第一讲 类和对象（一）

一、面向对象程序设计概述

二、类和对象

三、构造函数

一、面向对象程序设计概述

什么是程序？程序就是可以被计算机执行的用于解决问题的一系列计算机指令。

编程过程中，两种常用的编程思想：面向过程编程思想（代表语言：C语言）和面向对象编程思想（代表语言：如C++、Java等）。

面向过程的编程是一种自上而下的设计方式。使用三步表示此种编程过程为：获得用户输入的数据，对数据进行运算并做出某种决策，在屏幕显示计算结果。

而面向对象的编程首先需要将问题分解成各个对象，从而对对象的属性和行为以及对象之间的关系进行分析。

两者的主要区别可以简单概括为：面向过程的编程是一种直接的编程方法是按照编程语言的思路考虑问题；面向对象的编程是一种抽象度更高的编程方法，它的目标是使模块的抽象度更高，可复用性更好。

在面向对象的编程中，有三大特征：封装性、继承性和多态性。封装性把数据与操作数据的函数组织在一起，不仅使程序结构更加紧凑，并且提高了类内部数据的安全性；继承性增强了软件的可扩充性以及代码重用性；多态性使设计人员在设计程序时可以对问题进行更好的抽象，有利于代码的维护和可重用。

面向对象的编程中经常使用的类型除了基本类型之外，还有“类”类型。那么什么是“类”类型呢?“对象”又是什么呢?

其实，现实世界中客观存在的任何一个事物都可以看成一个“对象”，或者说，现实世界是由千千万万个对象组成。对象可以是有形的，如一座房子、一辆车子、一个人或者一个学生等；也可以是无形的，如一个银行账户、一次贷款、一次旅行等。

任何一个对象都具有两个要素：属性和行为。其中属性用于描述客观事物的静态特征，为名词。而行为用于描述事物的动态特征，为动词。如一个人，他有姓名、性别、身高、体重等属性，有走路、说话、打手势、学习和工作等行为。又如一个银行账户，属性：用户名，密码，余额等，行为：转账，查看余额，贷款等。

而类是对客观世界中具有相同属性和行为的一组对象的抽象，它为属于该类的全部对象提供了统一的抽象描述，其内容包括属性和行为。例如：人可以作为一个类，它是世界上所有实体人如张三、李四、王五等的类型，而实体人张三、李四等则是人这个类的具体实例。

所以类和对象的关系可以总结为：类是对象的类型，对象是类的变量。

【例1-1】一个简单的C++类

```
/**Point头文件**/

//预处理命令: #ifndef *** #define***  #endif作用为: 防止头文件被重复包含
#ifndef __test__Point__
#define __test__Point__

#include <iostream>
using namespace std;

class Point//类型名为Point
{
public:
    int getX();
    int getY();
    void setPoint(int _x,int _y);
    void print();

private:
    int x;
    int y;
};//分号不能省略

#endif
```

```
/**Point实现文件**/

#include "Point.h"
int Point::getX() {
    return x;
}

int Point::getY() {
    return y;
}

void Point::setPoint(int _x,int _y) {
    x = _x;
    y = _y;
}

void Point::print() {
    cout<<"("<<x<<","<<y<<")"<<endl;
}
```

```
/**测试文件**/

#include <iostream>
#include "Point.h"
using namespace std;
int main(int argc, const char * argv[]) {
    Point p;//声明对象p
    p.print();//打印对象p
    p.setPoint(1, 1);//修改对象p的值
    cout<<p.getX()<<endl;//获得对象p的x坐标并打印
    cout<<p.getY()<<endl;//获得对象p的y坐标并打印
    return 0;
}
```

程序运行结果如下：
(0,0)
1
1

示例分析：

(1) 定义类时，类名首字母一般大写；

(2) 同一个作用域中类名需要唯一；

二、类和对象

上一节示例1-1中，使用关键字class定义了一个简单的C++类。类中指定了属性（x、y），即类的成员变量；同时指定了行为（print、getX 、getY、 print），即类的成员函数。本节示例2-1将定义更复杂的类结构。

