|  |  |
| --- | --- |
| **编号:** | **4-1** |

****

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25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. 熟练掌握复杂类的设计方法（三构一析+普函）
3. 熟练掌握静态数据成员和函数的声明、定义及用法（static）
4. 掌握常数据，常函数和常对象的声明、定义及用法（const）
5. 掌握友元函数的声明、定义和使用方法（friend）
6. **实验内容**
7. **任务一：简答题**
8. 什么是友元，在类中为何要设计友元，它包括哪些种类？
9. 什么是静态成员数据，其本质和主要作用是什么？
10. 什么是静态成员函数，在类中为何要设计该函数？
11. 简述静态数据成员与普通数据成员的区别与联系.
12. **任务二：基本程序设计**

请设计一个点类Point，类内包含点的二维坐标数值，通过该类实现点坐标的初始化，坐标修改，坐标位置输出功能；然后基于该Point类设计一个友元函数double distance(Point &A, Point &B), 实现计算两个点对象之间的直线距离。

1. **任务三：综合程序设计**

请采用C++面向对象程序设计思想设计类，并实现类。要求程序代码：清晰简洁易读，良好的开闭性，编程风格规范统一。

设计某高校研究生Graduate类，该类包含研究生姓名(string name)、性别(bool gender)、学号(int id)、入学成绩(double score)、研究领域(string research)、指导教师(string tutor)和学校名称(const string university)与该研究生类有关的函数如下：

初始化与释放研究生对象数据（三构一析）；

采用普通成员函数形式输出研究生基本数据信息；

采用常成员函数形式输出研究生的基本数据信息；

输出多个研究生对象的平均成绩(要求在类内封装实现)；

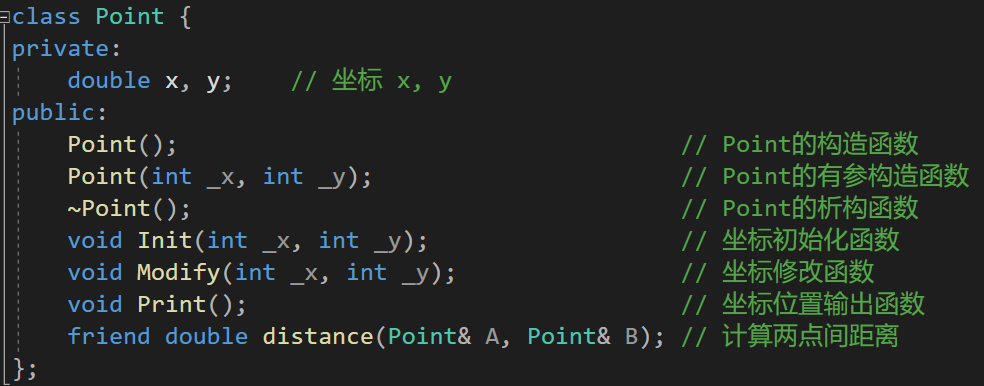
以友元函数形式计算每个研究生对象的成绩等级（90分以上输出A，80-89分输出B，70-79分输出C,60-69分输出D，60分以下输出E)；

