

实验 1 类的设计与对象使用

【实验目的】

- (1) 熟悉 Visual Studio 开发环境。
- (2) 掌握多文件结构程序项目的建立过程。
- (3) 初步掌握面向对象程序的多文件结构项目的开发运行步骤。
- (4) 掌握类的定义和使用以及类对象的声明和使用方法，理解具有不同访问属性的成员的访问方式。
- (5) 掌握构造函数和析构函数的作用和编写方法。
- (6) 了解并掌握静态成员、常成员的使用。

【实验内容】

1. 下面的程序在编译时会出错，请修改程序，使之能正确运行。然后，将得到的正确程序分成 3 个文件，文件 `MyClass.h` 包括类的定义，文件 `MyClass.cpp` 包括类成员函数的实现，文件 `App.cpp` 包括类的使用。调试、运行该程序，使之获得正确结果。

```
#include<iostream>

using namespace std;

class MyClass
{
public:
    MyClass() { x=0;y=0; }
    MyClass(int a = 0, b = 0);
    Print();
private:
    int x, y;
};

MyClass::MyClass(int a = 0, int b =0)
```

```

{
    x = a;
    y = b;
}
void MyClass::Print()
{
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
}
int main()
{
    MyClass obj1,obj2(5,8);
    obj1.x =1;
    obj1.y =3;
    obj1.Print();
    obj2.Print();
    return 0;
}

```

2. 已知两个矩形的长和宽，用面向对象的概念编程求它们的面积和周长。假设矩形 1 的长和宽分别为 20 和 50；矩形 2 的长和宽分别为 3.6 和 4.5。(先定义矩形类再实例化两个对象)

3. 下面程序中定义了一个整型动态数组，编写了 3 个构造函数。

```

#include<iostream>
using namespace std;
class IntArray {
public:
    IntArray(int sz);           // 数组初始化为 sz 个元素，初值全为 0

```

IntArray(int *array, int sz); // 用静态数组 array 的前 sz 个元素初始化新数组

IntArray(const IntArray &rhs); // 拷贝构造函数

void printAll();

private:

int *ia;

int size;

};

IntArray::IntArray(int sz) {

size = sz; // 设置数据成员

ia = new int[size];

for (int ix=0; ix < size; ++ix)

ia[ix] = 0; // 初始化数组元素

}

IntArray:: IntArray(int *array, int sz) {

size = sz; // 设置数据成员

ia = new int[size];

for (int ix=0; ix < size; ++ix)

ia[ix] = array[ix]; // 拷贝数据成员

}

IntArray:: IntArray(const IntArray &rhs)

{

size = rhs.size;

ia = new int[size];

for (int ix = 0; ix < size; ix++)

ia[ix] = rhs.ia[ix];

}

void IntArray::printAll()

{

for (int ix=0; ix < size; ++ix)

```

        cout<<ia[ix]<<" ";
    cout<<endl;
}
int main()
{
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    IntArray arr1(10), arr2(a,5),arr3(arr2);
    arr1.printAll();
    arr2.printAll();
    arr3.printAll();
    return 0;
}

```

(1) 为 IntArray 加上析构函数并编译运行程序，体会类的构造函数和析构函数的作用。

(2) 程序中，类 IntArray 的三个构造函数都是以相似的方式来实现的。一般来说，当两个或多个函数重复相同的代码时，可将这部分代码抽取出来，形成独立的函数，以便共享。以后如果需要改变这些实现，则只需改变一次，而且这种实现的共享本质更容易为大家所理解。怎样把构造函数中的代码抽取出来形成独立的共享函数呢？请给出一种可能的实现。按你给出的实现方法修改程序并重新编译运行。

4. 设有一个点 myPoint 类的定义如下：

```

class myPoint {
public:
    myPoint(double x0,double y0):x(x0),y(y0) {}
    myPoint(myPoint &np):x(np.x),y(np.y) {}
    double GetX() { return x;}
    double GetY() {return y;}
    void SetX(double x0) {x=x0;}
    void SetY(double y0) {x=y0;}
}

```

```

void SetPoint(double x0,double y0) {x=x0;y=y0;}

void SetPoint(myPoint &np) { x=np.x; y=np.y;}

double  GetLength(myPoint p) {
    return sqrt((x-p.x)*(x-p.x) +(y-p.y)*(y-p.y));
}

void Printit() { cout<<" ("<<x<<","<<y<<") ";}

private:

    double x ,y;

};

```

试定义一个三角形 **Triangle** 类，在 **Triangle** 类中以点 **myPoint** 类的 3 个对象 **p1**、**p2**、**p3** 作为数据成员，表示三角形的三个顶点。**Triangle** 类具有计算三角形的周长和面积的功能。请编写程序上机调试并运行。

【实验指导】

1. 头文件中通常包含常量、类型、函数原型的定义。类是用户定义的类型，故将类的定义放在头文件中。函数的定义即函数的实现应放在实现文件 **.cpp** 文件中。这样的安排体现了信息隐藏和模块化。将类成员函数的实现和类的使用放在不同的 **cpp** 文件中，可将类的实现与类的使用分离，体现了对接口编程的思想。

文件 **MyClass.h** 中的结构：

```

#include<iostream>

using namespace std;

// MyClass 的定义

```

文件 **MyClass.cpp** 中的结构：

```

#include "MyClass.h"

// MyClass 成员函数的定义

```

文件 **App.cpp** 中的结构：

```

#include "MyClass.h"

// main 函数的定义

```

避免头文件内容被重复引入。头文件内容被重复引入会导致标识符重复定义

的错误。用预编译条件指令来限制头文件内容的引入。例如：

```
#ifndef MYCLASS
#define MYCLASS
// MyClass.h 头文件的内容
#endif
```

通用格式说明：

```
#ifndef <标识> //如果没有定义<标识>
#define <标识> //定义<标识>
```

.....

.....

```
#endif
```

<标识>在理论上来说可以是自由命名的，但每个头文件的这个“标识”都应该是唯一的。标识的命名规则一般是头文件名全大写，前后加下划线，这样如果有两个地方都包含这个头文件，就不会出现两次包含的情况。因为在第二次包含时<标识>已经有定义了，所以就不再 `include` 了。

2. 关于`#include`"文件名"和`#include <文件名>`的说明。使用`<>`表示在包含文件目录中去查找（包含目录是由用户在设置环境时设置的），而不在源文件目录去查找；使用双引号`"`则表示首先在当前的源文件目录中查找，若未找到才到包含目录中去查找。

3. 题（3）可参考教材第 192 页关于“浅复制”和“深复制”的示例来实现析构函数的定义。在类中将构造函数的代码抽取出来形成独立的函数，然后在构造函数中调用执行，注意函数参数的设置和定义。

4. 题（4）由于在 `Triangle` 类中需要定义 `myPoint` 类的对象，因此需要在 `Triangle` 类的构造方法的成员初始化列表中实现对成员对象的初始化。