【例2-1】定义一个圆类。属性：圆心坐标以及半径，行为：对圆的信息进行读取、设置以及打印，计算两个圆之间的距离。

```
/**Circle头文件**/

#ifndef __test__Circle__
#define __test__Circle__

#include <iostream>
#include <cmath>
using namespace std;
class Circle
{
public:
    Circle();//构造函数
    float getRadius();//访问函数
    float getCenterX();//访问函数
```

```
    float getCenterY();//访问函数
    void setCircle(float _radius,float _xCenter,float _yCenter);//设置函数
    float distance(Circle c1);//求距离
    void print();//打印函数

private:
    float radius;
    float xCenter;
    float yCenter;
};//分号不能省略
#endif

/**Circle实现文件**/

#include "Circle.h"
//类定义之外实现成员函数 函数名之前必须加“类名::”修饰
Circle::Circle() {
    radius = 0.0;
    xCenter = 0.0;
    yCenter = 0.0;
}

float Circle::getRadius() {
    return radius;
}
```

```
float Circle::getCenterX() {
    return xCenter;
}

float Circle::getCenterY() {
    return yCenter;
}

void Circle::setCircle(float _radius, float _xCenter, float _yCenter) {
    radius = _radius;
    xCenter = _xCenter;
    yCenter = _yCenter;
}

float Circle::distance(Circle c1) {
    return sqrt((xCenter-c1.xCenter)*(xCenter-c1.xCenter)+(c1.yCenter-
yCenter)*(c1.yCenter-yCenter));
}

void Circle::print() {
    cout<<"radius:"<<radius<<"
xCenter:"<<xCenter<<"yCenter:"<<yCenter<<endl;
}
```

```
/**测试文件**/

#include <iostream>
#include "Circle.h"
using namespace std;

int main(int argc, const char * argv[]) {
    Circle c;
    Circle c1;
    c.print();
    c1.print();
    c1.setCircle(1.0, 1.0, 1.0);
    cout<<c.getCenterX()<<endl;
    cout<<c.getCenterY()<<endl;
    cout<<c.getRadius()<<endl;
    cout<<c.distance(c1)<<endl;
    return 0;
}
```

程序运行结果如下：

```
radius:0.0 xCenter:0.0 yCenter:0.0
radius:0.0 xCenter:0.0 yCenter:0.0
0.0
0.0
0.0
1.41421
```

示例分析:

⑴ 使用关键字class 定义类，类名为Circle;

⑵ 类中成员变量包括：radius,xCenter以及yCenter，类型为float。成员变量的类型可以指定为：基本类型、数组类型、结构体类型、指针类型或者其它类类型;

⑶ 类中成员函数包括：构造函数Circle,访问函数getRadius、 getCenterX、getCenterY ，设置函数setCircle，打印函数print，求距离函数distance;

⑷ 类中的成员函数既可以在类定义内部实现，也可以在类定义外部实现，在类定义外部实现成员函数时，需要在函数名之前添加“类名::”进行修饰;

⑸ 类中定义成员时，使用了关键字private以及public修饰。被private修饰的成员为类中私有成员，类外不可见。被public修饰的成员为类中公有成员，类外可见;

⑹ 定义类时，类结构末尾的分号不可以省略;

三、构造函数

构造函数是一种比较特殊的成员函数，用于创建并初始化对象。声明对象时，构造函数会被编译器自动调用。

构造函数的四个特点：

⑴ 类中构造函数的访问权限必须为公有（public）；

⑵ 构造函数名和类名相同；

⑶ 构造函数没有返回值；

⑷ 构造函数可以带参数；

【例3-1】构造函数的示例

```
/**Circle头文件**/

#ifndef __test__Circle__
#define __test__Circle__

#include <iostream>
using namespace std;
class Circle
{
public:
    Circle();
    Circle(int _r);
    Circle(int _r,int _x);
    Circle(int _r,int _x,int _y);
    void print();

private:
    float radius;
    float xCenter;
    float yCenter;
};//分号不能省略
#endif
```

```
/**Circle实现文件**/

#include "Circle.h"

Circle::Circle() {
    radius = 0.0;
    xCenter = 0.0;
    yCenter = 0.0;
}

Circle::Circle(int _r) {
    radius = _r;
    xCenter = 0.0;
    yCenter = 0.0;
}

Circle::Circle(int _r,int _x) {
    radius = _r;
    xCenter = _x;
    yCenter = 0.0;
}

Circle::Circle(int _r,int _x,int _y) {
    radius = _r;
    xCenter = _x;
    yCenter = _y;
}
```

```
void Circle::print() {
    cout<<"radius:"<<radius<<" xCenter:"<<xCenter<<"
yCenter:”<<yCenter<<endl;
}

/**测试文件**/

#include <iostream>
#include "Circle.h"
using namespace std;
int main(int argc, const char * argv[])
{
    Circle c;
    Circle c1(1.0);
    Circle c2(2.0,3.0);
    Circle c3(4.0,5.0,6.0);
    c.print();
    c1.print();
    c2.print();
    c3.print();
    return 0;
}
```

程序运行结果如下：

```
radius:0.0 xCenter:0.0 yCenter:0.0
radius:1.0 xCenter:0.0 yCenter:0.0
radius:2.0 xCenter:3.0 yCenter:0.0
radius:4.0 xCenter:5.0 yCenter:6.0
```

示例分析:

(1) 类中提供了四个构造函数，其中第一个为默认构造函数，其余三个为一般构造函数;

(2) 不传递任何参数就可以被调用的构造函数为默认构造函数;

(3) 如果类中没有定义构造函数，那么编译器会提供一个默认构造函数，否则，不再提供;