作业三

学号：2022217587 姓名：党存远 作业区间：11-21~11-28

**1. P148-4-1**

**解释public和private的作用，公有类型成员与私有类型成员有些什么区别？**

1)源代码

#include<iostream>

using namespace std;

class Person

{

private:

int age;

int score;

string name;

string sex;

public:

void input(string Name)

{

name = Name;

cout << "姓名为： " << name << endl;

}

};

int main()

{

Person p;

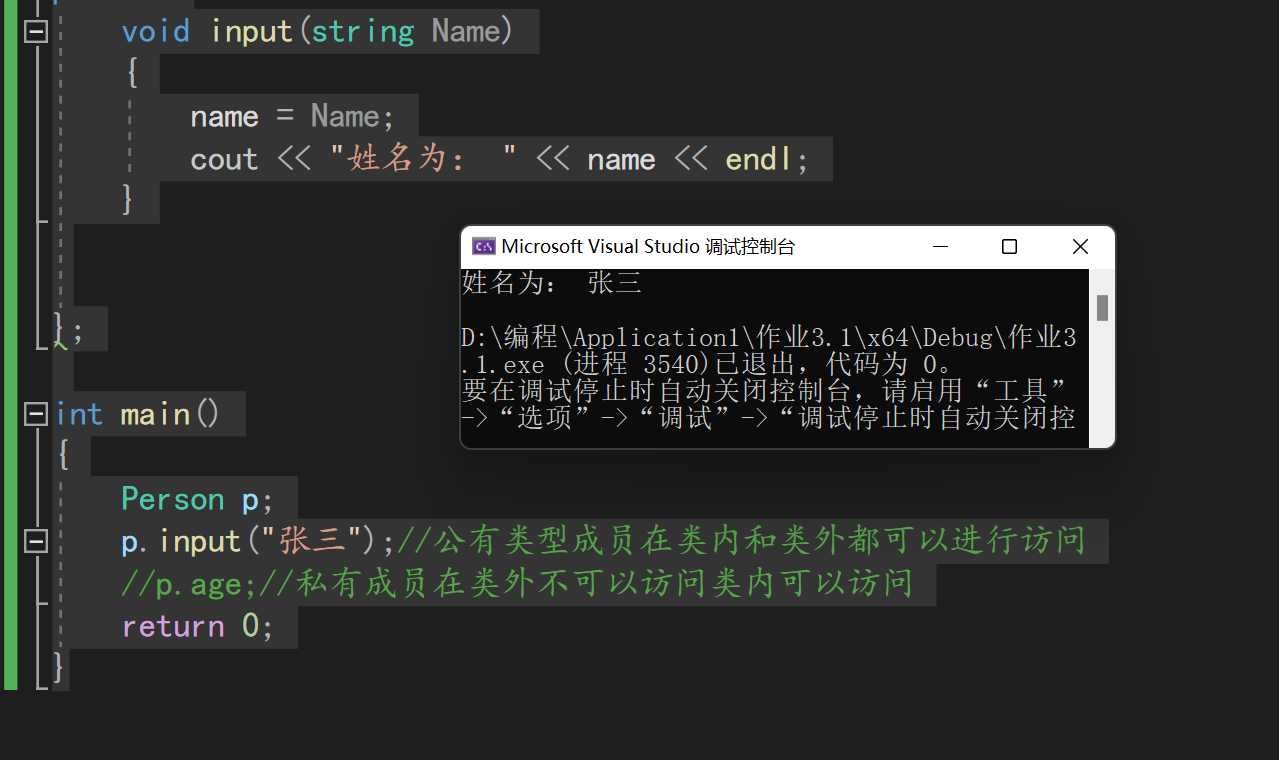
p.input("张三");//公有类型成员在类内和类外都可以进行访问

//p.age;//私有成员在类外不可以访问类内可以访问

return 0;

}

2)运行结果截图



图一：private和public类下的成员的区别

**2. P148-4-2**

**protected关键字有何作用？**

1)源代码

#include<iostream>

using namespace std;

class Person

{

protected:

int age;

int score;

string name;

string sex;

public:

void input(int Score)

{

score = Score;

cout << "成绩为： " << score << endl;

}

};

int main()

{

Person p;

p.input(94);//公有类型成员在类内和类外都可以进行访问

//p.score;//protected下的成员在类外不可以访问类内可以访问，对该类成员进行保护防止被修改

return 0;

