类的对象成员 – 基本概念

- 一个C++程序中可能会涉及到许多类和对象，这些类和对象之间如何发生联系？C++语言为类和对象之间的联系提供了如下方式：
- （1）一个类对象可能是另一个类的成员。
- （2）一个类的成员函数是另一个类的友元。
- （3）一个类定义在另一个类的说明中，即类嵌套。
- （4）一个类作为另一个类的派生类。
- 关于友元的概念已经在上一讲讲过了；一个类作为另一个类的派生类，将在后面介绍；类的嵌套是指在一个类的声明中包含另一个类的声明，由于嵌套类的使用不方便，不易多用，所以在此不对嵌套类进行介绍。下面介绍类的对象成员。

- 自定义类的数据成员可以是另一个类的对象，例如类B的对象是类A的一个成员，则该成员就称为类A的对象成员，这就意味着一个A类的“大对象”包含着一个B类的“小对象”，也就是说，类B对象属于类A对象。
- 类内声明一个对象成员与声明一个int型数据成员相同，都只说明类中数据成员的类型和名称。所以，在类中声明对象成员时并不会创建该对象，这与在类外声明对象的语句表明创建一个对象不同。

- 【例2-16】类的对象成员程序示例。下面Circle类中的表示圆心的数据成员m_center是Point类的对象。

```
// DefineClass.h
//定义平面上的一个点类
class Point
{
public:
    Point(double a,double b);
    double GetX();
    double GetY();
private:
    double m_x,m_y;

};
```

```
//定义圆类
class Circle
{
public:
    Circle(double cx,double cy,double cr);
    void DisplayCircleInfo();
private:
    Point m_center;    //对象成员
    double m_radius; //非对象成员
};
```

```
// DefineClass.cpp
#include "DefineClass.h"
#include <iostream>
using namespace std;
Point::Point(double a,double b)
{
    m_x=a;
    m_y=b;
}
double Point::GetX()
{
    return m_x;
}
double Point::GetY()
{
    return m_y;
}
```

```
Circle::Circle(double cx,double cy,double cr)
        :m_center(cx,cy)
{
     m_radius=cr;
}
void Circle::DisplayCircleInfo()
{
    cout<<"圆心为:"<<m_center.GetX()
            <<","<<m_center.GetY()<<endl;
    cout<<"半径为:"<<m_radius<<endl;
}
```

```
// testObjectMember.cpp
#include <iostream>
#include "DefineClass.h"
using namespace std;
int main()
{
    Circle circle(2.3,4.6,12.5);
    circle.DisplayCircleInfo();
    return 0;
}
```