|  |  |
| --- | --- |
| **编号:** | **3-2** |

****

信息科学与工程学院实验报告

《面向对象程序设计》

**Object-Oriented Programming**

|  |  |
| --- | --- |
| 姓名： | 张泽浩 |
| 学号： | 202111000212 |
| 班级： | 计工本2102 |
| 导师: | 张庆科 |
| 时间： | 2021年11月4日 |

**《面向对象程序设计》实验报告**

**基本要求：**实验报告包含实验目的、实验内容、实验过程（详细操作流程）、实验结果（程序运行结果高清截图）、实验分析总结五个部分。报告中若涉及代码程序，请在附录部分提供完整程序源码及源码托管地址(基于Highlight软件导入源码)。报告撰写完毕后请提交PDF格式版本报告到课程云班课系统。

1. **实验目的**
2. 理解对象的概念及本质
3. 掌握对象的基本使用方法
4. 掌握对象内成员的访问方法
5. 理解并掌握this指针的用法
6. 掌握对象作为函数参数的用法
7. **实验内容**
8. 任务一: 基础类编程设计实验

完成教材P100页,第(4)题程序设计

1. 任务二: 类内构造函数设计实验

设计一个矩形类Rectangle, 该类包含2个私有数据成员变量: double a, b; 对外提供的接口包括: 初始化类对象, 释放对象(三构一析), 输出矩形的边长(get函数), 修改矩形的边长(set函数),判断矩形是否为正方形, 输出矩形的周长,输出矩形面积. 然后, 基于该矩形类探索分析其对象作为函数void func(Rectangle v, Rectangle\* p, Rectangle& r)参数时的用法.

**具体包括:**

1. 设计三种不同类型构造函数完成对象初始化:无参构造函数, 有参构造函数(或含有默认参数数值), 拷贝构造函数.
2. 设计任意类的成员函数,通过该函数验证类内各个成员函数中this指针的本质及其作用.
3. 通过sizeof运算符计算并验证类的对象存储的内容,探索对象存储的规律.
4. 根据下面的程序段,论证分析不同传参方式对实参对象的影响

01 void func(Rectangle v, Rectangle\* p, Rectangle& r) {

02 修改矩形对象v中的边长a,b的数值为10,20;

03 输出修改后的边长数值.

04

05 修改p指向的对象中的边长a,b的数值为10, 20;

06 输出修改后的边长数值.

07

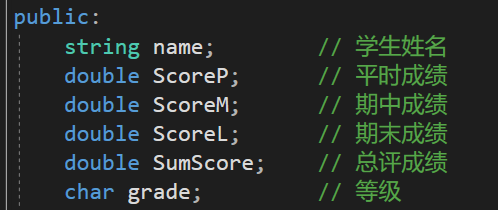
08 修改r引用的对象中的边长a,b的数值为10,20;

09 输出修改后的边长数值.

10 }

1. **实验过程**
2. 任务一: 基础类编程设计实验
3. **设计类**
   1. 类内成员变量

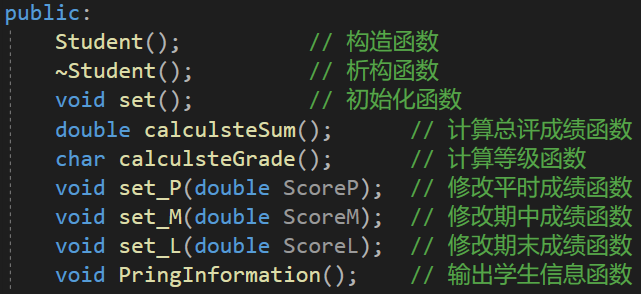
类内成员变量包括学生姓名、平时成绩、期中成绩、期末成绩、总评成绩和等级等（如图1）。其中，学生姓名、平时成绩、期中成绩、期末成绩是用户输入的，而总评成绩和等级是根据公式计算得出的。这些成员变量的属性都是公有属性public。