}

2)运行结果截图

**.**

图二：protected类下的成员特性显示

**3. P148-4-6**

**什么叫做复制构造函数？复制构造函数何时被调用？**

1)源代码

#include<iostream>

using namespace std;

//复制构造函数的调用时机

class Person

{

public:

Person()

{

cout << "Person的默认构造函数的调用" << endl;

}

~Person()

{

cout << "Person的析构函数调用" << endl;

}

int m\_Age;

Person(int age)//Person的有参构造函数

{

cout << "Person的有参构造函数的调用" <<endl;

m\_Age = age;

}

Person (const Person&p)

{

m\_Age = p.m\_Age;

cout << "Person的复制构造函数的调用： " << endl;

}

};

//1、使用一个已经创建完毕的对象来初始化一个新对象

void test01()

{

Person p1(20);

Person p2(p1);

cout << "p2的年龄为： " << p2.m\_Age << endl;

}

//2、值传递的方式给参数传值

void dowork1(Person p)

{

}

void test02()

{

Person p;

dowork1(p);

}

//3、值方式返回局部对象

Person dowork2()

{

Person p1;

return p1;

}

void test03()

{

Person p = dowork2();

}

int main()

{

//复制构造函数的调用

test01();//1、使用一个已经创建完毕的对象来初始化一个新对象

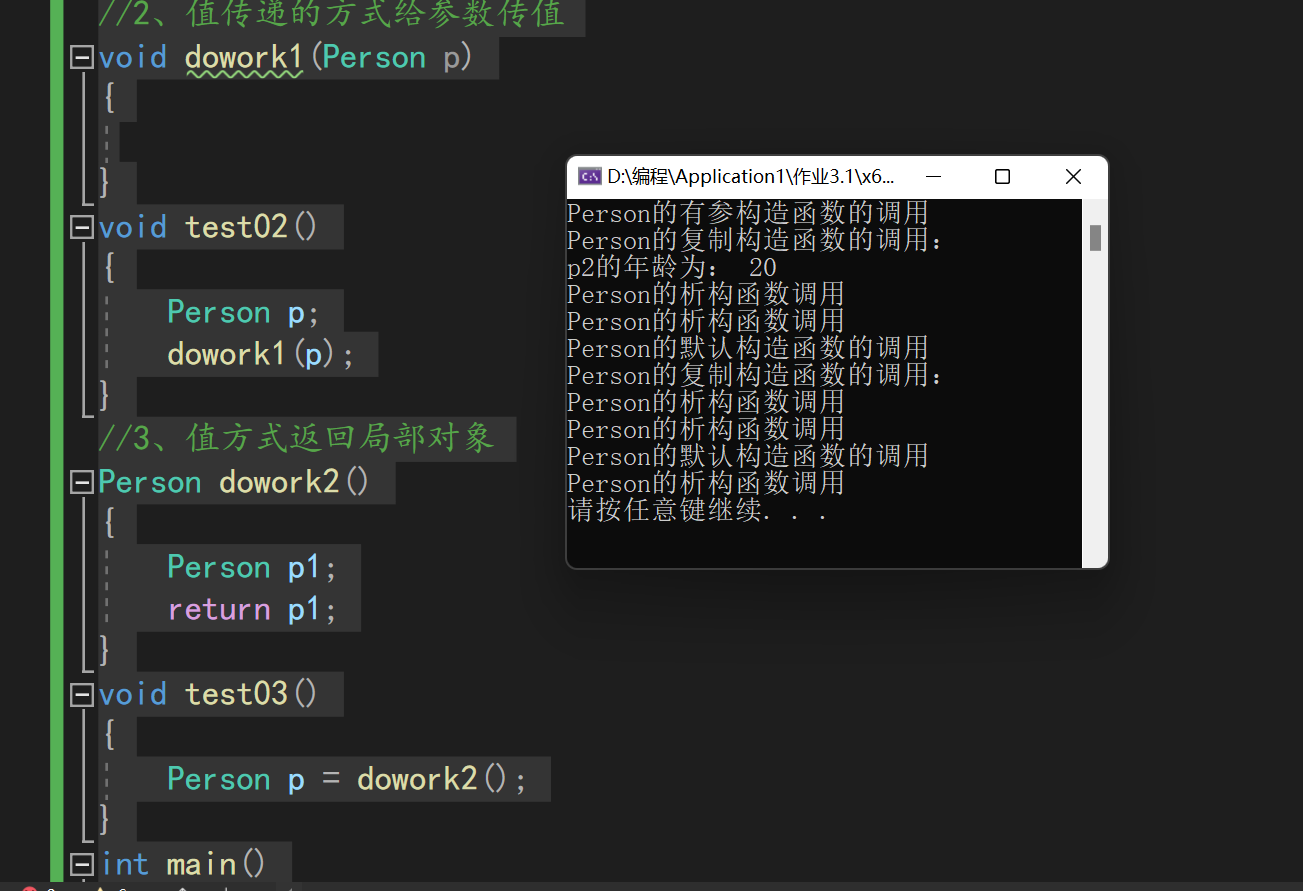
test02();//2、值传递的方式给参数传值

test03();//3、值方式返回局部对象

system("pause");

return 0;

}2)运行结果截图



图三：复制构造函数的调用时机

**4. P148-4-8**

**定义一个Dog🐶🦴类，包含的age、weight等属性，以及对这些属性操作的方法。实现并测试这个类。**

1)源代码

#include<iostream>

using namespace std;

class dog

{

