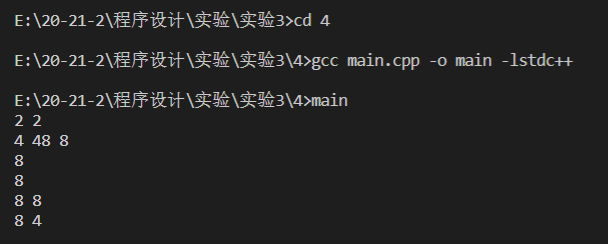
#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase3  
{  
  
protected:  
 int y;  
 void fun2()  
 {  
 cout << "MyBase3---fun2()" << endl;  
 }  
  
public:  
 int x;  
 void fun1()  
 {  
 cout << "MyBase3---fun1()" << endl;  
 }  
 int z;  
 void MyBase(int a, int b, int c)  
 {  
 x = a;  
 y = b;  
 z = c;  
 }  
  
 int getX()  
 {  
 cout << "MyBase3---x:";  
 return x;  
 }  
  
 int getY()  
 {  
 cout << "MyBase3---y:";  
 return y;  
 }  
  
 int getZ()  
 {  
 cout << "MyBase3---z:";  
 return z;  
 }  
  
 void fun3()  
 {  
 cout << "MyBase3---fun3()" << endl;  
 }  
};  
  
class MyDerived2 : public MyBase3  
{  
public:  
 int p;  
 MyDerived2(int a) : p(a) {}  
 int getP()  
 {  
 cout << "MyDerived---p:";  
 return p;  
 }  
  
 void display()  
 {  
 cout << p << " " << x << " " << y << " " << z << " " << endl;  
 fun1();  
 fun2();  
 fun3();  
 }  
};  
  
class MyDerived21 : public MyBase3  
{  
public:  
 int p;  
 MyDerived21(int a) : p(a) {}  
 int getP()  
 {  
 cout << "MyDerived21---p:" << endl;  
 return p;  
 }  
 void display1()  
 {  
 cout << p << " " << x << " " << y << " " << z << " " << endl;  
 }  
};  
  
int main()  
{  
 MyDerived2 a(3);  
 MyDerived21 b(6);  
 a.display();  
 //cout << a.x << " " << a.p << " " << a.y << " " << a.z << endl;  
 cout << a.getX() << "\n"  
 << a.getP() << "\n"  
 << a.getY() << "\n"  
 << a.getZ() << endl;  
 b.display1();  
}



# EX4

**main.cpp**

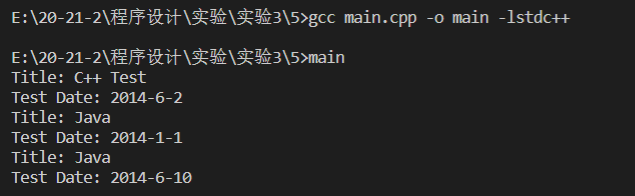
#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase  
{  
 int x;  
  
public:  
 MyBase(int a) : x(a){}  
 int getX()  
 {  
 return x;  
 }  
};  
  
class MyDerived : public MyBase  
{  
 int y;  
  
public:  
 MyDerived(int a) : y(a), MyBase(a + 4){}  
 int getY()  
 {  
 return y;  
 }  
};  
  
int main()  
{  
 MyBase a(2), \*p = &a;  
 MyDerived b(4), \*q = &b;  
 MyBase &c = a;  
 MyBase &d = b;  
 cout << a.getX() << " " << p->getX() << endl;  
 cout << b.getY() << " " << q->getY() << b.getX() << " " << q->getX() << endl;  
 a = b;  
 cout << a.getX() << " " << endl; //<< a.getY()  
 p = q;  
 cout << p->getX() << " " << endl; //<< p->getY()  
 cout << c.getX() << " " << d.getX() << " " << endl; //<< d.getY()  
 //b = a;  
 cout << b.getX() << " " << b.getY() << endl;  
}



# EX5

**main.cpp**

#include <iostream>  
using std::cout;  
using std::endl;  
using std::ostream;  
using std::string;  
  
class Date  
{  
public:  
 Date(int iyear = 2014, int imonth = 1, int iday = 1) : year(iyear), month(imonth), day(iday) {}  
  
 friend ostream &operator<<(ostream &output, Date k)  
 {  
 output << k.year << "-" << k.month << "-" << k.day;  
 return output;  
 }  
  
private:  
 int year;  
 int month;  
 int day;  
};  
  
class FinalTest : public Date  
{  
public:  
 FinalTest(string ititle = "", Date idate = Date()) : title(ititle), date(idate) {}  
  
 void print()  
 {  
 cout << "Title: " << title << endl;  
 cout << "Test Date: " << date << endl;  
 }  
  
 void setDue(Date k)  
 {  
 date = k;  
 }  
  
private:  
 string title;  
 Date date;  
};  
  
int main()  
{  
 FinalTest item1("C++ Test", Date(2014, 6, 2));  
 item1.print();  
 FinalTest item2("Java");  
 item2.print();  
 item2.setDue(Date(2014, 6, 10));  
 item2.print();  
}



# EX6

**main.cpp**

#include <iostream>  
#include <string>  
using std::cout;  
using std::endl;  
using std::string;  
  
class Shape  
{  
 public:  
 Shape(string input){  
 id = input;  
 cout << "Shape " << id << " is created" << endl;  
 }  
 ~Shape(){  
 cout << "destructed" << endl;  
 }  
 protected:  
 string id;  
};  
  
class Rectangle: public Shape  
{  
 public:  
 Rectangle(double input\_length, double input\_width= 0,string id = "Rectangle"):length(input\_length),width(input\_width),Shape(id){  
 cout << "Rectangle is created" << endl;  
 }  
 ~Rectangle(){  
 cout << "Rectangle distructed" << endl;  
 }  
 int area(){  
 return length\*width;  
 };  
 protected:  
 int length;  
 int width;  
};  
  
class circle: public Shape  
{  
 public:  
 int area(){  
 return radius\*radius\*3.14;  
 }  
 void print(){  
  
 }  
 protected:  
 int radius;  
};  
  
class square: public Rectangle, public circle  
{  
 protected:  
 circle incribe\_circle;  
};  
  
int main(){  
 Rectangle a(1,2);  
}

