实验 2 继承性实现

【实验目的】

（1） 掌握继承的概念，能够定义和使用类的继承关系。

（2） 了解在派生类中如何使用基类的成员以及基类成员在派生类中的访问控制特性。

（3） 理解虚基类在解决二义性问题中的作用。

【实验内容】

1. 有如下的程序：

```
#include <iostream>
using namespace std;
class Base
{
public :
    Base(int p1, int p2) { data1 = p1; data2 = p2; }
    int Inc1( )   { return ++data1; }
    int Inc2( )   { return ++data2; }
    void Display()
    { cout<<"data1 = "<< data1 << " data2 = " << data2 << endl; }
protected:
   int data1, data2;
};
class D1 : public Base {
public :
    D1(int p1, int p2, int p3) : Base(p1, p2) { data3 = p3; }
    int Inc1( ) { return Base::Inc1( ); }
    int Inc3( ) { return ++data3; }
    void Display( )
```

```
        {
            cout<<"data1 = "<< data1 << " data2 = " << data2 << " data3 = " << data3
<<endl;
            cout<<"Base::Display() ----";
            Base::Display( );
        }
    protected:
        int data3;
    };
    class D2 : public Base {
    public :
        D2(int p1, int p2, int p4) : Base(p1, p2) { data4 = p4; }
        int Inc1()
        {
            Base::Inc1( );          Base::Inc2( );
              return Base::Inc1( );
        }
        int Inc4( )    { return ++data4; }
        void Display( )
        {
            cout<<"data1 = "<< data1 << " data2 = " << data2 << " data4 = " << data4
<<endl;
            cout<<"Base::Display() ----";
            Base::Display( );
        }
    protected:
        int data4;
    };
    class D12 : public D1, public D2 {
    public :
```

```
        D12(int p11,int p12,int p13,int p21,int p22,int p23,int p)
        : D1(p11, p12, p13), D2(p21, p22, p23) { data5 = p; }
        int Inc1( )
        {     D1::Inc1();     D2::Inc1( );
              return D1::Inc1( );
        }
        int Inc5( ) { return ++data5; }
        void Display( )
        {
            cout<<"data1 = "<< data1 << " data2 = " << data2 <<endl;          // ①
            cout <<" data3 = " << data3 <<"data4 = "<< data4 << " data5 = " << data5
<<endl;
             cout << "D1::Display( )----";
             D1::Display( );
             cout << "D2::Display( )----";
            D2::Display( );
        }
    private:
        int data5;
    };
    int main()
    {
        D12 d(1, 2, 3, 4, 5, 6, 7);
        d. Display( );
        cout << endl;
        d.Inc1();
        d.Inc2();                                          // ②
        d.Inc3();
        d.Inc4();
        d.Inc5();
```

```
    d.D12::Inc1();
    d.Display();
    return 0;
}
```

（1）这个程序在编译时会出现错误，请根据出错提示信息找出出错的原因。

（2）修改程序中的错误，使之能正确运行。

2．设有一个点类 Point 的定义如下：

```
class Point {
    public:
        Point()    {x = 0; y = 0; }
        Point(double xv,double yv) {x = xv;y = yv;}
        Point(Point& pt) { x = pt.x;    y = pt.y; }
        double getx() { return x; }
        double gety() { return y; }
        double Area() { return 0; }
        void Show() { cout<<"x="<<x<<' '<<"y="<<y<<endl; }
    private:
        double x,y;
    };
```

编写程序，以点 Point 类为基类，派生出矩形类 Rectangle 和圆类 Circle。矩形由左上角的顶点和长、宽定义。圆由圆心和半径定义。派生类中新增的成员函数 position(Point &pt)用于判断任一坐标点是在图形内、还是在图形的边缘上，还是在图形外。

【实验指导】

1. 在派生类中对基类成员的访问应该是唯一的。但是，在有多重继承的情况下，可能会造成派生类对基类中某个成员的访问出现不唯一的情况，这时就称对基类成员的访问产生了二义性。C++为此提供了虚基类，以解决这种二义性问题。

2. 由于基类的构造函数不能够被派生类继承。因此，派生类的构造函数必须通过调用基类的构造函数来初始化基类数据成员。所以在定义矩形类 Rectangle 和圆类 Circle 的构造函数时，除了对新增数据成员进行初始化，还必须负责调用基类的构造函数使基类数据成员得以初始化。同时，在矩形类 Rectangle 和圆类 Circle 还需分别提供计算面积、周长等函数，以满足实际应用的需求。