public:

void inputage(int age)

{

D\_age = age;

}

void inputweight(int weight)

{

D\_weight = weight;

}

void coutage(int age)

{

cout << "小狗的年龄为： " << age<<"个月" << endl;

}

void coutweight(int weight)

{

cout << "小狗的体重为： " << weight<<"千克" << endl;

}

private:

int D\_age;

int D\_weight;

};

int main()

{

int age1 = 0, weight1 = 0;

cout << "请输入年龄： " << endl;

cin >> age1;

cout << "请输入体重： " << endl;

cin >> weight1;

dog D1;

D1.inputage(age1);

D1.inputweight(weight1);

D1.coutage(age1);

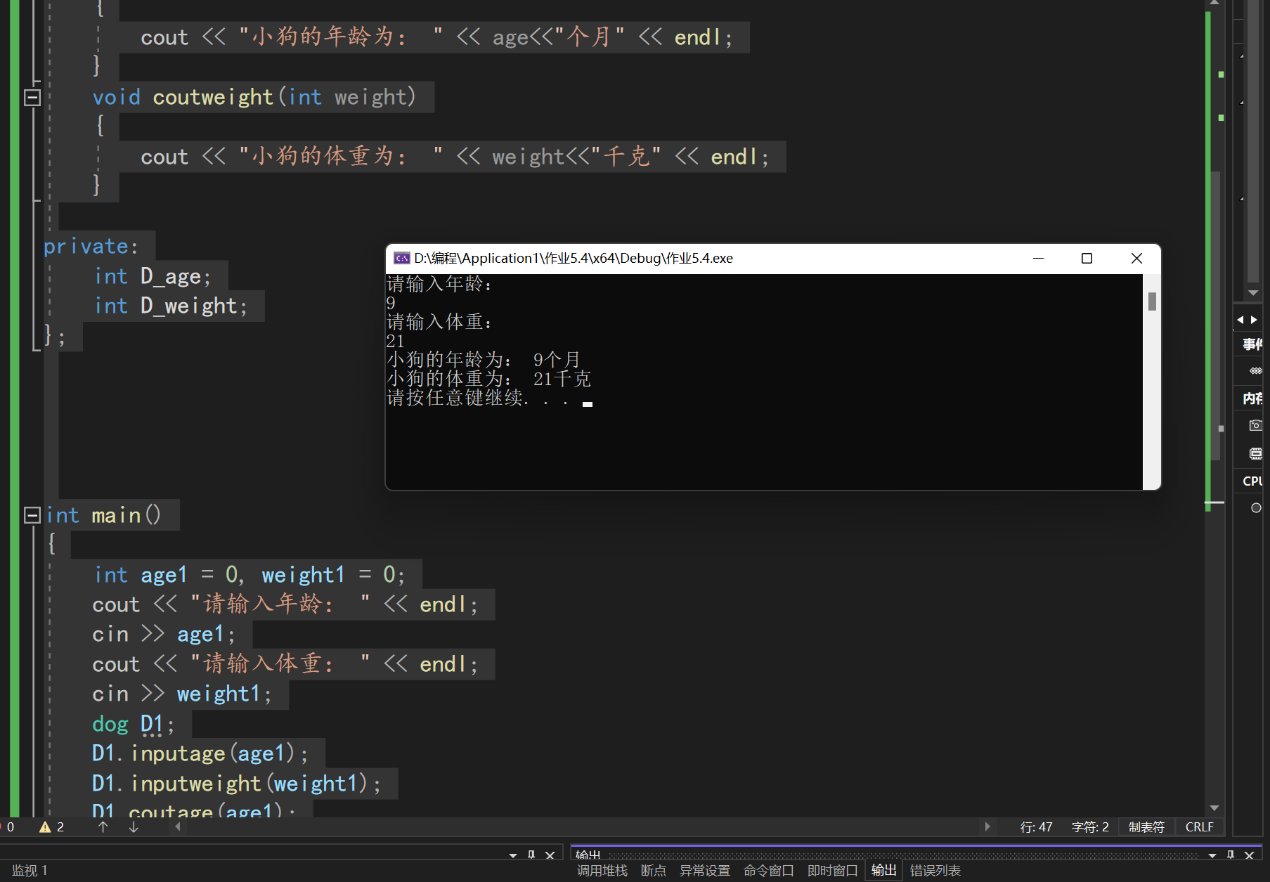
D1.coutweight(weight1);