提示: string是C++系统内置的一个类，可以直接使用，功能强大，使用前带上头文件<string>

1. **实验过程**
2. **任务一：简答题**

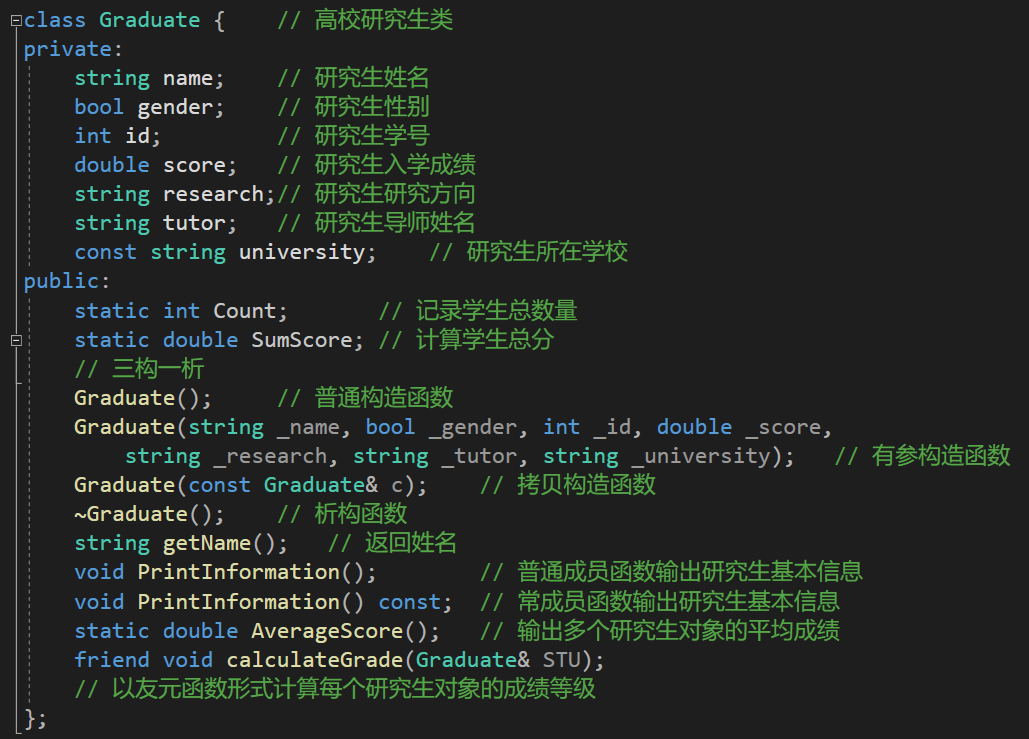
复习课本及相关PPT，对本问题做出解答。

1. **任务二：基本程序设计**
2. 设计Point类成员变量：私有成员变量x、y，分别表示点的横纵坐标（如图1）。
3. 设计Point类成员函数：
   1. 公有属性：普通构造函数Point()、有参构造函数Point(int \_x, int \_y)、析构函数~Point()、坐标初始化函数void Init(int \_x, int \_y)、坐标修改函数void Modify(int \_x, int \_y)、坐标位置输出函数void Print().（如图1）
   2. 友元：计算两点间距离函数friend double distance(Point& A, Point& B)（如图1）

****

**图1 Point类成员变量及成员函数**

1. **任务三：综合程序设计**
2. 设计Graduate类成员变量：
3. 所有与学生基本信息相关的成员变量的属性均为私有属性（private）。包括研究生姓名string name、性别bool gender、学号int id、入学成绩double score、研究领域string research、指导教师string tutor和学校名称const string university。其中，学校名称不可更改，故设计为常成员变量（如图2）。
4. 设计两个公有属性（public）的静态成员变量static int Count和static double SumScore，用于保存学生总数量及学生总分，从而计算学生的平均成绩（如图2）。
5. 设计Graduate类成员函数：
6. 该类的成员函数均为公有属性（public），供主函数调用。包括“三构一析”（普通构造函数、有参构造函数、拷贝构造函数、析构函数）、输出研究生基本信息函数、静态成员函数输出多个研究生的平均成绩static double AverageScore()以及友元函数计算研究生对象的等级friend void calculateGrade(Graduate& STU)（如图2）。
7. 成员变量的初始化：对于静态成员变量的初始化应该在类外进行，对于常成员变量的初始化应该在构造函数中借助“:”运算符进行初始化，对于其他普通成员变量应在构造函数中初始化。

****

**图2 Graduate类成员变量及成员函数**

1. **实验结果**
2. **任务一：简答题**
3. **什么是友元，在类中为何要设计友元，它包括哪些种类？**
4. 友元的概念：友元是一种在类内声明、定义在类外部的普通函数或类。声明时，需要前面加关键字friend。
5. 为何要设计友元：可以不用调用类的接口函数而直接访问本类的所有私有成员，进而降低了频繁调用类接口函数导致的内存销(空间)，提高了程序的运行速度(时间)。
6. 友元的种类：友元函数、友元成员、友元类
7. **什么是静态数据成员，其本质和主要作用是什么？**
8. 静态数据成员：类体中的数据成员的声明前加上static关键字，该数据成员就成为了该类的静态数据成员。
9. 静态成员数据的本质：本质上为全局变量，被类的对象共享。
10. 静态成员数据的作用：解决同类对象间数据共享问题，同一个类所有的对象共享此成员，该成员只有一分拷贝。将其放置在类内，具有良好的安全性和封装特性。
11. **什么是静态成员函数，在类中为何要设计该函数？**
12. 静态成员函数：类体中的成员函数的声明前加上static关键字，该成员函数就成为了该类的静态成员函数。
13. 静态成员函数的作用：作为类与静态成员变量之间的桥梁，通过这个函数完成对全局变量的管理。
14. **简述静态数据成员与普通数据成员的区别与联系？**
15. 区别：