****

**图1 student类内成员变量**

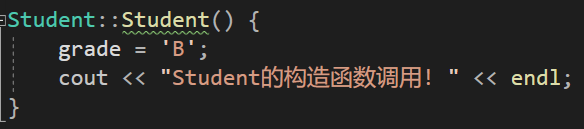
* 1. 类内成员函数

类内成员函数包括常规的构造函数和析构函数，以及初始化函数、计算总评成绩函数、计算等级函数、修改成绩函数和输出学生信息函数（如图2）。

****

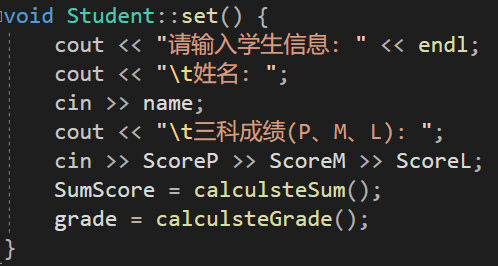
**图2 student类内成员函数**

① 在构造函数中，根据实验要求将等级初始化为B，其他变量等待初始化函数进行初始化（如图3）。

****

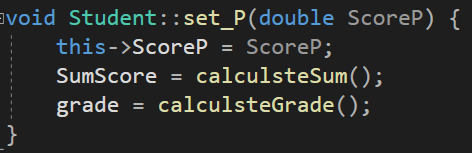
**图3 student类的构造函数**

② 在类内初始化函数中，首先输入学生的基本信息，包括学生姓名、平时成绩、期中成绩、期末成绩，然后利用两个函数calculsteSum( )和calculsteGrade( )计算学生总评成绩和等级（如图4）。

****

**图4 student类的初始化函数**

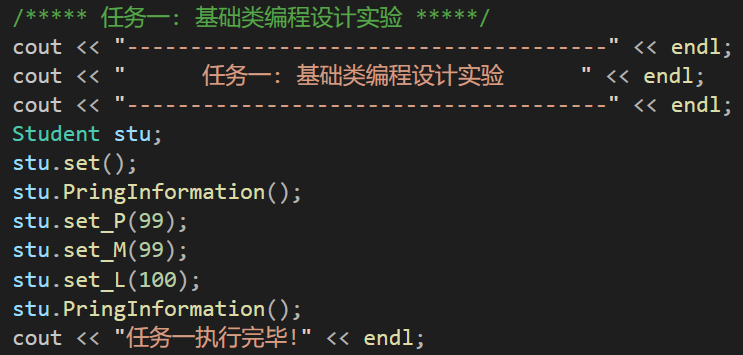
③ 在修改学生信息函数中，使用this指针探究了this指针的本质及作用（如图5）。



**图5 修改学生信息函数 - 探究this指针**

1. **实现类**

在主函数中，分别首先利用初始化函数对对象stu进行初始化，并对当前学生信息进行打印输出；然后利用set\_P( )、set\_M( )、set\_L( )函数修改学生成绩信息，并打印输出修改后的信息（如图6）。

****

**图6 主函数中对student类的实现**

1. 任务二: 类内构造函数设计实验
2. **设计类**
   1. 类内成员变量

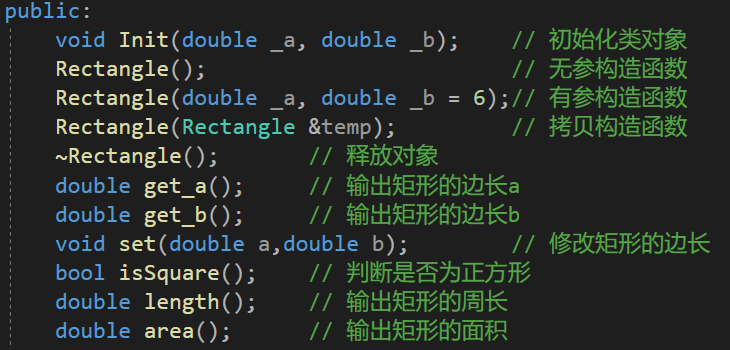
该矩形Rectangle类中，成员变量只有两个，分别为矩形的长和宽（a和b）。根据实验要求，这两个成员变量的属性均为私有属性private（如图7）。

****

**图7 Rectangle类的成员变量**

* 1. 类内成员函数

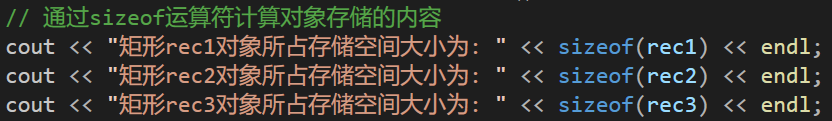
类内成员函数包括初始化函数、构造函数、析构函数、设置矩形边长的函数、修改矩形边长的函数、计算矩形周长的函数和计算矩形面积的函数（如图8）。其中，构造函数又分为无参构造函数、有参构造函数和拷贝构造函数。