system("pause");

return 0;

}

2)运行结果截图



图四：封装狗狗类

**5. P148-4-9**

**设计并测试一个名为Rectangle的矩形类，其属性为矩形的左下角与右上角两个点的坐标，能计算矩形的面积。**

1)源代码

#include<iostream>

using namespace std;

class Rectangle

{

public:

double Area(double len, double wid)

{

l = len;

w = wid;

s = l \* w;

return s;

}

private:

double l, w, s;

};

int main()

{

double l1, l2, w1, w2;

double len, wid, Area;

cout << "请输入A点的坐标：(请先输入横坐标，再输入纵坐标)" << endl;

cin >> l1 >> w1;

cout << "请输入B点的坐标：(请先输入横坐标，再输入纵坐标)" << endl;

cin >> l2 >> w2;

len = abs(l1 - l2);

wid = abs(w1 - w2);

Rectangle R1;

Area = R1.Area(len, wid);

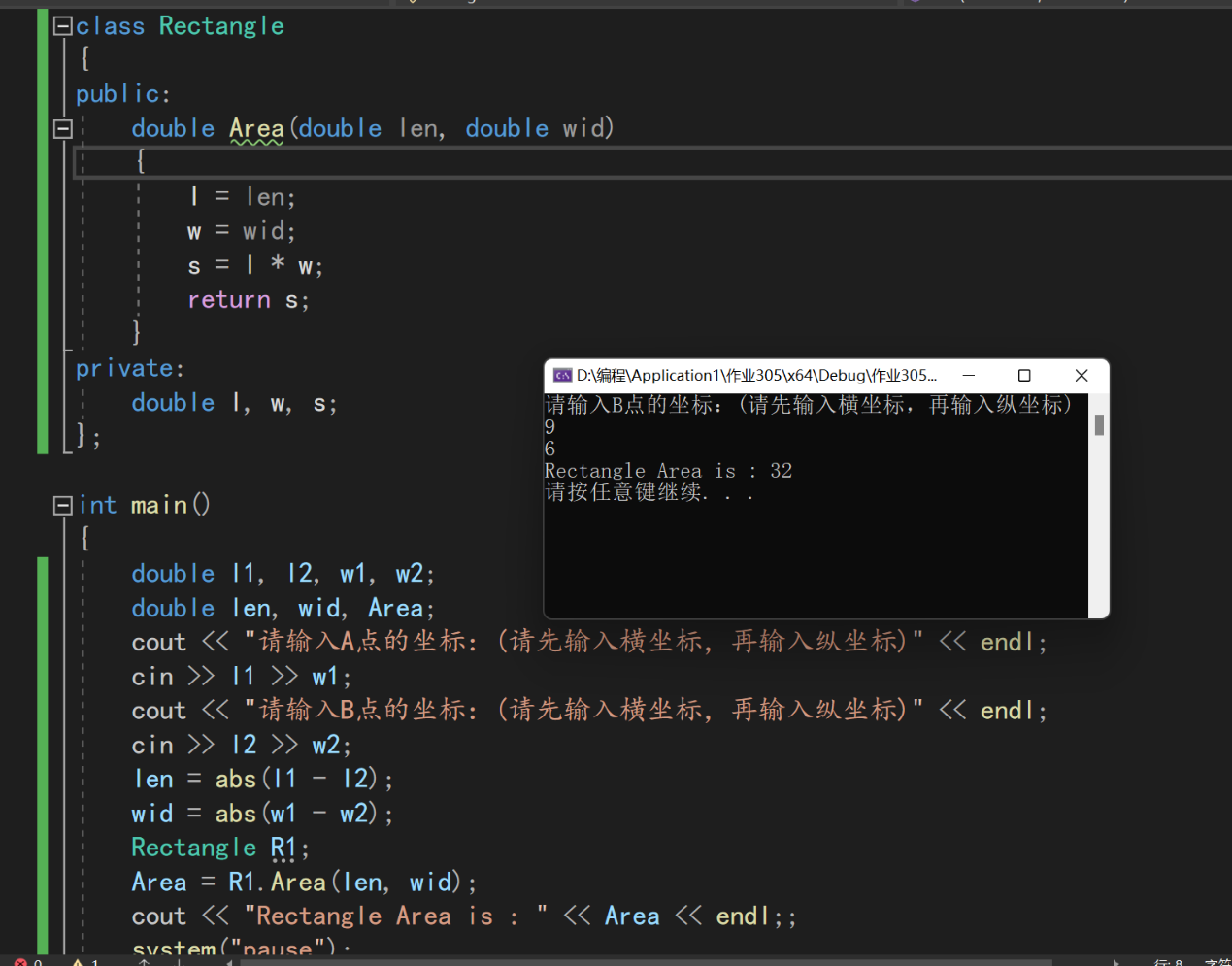
cout << "Rectangle Area is : " << Area << endl;;