① 声明时，静态数据成员与普通数据成员相比，在最前面加上了static关键字。

② 普通数据成员属于类的一个具体的对象，只有对象被创建了，普通数据成员才会被分配内存。而静态数据成员属于整个类，即使没有任何对象创建，类的静态数据成员变量也存在。

③ 类的静态成员函数无法直接访问普通数据成员（可以通过对象名间接的访问），而类的任何成员函数都可以访问类的静态数据成员。

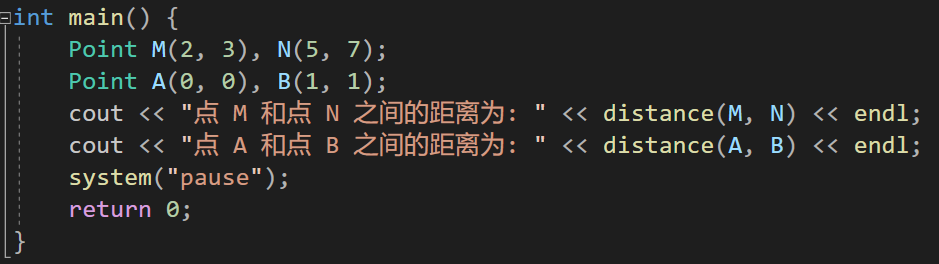
1. 联系：静态成员和类的普通成员一样，也具有public、protected、private3种访问级别，也可以具有返回值、const修饰符等参数。
2. **任务二：基本程序设计**
3. 测试案例

本程序选用四个点对程序进行测试，点的信息如下表（表1）。

**表1 测试点信息**

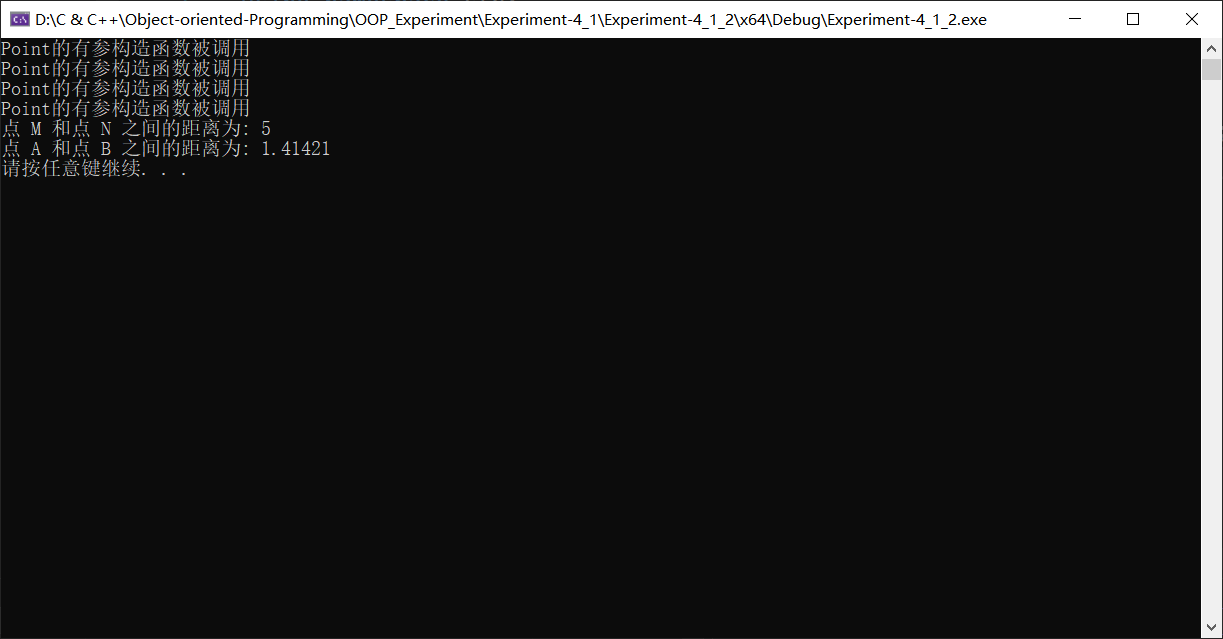
|  |  |  |
| --- | --- | --- |
| 点 | 横坐标x | 纵坐标y |
| A | 0 | 0 |
| B | 1 | 1 |
| M | 2 | 3 |
| N | 5 | 7 |