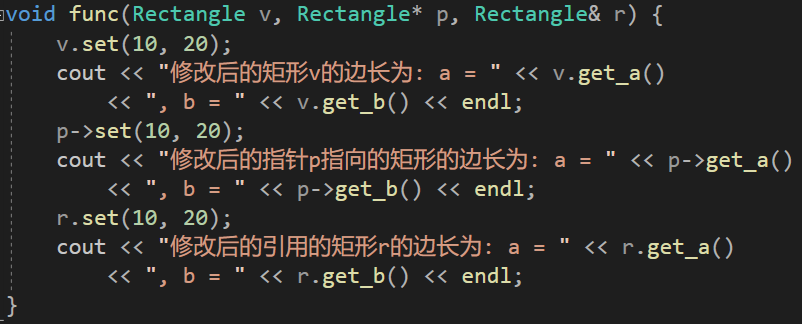
**图8 Rectangle类的成员函数**

1. **实现类**

在主函数中，定义3个矩形，分别用三种不同的构造函数完成构造，并判断这3个矩形是否是正方形以及计算它们的周长和面积。利用sizeof运算符计算类对象存储的内容所占存储空间的大小（图9），并利用程序段void func(Rectangle v, Rectangle\* p, Rectangle& r)分析不同传参方式对实参对象的影响（图10）。

****

**图9 sizeof计算存储空间大小**

****

**图10 func程序段 - 探究传参方式对实参对象的影响**

1. **实验结果**
2. 任务一: 基础类编程设计实验
3. **测试案例**

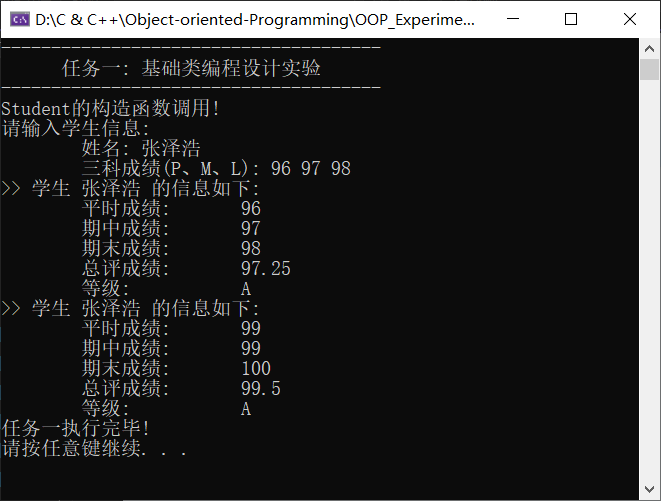
**表1 修改前的student类的测试案例信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | **平时成绩** | **期中成绩** | **期末成绩** | **总评成绩** | **等级** |
| 张泽浩 | 96 | 97 | 98 | 97.25 | A |

**表2 修改后的student类的测试案例信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | **平时成绩** | **期中成绩** | **期末成绩** | **总评成绩** | **等级** |
| 张泽浩 | 99 | 99 | 100 | 99.5 | A |