system("pause");

return 0;

}

2)运行结果截图



图五：矩形面积的计算

**6. P148-4-13**

**定义一个Circle类，有数据成员radius（半径），成员函数getArea()，计算圆的面积，构造一个Circle的对象进行测试。**

1)源代码

#include<iostream>

using namespace std;

#define pai 3.1415;

#include <cmath>

class Circle

{

public:

void getArea(double r)

{

radius = r;

double Area = radius \* radius \* pai;

cout << "半径为：" << radius << "的圆的面积为：" << Area << endl;

}

private:

double radius;

};

int main()

{

Circle c1;

double r1=0;

cout << "请输入所求圆的半径：" << endl;

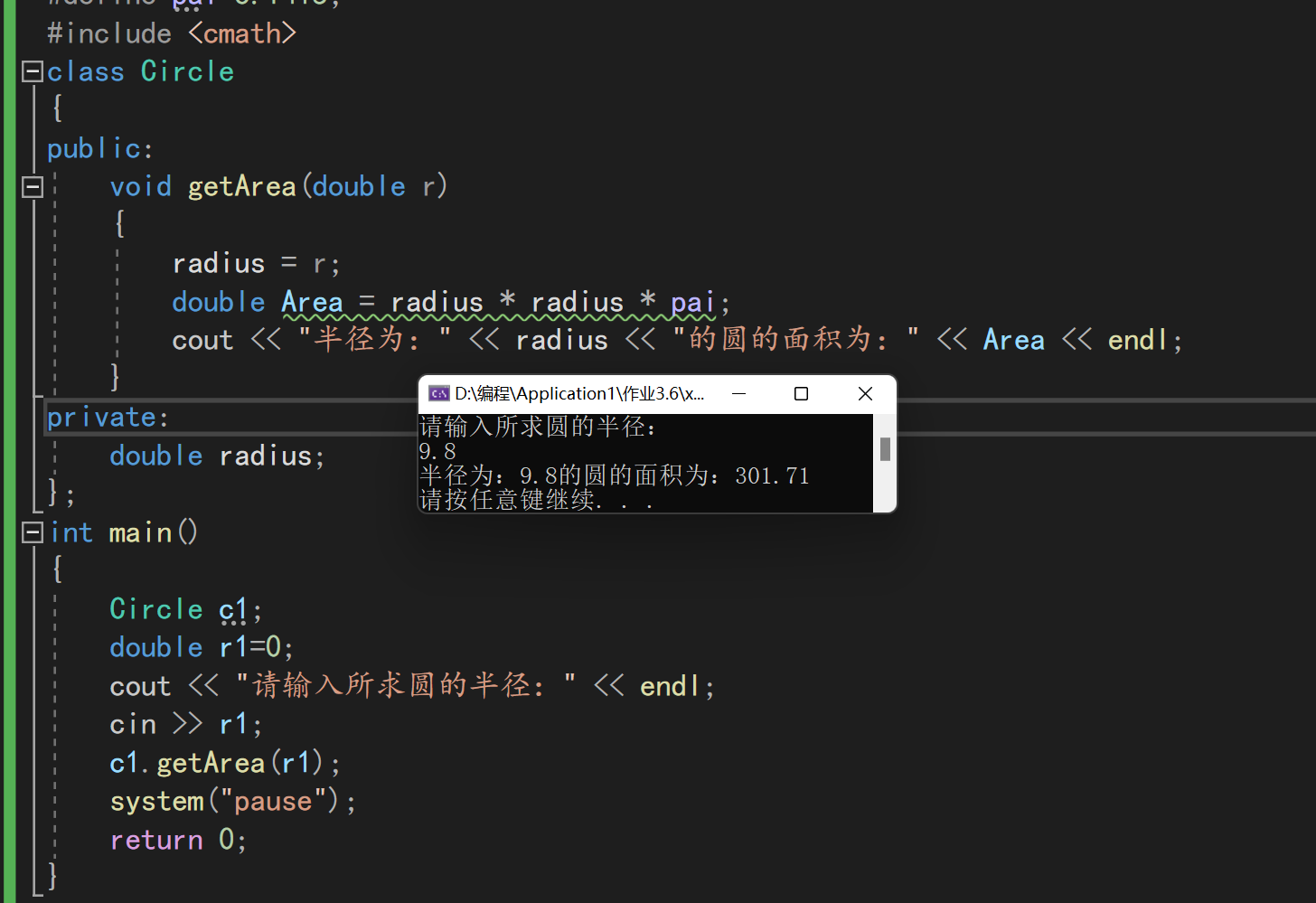
cin >> r1;

c1.getArea(r1);

system("pause");

return 0;

}2)运行结果截图



图六：圆的面积的计算

**7. P149-4-20**

**定义一个复数类Complex，使得下面的代码能够工作：**

Complex c1(3, 5); //用复数 3 + 5i 初始化c1

Complex c2 = 4.5; //用实数 4.5 初始化c2

c1.add(c2);       //将 c1 与 c2 相加，结果保存在c1中

c1.show();        //将 c1 输出（这时的结果应该是 7.5 + 5i ）

1)源代码

#include<iostream>

using namespace std;

class Complex

{

public:

Complex(double r, double i=0)

{

re = r;

im = i;

}

void add(Complex c)

{

re = re + c.re;

im = im + c.im;

}

void show()

{

cout << re << "+" << im << "i" << endl;

}

private:

double re, im;

};

int main()

{

Complex c1(3, 5);

Complex c2 = 4.5;

c1.add(c2);