1. 运行结果
   1. 主函数：

****

**图3 主函数代码截图**

* 1. 程序运行结果截图：

****

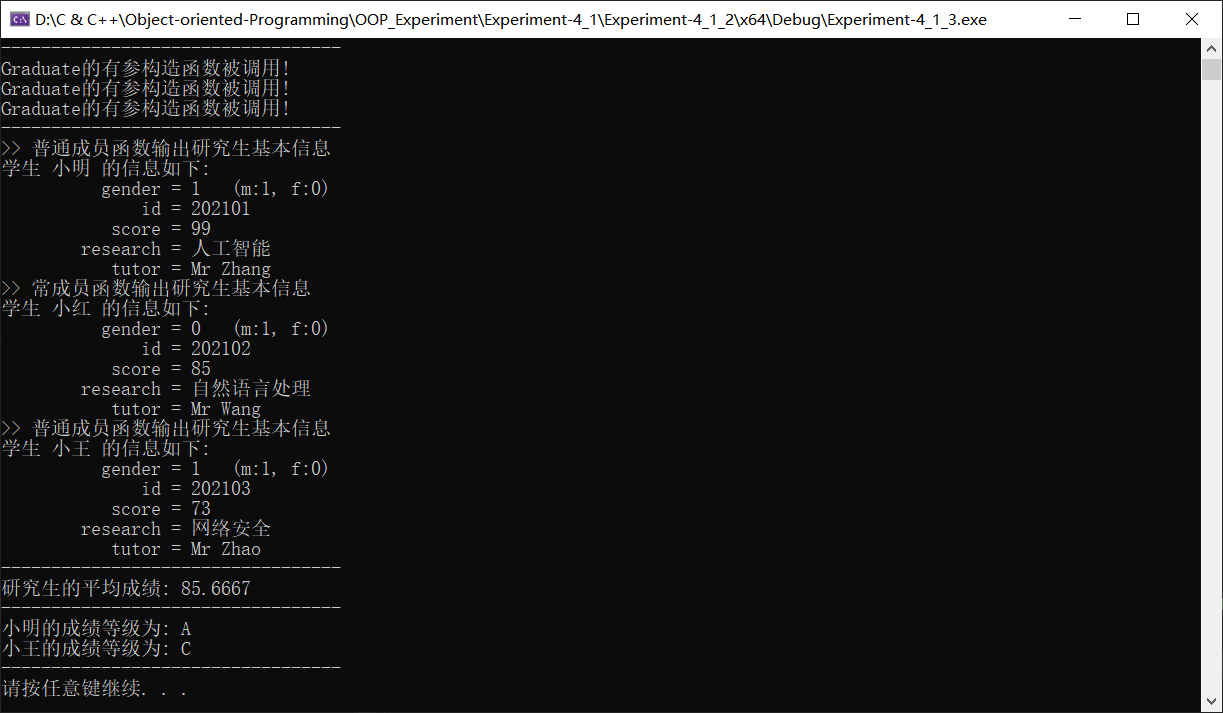
**图4 任务二程序运行结果截图**

1. **任务三：综合程序设计**
2. 测试案例

**表2 研究型信息测试案例**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 性别 | 学号 | 入学成绩 | 研究方向 | 导师 | 学校 |
| 小明 | 男 | 202101 | 99 | 人工智能 | Mr Zhang | SDNU |
| 小红 | 女 | 202102 | 85 | 自然语言处理 | Mr Wang | SDNU |
| 小王 | 男 | 202103 | 73 | 网络安全 | Mr Zhao | SDNU |

1. 程序运行结果

****

**图5 程序运行结果截图**

1. **实验总结**

通过本次实验，掌握了类中静态成员变量、静态成员函数及友元的使用方法。对于友元，了解了其概念和作用，以及如何定义、声明和使用；对于静态成员，了解了其本质及作用，掌握了如何定义、声明和使用静态成员变量和静态成员函数，且静态成员变量要在类内定义、类外初始化；对于常函数和常对象，了解了其“一一对应”的关系。