1. **运行结果**



**图11 任务一的程序运行结果**

1. **探究this指针的本质及作用**

this是C++中的一个关键字，也是一个const指针，它指向当前对象，通过它可以访问当前对象的所有成员。

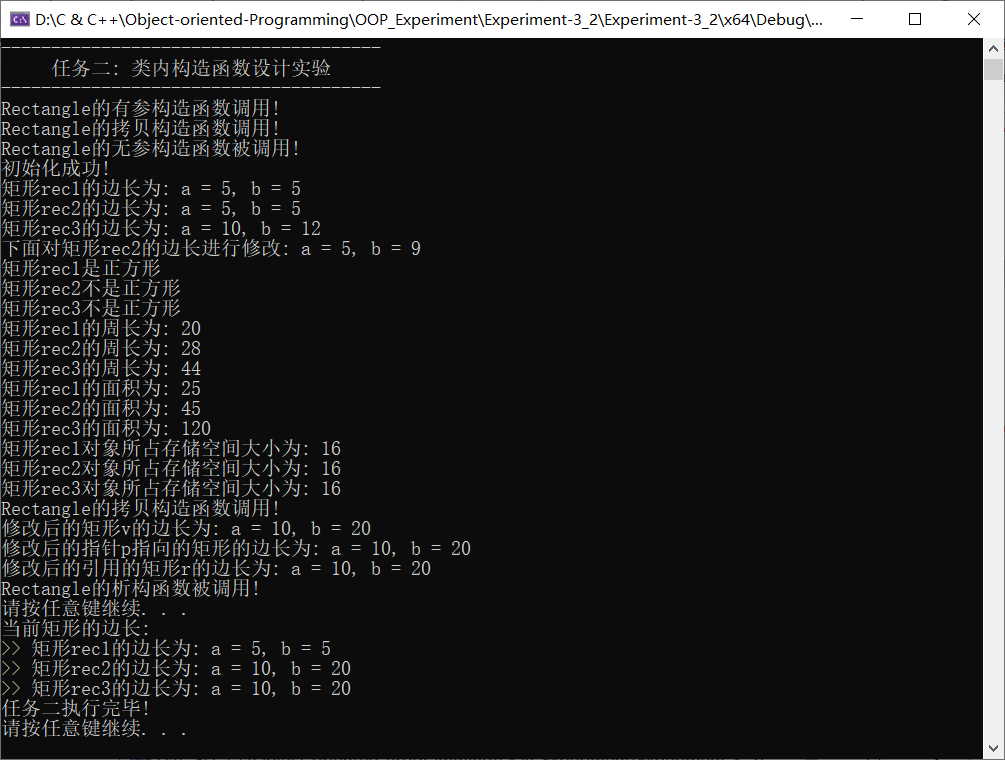
在C++中，每一个对象都能通过this指针来访问自己的地址.this作为隐式形参，本质上是成员函数的局部变量，所以只能用在成员函数的内部，并且只有在通过对象调用成员函数时才给this赋值。

1. 任务二: 类内构造函数设计实验
2. **测试案例**

**表3 Rectangle类的测试案例**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **矩形** | **修改前** | | **修改后** | | **是否为正方形** | **周长** | **面积** |
| **长a** | **宽b** | **长a** | **宽b** |
| rec1 | 5 | 5 | 5 | 5 | 是 | 20 | 25 |
| rec2 | 5 | 5 | 5 | 9 | 否 | 28 | 45 |
| rec3 | 10 | 12 | 10 | 12 | 否 | 44 | 120 |

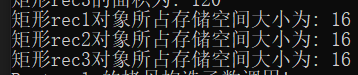
1. **运行结果**

****

**图12 任务二的程序运行结果**

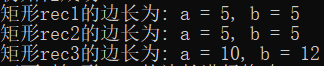
1. **探究对象存储的规律**

通过利用sizeof运算符计算Rectangle类对象所占空间大小为16字节。而类对象中所包含的成员变量为两个double类型的变量，其大小正好为16字节。说明在类中，成员变量占用类的存储空间，而成员函数是不占用类的存储空间的。

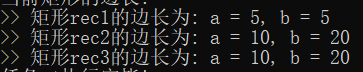
****

**图13 sizeof计算存储空间的运行结果**

1. **探究不同传参方式对实参存储规律的影响**

****

**图14 传参前各矩形边长的数据**

****

**图15 传参后各矩形边长的数据**

基于上图（图14、图15）总结不同传参方式对实参变化的影响，进而总结其对实参存储规律的影响：

(1) 对于值传递，形参和实参是各占一个独立的存储空间的，形参的存储空间是函数被调用时才分配的。

(2) 对于引用传递，是以引用为参数传递的，引用为实参的别名，其形参和实参占用同一个存储空间，形参被改变实参也会随之变化。

(3) 对于地址传递，是以指针为参数传递的，即把实参的地址传递给形参，形参和实参两个指针均指向同一块内存空间，因此形参的改变也会影响实参。

1. **实验总结**

通过本次实验深刻理解了类和对象的基本概念及使用方法，掌握了类的三种构造函数（无参构造函数、有参构造函数、拷贝构造函数）的使用方法，了解了析构函数的作用时间。

通过利用sizeof运算符计算对象的存储内容，了解了类中对象所占内存空间的部分规律；通过再成员函数中使用this指针，掌握了this指针的本质及使用时机（成员变量和形参同名时、返回成员变量时等）；还掌握了值传递、引用传递和地址传递的使用时机。