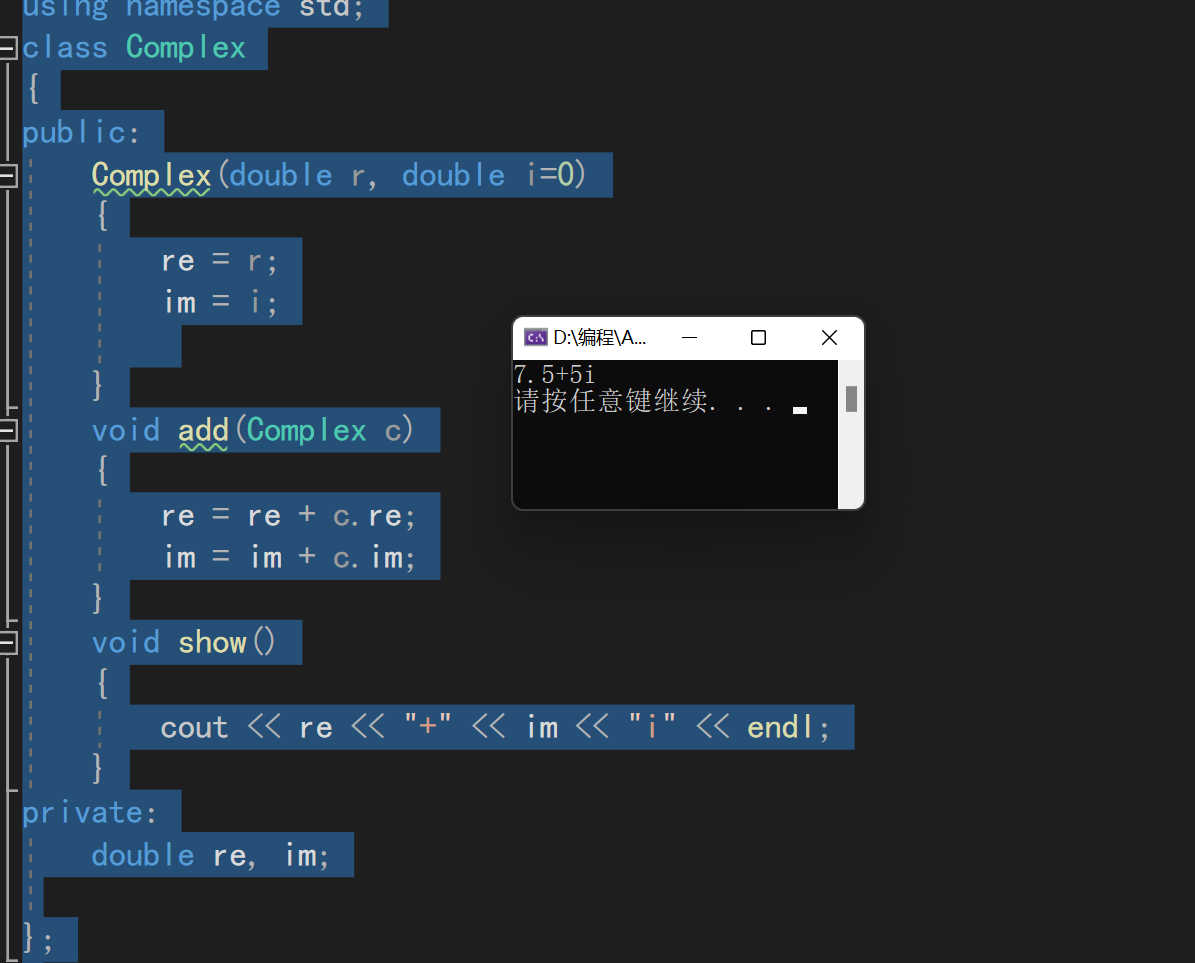
c1.show();

system("pause");

return 0;

}

2)运行结果截图



**图七：复数类的实现**

**8. P149-4-22**

**声明枚举类型Weekday，包括SUNDAY到SATURDAY七个元素。在程序中声明Weekday类型的变量，对其赋值，并声明整型变量，看看能否对其赋Weekday类型的值。**

1)源代码

#include<iostream>

using namespace std;

enum Day

{

Monday = 1,Tuesday=2,Wednesday,Thursday,

Friday,Saturday,Sunday=7

};

int main()

{

Day d1;

d1 = Sunday;

cout << "d1代表的是：星期"<<d1<< endl;

int d2;

d2 = Friday;