在本次实验中，还巩固了类的设计方法，以及如何在三种构造函数（普通构造函数、有参构造函数、拷贝构造函数）中初始化常数据。

* **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-4\_1 at master · keepIHDR/Object-oriented-Programming (github.com)](https://github.com/keepIHDR/Object-oriented-Programming/tree/master/OOP_Experiment/Experiment-4_1)
* 任务二：基本程序设计
* point.h
* 01 #pragma once
* 02 class Point {
* 03 private:
* 04 double x, y;
* 05 public:
* 06 Point(); // Point的构造函数
* 07 Point(int \_x, int \_y); // Point的有参构造函数
* 08 ~Point(); // Point的析构函数
* 09 void Init(int \_x, int \_y); // 坐标初始化函数
* 10 void Modify(int \_x, int \_y);// 坐标修改函数
* 11 void Print(); // 坐标位置输出函数
* 12 friend double distance(Point& A, Point& B);
* 13 };
* 14
* 15 Point::Point() {
* 16 cout << "Point的构造函数被调用" << endl;
* 17 }
* 18
* 19 Point::Point(int \_x, int \_y) {
* 20 x = \_x;
* 21 y = \_y;
* 22 cout << "Point的有参构造函数被调用" << endl;
* 23 }
* 24
* 25 Point::~Point() {
* 26 cout << "Point的析构函数被调用" << endl;
* 27 }
* 28
* 29 void Point::Init(int \_x, int \_y) {
* 30 x = \_x;
* 31 y = \_y;
* 32 cout << "初始化成功! " << endl;
* 33 }
* 34
* 35 void Point::Modify(int \_x, int \_y) {
* 36 x = \_x;
* 37 y = \_y;
* 38 cout << "修改成功! " << endl;
* 39 }
* 40
* 41 void Point::Print() {
* 42 cout << "该点的坐标位置为: (" << x << " ," << y <
* 43 < " )" << endl;
* 44 }
* main.cpp
* 01 #include<iostream>
* 02 #include<cmath>
* 03 using namespace std;
* 04 #include"point.h"
* 05
* 06 double distance(Point& A, Point& B) {
* 07 double ans = (A.x - B.x)\*(A.x - B.x) + (A.y - B.y) \* (A.
* 08 y - B.y);
* 09 return sqrt(ans);
* 10 }
* 11
* 12 int main() {
* 13 Point M(2, 3), N(5, 7);
* 14 cout << "点 M 和点 N 之间的距离为: " <<
* 15 distance(M, N) << endl;
* 16 system("pause");
* 17 return 0;
* 18 }
* 任务二：综合程序设计
* graduate.h
* 01 #pragma once
* 02 #include<string>
* 03 #include<iostream>
* 04 using namespace std;
* 05
* 06 class Graduate { // 高校研究生类
* 07 private:
* 08 string name; // 研究生姓名
* 09 bool gender; // 研究生性别
* 10 int id; // 研究生学号
* 11 double score; // 研究生入学成绩
* 12 string research;// 研究生研究方向
* 13 string tutor; // 研究生导师姓名
* 14 const string university; // 研究生所在学校
* 15 public:
* 16 static int Count; // 记录学生总数量
* 17 static double SumScore; // 计算学生总分
* 18 // 三构一析
* 19 Graduate(); // 普通构造函数
* 20 Graduate(string \_name, bool \_gender, int \_id, double
* 21 \_score,
* 22 string \_research, string \_tutor, string
* 23 \_university); // 有参构造函数
* 24 Graduate(const Graduate& c); // 拷贝构造函数
* 25 ~Graduate(); // 析构函数
* 26 string getName(); // 返回姓名
* 27 void PrintInformation(); //
* 28 普通成员函数输出研究生基本信息
* 29 void PrintInformation() const; //
* 30 常成员函数输出研究生基本信息
* 31 static double AverageScore(); //
* 32 输出多个研究生对象的平均成绩
* 33 friend void calculateGrade(Graduate& STU);
* 34 //
* 35 以友元函数形式计算每个研究生对象的成绩
* 36 等级
* 37 };
* 38
* 39 Graduate::Graduate():university("SDNU") {
* 40 cout << "Graduate的普通构造函数被调用!" <<
* 41 endl;
* 42 cout << "请输入学生信息:" << endl;
* 43 cout << " name = ";
* 44 cin >> name;
* 45 cout << " gender(m:1, f:0) = ";
* 46 cin >> gender;
* 47 cout << " id = ";
* 48 cin >> id;
* 49 cout << " score = ";
* 50 cin >> score;
* 51 cout << " research = ";
* 52 cin >> research;
* 53 cout << " tutor = ";
* 54 cin >> tutor;
* 55 Count++;
* 56 SumScore += this->score;
* 57 }
* 58
* 59 int Graduate::Count = 0;