* **附录：实验源代码（基于Highlight软件粘贴带有行号的源码）**
* 代码托管地址:
* [Object-oriented-Programming/OOP\_Experiment/Experiment-3\_2 at master · keepIHDR/Object-oriented-Programming (github.com)](https://github.com/keepIHDR/Object-oriented-Programming/tree/master/OOP_Experiment/Experiment-3_2)
* **main.cpp**
* 01 #include <iostream>
* 02 #include "rectangle.h"
* 03 #include "student.h"
* 04
* 05 **using namespace** std;
* 06
* 07 /\*
* 08 类的变量记心间, 三构一析放在前
* 09 普通函数按需添, 万法归宗永不变
* 10 \*/
* 11
* 12 **void** func(Rectangle v, Rectangle\* p, Rectangle& r) {
* 13 v.set(10, 20);
* 14 cout << "修改后的矩形v的边长为: a = " << v.
* 15 get\_a()
* 16 << ", b = " << v.get\_b() << endl;
* 17 p->set(10, 20);
* 18 cout << "修改后的指针p指向的矩形的边长为:
* 19 a = " << p->get\_a()
* 20 << ", b = " << p->get\_b() << endl;
* 21 r.set(10, 20);
* 22 cout << "修改后的引用的矩形r的边长为: a = "
* 23 << r.get\_a()
* 24 << ", b = " << r.get\_b() << endl;
* 25 }
* 26
* 27 **int** main() {
* 28
* 29 /\*\*\*\*\* 任务一: 基础类编程设计实验 \*\*\*\*\*/
* 30 cout << "--------------------------------------" << endl;
* 31 cout << " 任务一: 基础类编程设计实验
* 32 " << endl;
* 33 cout << "--------------------------------------" << endl;
* 34 Student stu;
* 35 stu.set();
* 36 stu.PringInformation();
* 37 stu.set\_P(99);
* 38 stu.set\_M(99);
* 39 stu.set\_L(100);
* 40 stu.PringInformation();
* 41 cout << "任务一执行完毕!" << endl;
* 42 system("pause");
* 43 system("cls");
* 44
* 45
* 46 /\*\*\*\*\* 任务二: 类内构造函数设计实验 \*\*\*\*\*\*/
* 47 cout << "--------------------------------------" << endl;
* 48 cout << " 任务二: 类内构造函数设计实验
* 49 " << endl;
* 50 cout << "--------------------------------------" << endl;
* 51 Rectangle rec1(5, 5), rec2(rec1), rec3;
* 52 // 初始化类对象
* 53 rec3.Init(10, 12);
* 54 // 输出矩形的边长
* 55 cout << "矩形rec1的边长为: a = " << rec1.get\_a()
* 56 << ", b = " << rec1.get\_b() << endl;
* 57 cout << "矩形rec2的边长为: a = " << rec2.get\_a()
* 58 << ", b = " << rec2.get\_b() << endl;
* 59 cout << "矩形rec3的边长为: a = " << rec3.get\_a()
* 60 << ", b = " << rec3.get\_b() << endl;
* 61 // 修改矩形的边长
* 62 cout << "下面对矩形rec2的边长进行修改: a = 5,
* 63 b = 9" << endl;
* 64 rec2.set(5, 9);
* 65 // 判断矩形是否为正方形
* 66 cout << "矩形rec1";
* 67 rec1.isSquare() ? cout << "是" : cout << "不是";
* 68 cout << "正方形" << endl;
* 69 cout << "矩形rec2";
* 70 rec2.isSquare() ? cout << "是" : cout << "不是";
* 71 cout << "正方形" << endl;
* 72 cout << "矩形rec3";
* 73 rec3.isSquare() ? cout << "是" : cout << "不是";
* 74 cout << "正方形" << endl;
* 75 // 输出矩形的周长
* 76 cout << "矩形rec1的周长为: " << rec1.length() <<
* 77 endl;
* 78 cout << "矩形rec2的周长为: " << rec2.length() <<
* 79 endl;
* 80 cout << "矩形rec3的周长为: " << rec3.length() <<
* 81 endl;
* 82 // 输出矩形的面积
* 83 cout << "矩形rec1的面积为: " << rec1.area() <<
* 84 endl;