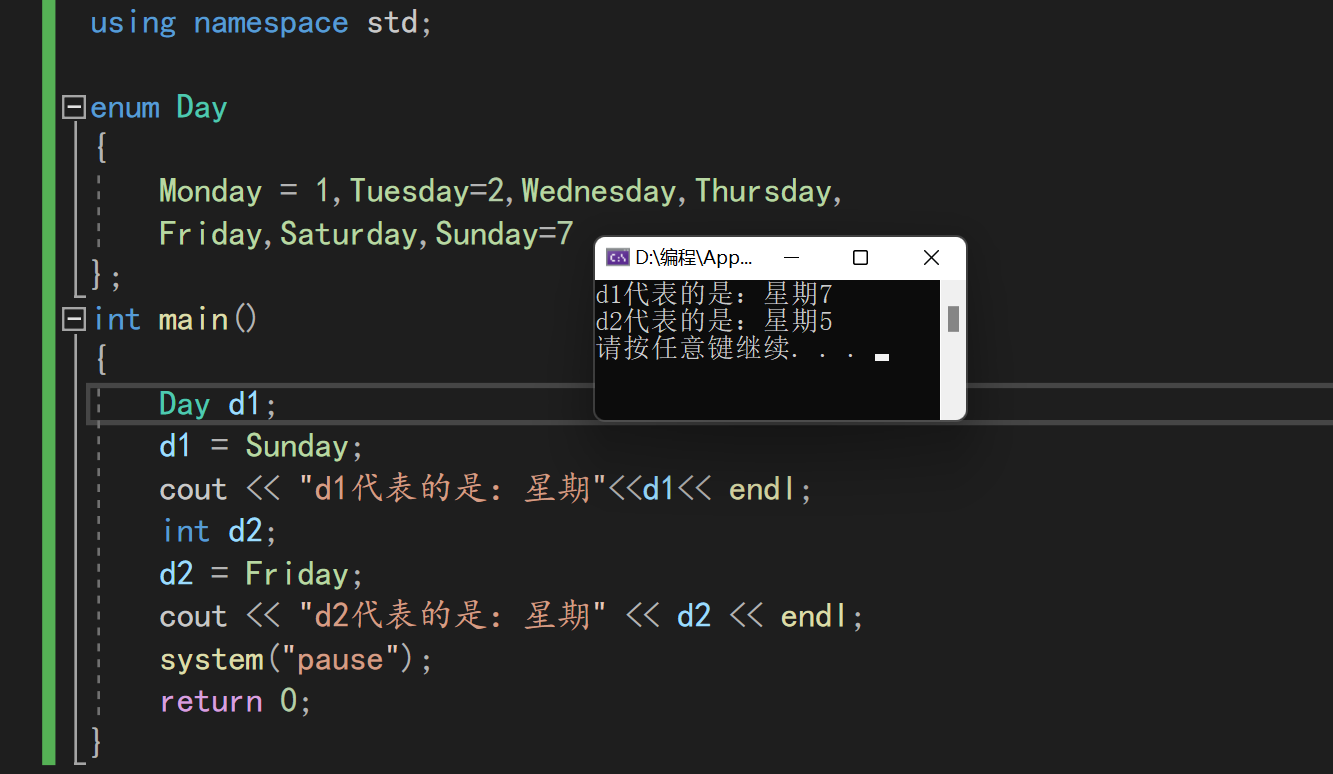
cout << "d2代表的是：星期" << d2 << endl;

system("pause");

return 0;

}

2)运行结果截图



图八：枚举类的实现

**2. Bug总结及心得体会**

**1、有关复制构造函数的调用时机不够熟练，不能完全记忆清楚，导致编程时出现问题。**

**2、进行矩形面积的计算程序编写过程中，总希望将最左下角的点与最右上角的的点作为一个类包含在矩形类中，但未能实现，后采用将两者的横纵坐标之差分别作为矩形的长和宽进行矩形面积的计算。**

**3、在定义复数的类的过程中，搞混了该类中的虚部与实部，错误的将不同的对象的实部与虚部看作相同的，构造了很多private的属性导致程序崩溃，后续发现每个对象的re与im是不同的，只需要注明清楚是谁的re,im即可。**

**4、在定义一个枚举类时对于该类的定义方式不够熟练导致无法快速进行编程。**

**在这次的编程过程中，发现了许多的问题，明白了需要继续多写代码来提升能力。**

**3. 评分**

|  |  |  |
| --- | --- | --- |
| **评分标准及细节** | **整体评价** | **评分及签名** |
| **1）（90-100）按时提交作业；编码精练，交互完好，能够得到正确的运行结果；格式规范，截图清晰、美观，在必要的情况下给予说明；小结及体会深刻。** | **□** |  |
| **2）（80-90）按时提交作业；编码良好，能够得到正确的运行结果；格式较为规范，截图较为清晰、美观；小结及体会较为深刻。** | **□** |  |
| **2）（70-80）按时提交作业；编码质量一般，能够得到正确的运行结果；格式较为规范，截图较为清晰、美观；小结及体会完成敷衍。** | **□** |  |
| **3）（60-70）作业未能按时提交；或编码较差；或采用手机拍照作为截图。** | **□** |  |
| **4）（60以下）未提交作业。** | **□** |  |