* 60 double Graduate::SumScore = 0.0;
* 61
* 62 Graduate::Graduate(string \_name, bool \_gender, int \_id,
* 63 double \_score, string \_research, string \_tutor, string
* 64 \_university):university(\_university) {
* 65 cout << "Graduate的有参构造函数被调用!" <<
* 66 endl;
* 67 this->name = \_name;
* 68 this->gender = \_gender;
* 69 this->id = \_id;
* 70 this->score = \_score;
* 71 this->research = \_research;
* 72 this->tutor = \_tutor;
* 73 Count++;
* 74 SumScore += this->score;
* 75 }
* 76
* 77 Graduate::Graduate(const Graduate& c) {
* 78 cout << "Graduate的拷贝构造函数被调用!" <<
* 79 endl;
* 80 this->name = c.name;
* 81 this->gender = c.gender;
* 82 this->id = c.id;
* 83 this->score = c.score;
* 84 this->research = c.research;
* 85 this->tutor = c.tutor;
* 86 Count++;
* 87 SumScore += this->score;
* 88 }
* 89
* 90 Graduate::~Graduate() {
* 91 cout << "Graduate的析构函数被调用" << endl;
* 92 }
* 93
* 94 string Graduate::getName() {
* 95 return this->name;
* 96 }
* 97
* 98 void Graduate::PrintInformation() {
* 99 cout << ">>
* 100 普通成员函数输出研究生基本信息 " << endl;
* 101 cout << "学生 " << this->name << " 的信息如下:" <<
* 102 endl;
* 103 cout << "\t gender = " << this->gender << " (m:1, f:0)
* 104 " << endl;
* 105 cout << "\t id = " << this->id << endl;
* 106 cout << "\t score = " << this->score << endl;
* 107 cout << "\tresearch = " << this->research << endl;
* 108 cout << "\t tutor = " << this->tutor << endl;
* 109 }
* 110
* 111 void Graduate::PrintInformation() const {
* 112 cout << ">> 常成员函数输出研究生基本信息 "
* 113 << endl;
* 114 cout << "学生 " << this->name << " 的信息如下:" <<
* 115 endl;
* 116 cout << "\t gender = " << this->gender << " (m:1, f:0)
* 117 " << endl;
* 118 cout << "\t id = " << this->id << endl;
* 119 cout << "\t score = " << this->score << endl;
* 120 cout << "\tresearch = " << this->research << endl;
* 121 cout << "\t tutor = " << this->tutor << endl;
* 122 }
* 123
* 124 double Graduate::AverageScore() {
* 125 return double(SumScore / (Count \* 1.0));
* 126 }
* 127
* main.cpp
* 01 #include<iostream>
* 02 #include<string>
* 03 using namespace std;
* 04 #include"graduate.h"
* 05
* 06 void calculateGrade(Graduate& STU) {
* 07 cout << STU.getName() << "的成绩等级为: ";
* 08 if (STU.score >= 90) {
* 09 cout << "A";
* 10 } else if (STU.score >= 80) {
* 11 cout << "B";
* 12 } else if (STU.score >= 70) {
* 13 cout << "C";
* 14 } else if (STU.score >= 60) {
* 15 cout << "D";
* 16 } else {
* 17 cout << "E";
* 18 }
* 19 cout << endl;
* 20 }
* 21
* 22 int main() {
* 23 cout << "----------------------------------" << endl;
* 24 Graduate A("小明", 1, 202101, 99, "人工智能", "Mr
* 25 Zhang", "SDNU");
* 26 const Graduate B("小红", 0, 202102, 85,
* 27 "自然语言处理", "Mr Wang", "SDNU");
* 28 Graduate C("小王", 1, 202103, 73, "网络安全", "Mr
* 29 Zhao", "SDNU");
* 30 cout << "----------------------------------" << endl;
* 31 A.PrintInformation();
* 32 B.PrintInformation();
* 33 C.PrintInformation();
* 34 cout << "----------------------------------" << endl;
* 35 cout << "研究生的平均成绩: " << Graduate::
* 36 AverageScore() << endl;
* 37 cout << "----------------------------------" << endl;
* 38 calculateGrade(A);
* 39 calculateGrade(C);
* 40 cout << "----------------------------------" << endl;
* 41 system("pause");
* 42 return 0;
* 43 }