* 85 cout << "矩形rec2的面积为: " << rec2.area() <<
* 86 endl;
* 87 cout << "矩形rec3的面积为: " << rec3.area() <<
* 88 endl;
* 89 // 通过sizeof运算符计算对象存储的内容
* 90 cout << "矩形rec1对象所占存储空间大小为: " <
* 91 < **sizeof**(rec1) << endl;
* 92 cout << "矩形rec2对象所占存储空间大小为: " <
* 93 < **sizeof**(rec2) << endl;
* 94 cout << "矩形rec3对象所占存储空间大小为: " <
* 95 < **sizeof**(rec3) << endl;
* 96 // 探索对象作为参数时的用法
* 97 func(rec1, &rec2, rec3);
* 98 // 输出当前矩形的边长
* 99 cout << "当前矩形的边长:" << endl;
* 100 cout << ">> 矩形rec1的边长为: a = " << rec1.get\_a()
* 101 << ", b = " << rec1.get\_b() << endl;
* 102 cout << ">> 矩形rec2的边长为: a = " << rec2.get\_a()
* 103 << ", b = " << rec2.get\_b() << endl;
* 104 cout << ">> 矩形rec3的边长为: a = " << rec3.get\_a()
* 105 << ", b = " << rec3.get\_b() << endl;
* 106
* 107 cout << "任务二执行完毕!" << endl;
* 108 system("pause");
* 109 system("cls");
* 110 cout << "析构函数执行: " << endl;
* 111 **return** 0;
* 112 }
* **student.h**
* 01 #pragma once
* 02 #include <cstring>
* 03
* 04 **using namespace** std;
* 05
* 06 class Student {
* 07 public:
* 08 string name; // 学生姓名
* 09 **double** ScoreP; // 平时成绩
* 10 **double** ScoreM; // 期中成绩
* 11 **double** ScoreL; // 期末成绩
* 12 **double** SumScore; // 总评成绩
* 13 **char** grade; // 等级
* 14 public:
* 15 Student(); // 构造函数
* 16 ~Student(); // 析构函数
* 17 **void** set(); // 初始化函数
* 18 **double** calculsteSum(); // 计算总评成绩函数
* 19 **char** calculsteGrade(); // 计算等级函数
* 20 **void** set\_P(**double** ScoreP); // 修改平时成绩函数
* 21 **void** set\_M(**double** ScoreM); // 修改期中成绩函数
* 22 **void** set\_L(**double** ScoreL); // 修改期末成绩函数
* 23 **void** PringInformation(); // 输出学生信息函数
* 24 };
* **student.cpp**
* 01 #include <iostream>
* 02 #include "student.h"
* 03
* 04 **using namespace** std;
* 05
* 06 Student::Student() {
* 07 grade = 'B';
* 08 cout << "Student的构造函数调用! " << endl;
* 09 }
* 10
* 11 Student::~Student() {
* 12 cout << "Student的析构函数调用! " << endl;
* 13 system("pause");
* 14 }
* 15
* 16 **void** Student::set() {
* 17 cout << "请输入学生信息: " << endl;
* 18 cout << "\t姓名: ";
* 19 cin >> name;
* 20 cout << "\t三科成绩(P、M、L): ";
* 21 cin >> ScoreP >> ScoreM >> ScoreL;
* 22 SumScore = calculsteSum();
* 23 grade = calculsteGrade();
* 24 }
* 25
* 26 **double** Student::calculsteSum() {
* 27 **return** ScoreP \* 0.25 + ScoreM \* 0.25 + ScoreL \* 0.5;
* 28 }
* 29
* 30 **char** Student::calculsteGrade() {
* 31 **if** (SumScore >= 90) {
* 32 **return** 'A';
* 33 } **else if** (SumScore >= 80) {
* 34 **return** 'B';
* 35 } **else if** (SumScore >= 70) {
* 36 **return** 'C';
* 37 } **else if** (SumScore >= 60) {
* 38 **return** 'D';
* 39 } **else** {
* 40 **return** 'E';
* 41 }
* 42 }
* 43
* 44 **void** Student::set\_P(**double** ScoreP) {
* 45 **this**->ScoreP = ScoreP;
* 46 SumScore = calculsteSum();
* 47 grade = calculsteGrade();
* 48 }
* 49
* 50 **void** Student::set\_M(**double** ScoreM) {
* 51 **this**->ScoreM = ScoreM;
* 52 SumScore = calculsteSum();
* 53 grade = calculsteGrade();
* 54 }
* 55
* 56 **void** Student::set\_L(**double** ScoreL) {
* 57 **this**->ScoreL = ScoreL;
* 58 SumScore = calculsteSum();
* 59 grade = calculsteGrade();
* 60 }
* 61
* 62 **void** Student::PringInformation() {
* 63 cout << ">> 学生 " << name << " 的信息如下:" <<
* 64 endl;
* 65 cout << "\t平时成绩:\t" << ScoreP << endl;
* 66 cout << "\t期中成绩:\t" << ScoreM << endl;
* 67 cout << "\t期末成绩:\t" << ScoreL << endl;
* 68 cout << "\t总评成绩:\t" << SumScore << endl;
* 69 cout << "\t等级:\t\t" << grade << endl;
* 70 }
* **rectangle.h**
* 01 #pragma once
* 02
* 03 /\*
* 04 类的变量记心间, 三构一析放在前
* 05 普通函数按需添, 万法归宗永不变
* 06 \*/
* 07
* 08 class Rectangle {
* 09 private:
* 10 **double** a, b; // 私有数据成员变量
* 11 // 对外提供的接口
* 12 public:
* 13 **void** Init(**double** \_a, **double** \_b); // 初始化类对象
* 14 Rectangle(); // 无参构造函数
* 15 Rectangle(**double** \_a, **double** \_b = 6);//
* 16 有参构造函数
* 17 Rectangle(Rectangle &temp); // 拷贝构造函数
* 18 ~Rectangle(); // 释放对象
* 19 **double** get\_a(); // 输出矩形的边长a
* 20 **double** get\_b(); // 输出矩形的边长b
* 21 **void** set(**double** a,**double** b); // 修改矩形的边长
* 22 **bool** isSquare(); // 判断是否为正方形
* 23 **double** length(); // 输出矩形的周长
* 24 **double** area(); // 输出矩形的面积
* 25 };
* **Rectangle.cpp**
* 01 #include <iostream>
* 02 #include "rectangle.h"
* 03 **using namespace** std;
* 04
* 05 **void** Rectangle::Init(**double** \_a, **double** \_b) {
* 06 a = \_a;
* 07 b = \_b;
* 08 cout << "初始化成功! " << endl;
* 09 }
* 10
* 11 Rectangle::Rectangle() {
* 12 cout << "Rectangle的无参构造函数被调用! " <<
* 13 endl;
* 14 }
* 15
* 16 Rectangle::Rectangle(**double** \_a, **double** \_b) {
* 17 a = \_a;
* 18 b = \_b;
* 19 cout << "Rectangle的有参构造函数调用! " << endl;
* 20 }
* 21
* 22 Rectangle::Rectangle(Rectangle& temp) {
* 23 a = temp.get\_a();
* 24 b = temp.get\_b();
* 25 cout << "Rectangle的拷贝构造函数调用! " << endl;
* 26 }
* 27
* 28 Rectangle::~Rectangle() {
* 29 cout << "Rectangle的析构函数被调用! " << endl;
* 30 system("pause");
* 31 }
* 32
* 33 **double** Rectangle::get\_a() {
* 34 **return** a;
* 35 }
* 36
* 37 **double** Rectangle::get\_b() {
* 38 **return** b;
* 39 }
* 40
* 41 **void** Rectangle::set(**double** \_a,**double** \_b) {
* 42 a = \_a;
* 43 b = \_b;
* 44 }
* 45
* 46 **bool** Rectangle::isSquare() {
* 47 **if** (a == b) {
* 48 **return true**;
* 49 }
* 50 **return false**;
* 51 }
* 52
* 53 **double** Rectangle::length() {
* 54 **return** 2 \* (a + b);
* 55 }
* 56
* 57 **double** Rectangle::area() {
* 58 **return** a \* b;
* 59 }