**实验五 类与对象**

**【实验目的与要求】**

1、掌握声明类的方法，类和类的成员的概念以及定义对象的方法。

2、初步掌握用类和对象编制基于对象的程序。

3、学习检查和调试基于对象的程序。

**【实验内容】**

1、有以下程序：

#include<iostream>

using namespace std；

class Time // 定义Time类

{

public： // 数据成员为公用的

int hour;

int minute;

int sec ;

};

int main()

{

Time tl; //定义t1为Time类对象

cin>>t1.hour; //输入设定的时间

cin>>t1.minute;

cin>>t1.sec;

cout<<t1.hourl<<”：”<<t1.minute<<”：”<<t1.sec<<endl ;

return 0;

}

改写程序，要求：

(1)将数据成员改为私有的；

(2)将输入和输出的功能改为由成员函数实现；

(3)在类体内定义成员函数。

然后编译和运行程序。请分析什么成员应指定为公用的?什么成员应指定为私有的? 什么函数最好放在类中定义? 什么函数最好在类外定义?

2、分别给出如下的3个文件：

(1)含类定义的头文件student.h，

//student.h (这是头文件，在此文件中进行类的声明)

class Student //类声明

{

public: //公用成员函数原型声明

void display();

private:

int num;

char name[20];

char sex;

};

(2)包含成员函数定义的源文件student.cpp

//student.cpp 在此文件中进行函数的定义

#include <iostream>

#include”student.h” //不要漏写此行，否则编译通不过

void Student::display() //在类外定义display类函数

{

cout<<”num：”<<num<<endl;

cout<<”name：”<<name<<endl;

cout<<”sex：”<<sex<<endl;

}

(3)包含主函数的源文件main.cpp。

为了组成一个完整的源程序，应当有包括主函数的源文件：

//main.cpp 主函数模块

#include <iostream> //将类声明头文件包含进来

#include “student.h”

int main()

{

Student stud; //定义对象

Student stud1(007,”tcg”,’m’);

stud.display(); //执行stud对象的display函数

return 0;

}

请完善该程序，在类中增加一个对数据成员赋初值的成员函数set\_value。上机调试并运行。

3、需要求3个长方柱的体积，请编一个基于对象的程序。数据成员包括length(长)、width(宽)、height(高)。要求用成员函数实现以下功能：

(1)由键盘分别输入3个长方柱的长、宽、高；

(2)计算长方柱的体积；

(3)输出3个长方柱的体积。

请编程序，上机调试并运行。

4、建立一个对象数组，内放5个学生的（学号，成绩），设立一个函数max，用指向对象的指针作函数参数，在max函数中找出5个学生的最高成绩者，并输出其学号。

1. 设有一描述坐标点的类Point，其私有变量x和y代表一个点的(x,y)坐标值。请编写程序实现以下功能：利用构造函数传递参数，在定义对象时将x、y坐标值初始化为（60,80）；利用公有成员函数void setPoint(int i, int j)将坐标值修改为(60+i,80+j)；利用公有成员函数display()输出修改后的坐标值。主函数中通过定义对象，验证各个函数。

【程序设计部分】

1. 改写后的代码如下:

#include <iostream>

using namespace std;

class Time {

private: // 数据成员改为私有

int hour;

int minute;

int sec;

public:

// 设置成员函数，用于输入时间

void inputtime() {

cout << "请输入时间（小时 分钟 秒）：";

cin >> hour >> minute >> sec;

}

// 设置成员函数，用于输出时间

void output() {

cout << hour << "：" << minute << "：" << sec << endl;

}

};

int main() {

Time t1; // 创建Time类对象

t1.inputtime(); // 调用成员函数进行输入

t1.output(); // 调用成员函数进行输出

return 0;

}

运行结果截图：



Q1:什么成员应指定为公用的？

通常，数据成员应该是私有的（private），以保护它们不被随意修改。只有当需要从外部访问时，才将成员函数设置为公有（public）。因此，在本题中类中的数据成员（ hour, minute, sec）应该是私有的，操作这些数据的函数应该是公有的。

Q2: 什么成员应指定为私有的？

一般来说，数据成员及一些不可修改的变量一般应该是私有的。这样可以封装数据，避免外部代码直接修改数据，使得类的实现细节对外界不可见，在类外才不能够直接访问到。

Q3：什么函数最好放在类中定义？

对于简短的、直接使用的成员函数，如 input() 和 output()，可以在类内定义。这是因为这些函数通常不复杂，且频繁地使用。

Q4：什么函数最好在类外定义?

**第一种情况为函数逻辑复杂的时候**当函数需要较复杂的逻辑或多条语句时，可以将其定义在类外。例如，涉及多个条件判断、循环或递归的函数。或者**函数较长的时候**如果函数体很长，或者涉及复杂的计算和处理，那么最好将其实现放在类外部，以避免使类体变得庞大且难以维护。

2.

代码如下：

// student.cpp

#include <iostream>

#include "student.h" // 包含类声明

using namespace std;

// 构造函数定义

Student::Student(int n, const char\* nm, char s)

{

num = n;

strncpy(name, nm, sizeof(name) - 1); // 防止缓冲区溢出

name[sizeof(name) - 1] = '\0'; // 确保字符串以'\0'结尾

sex = s;

}

// display 函数定义

void Student::display()

{

cout << "num: " << num << endl;

cout << "name: " << name << endl;

cout << "sex: " << sex << endl;

}

// student.h

#ifndef STUDENT\_H // 防止重复包含

#define STUDENT\_H

class Student

{

public:

// 构造函数声明

Student(int n = 0, const char\* nm = "", char s = 'm');

// 公用成员函数声明

void display();

private:

int num;

char name[20];

char sex;

};

#endif

// main.cpp

#include <iostream>

#include "student.h"

int main()

{

// 使用无参构造函数创建对象

Student stud;

stud.display(); // 调用 display 方法

// 使用有参构造函数创建对象

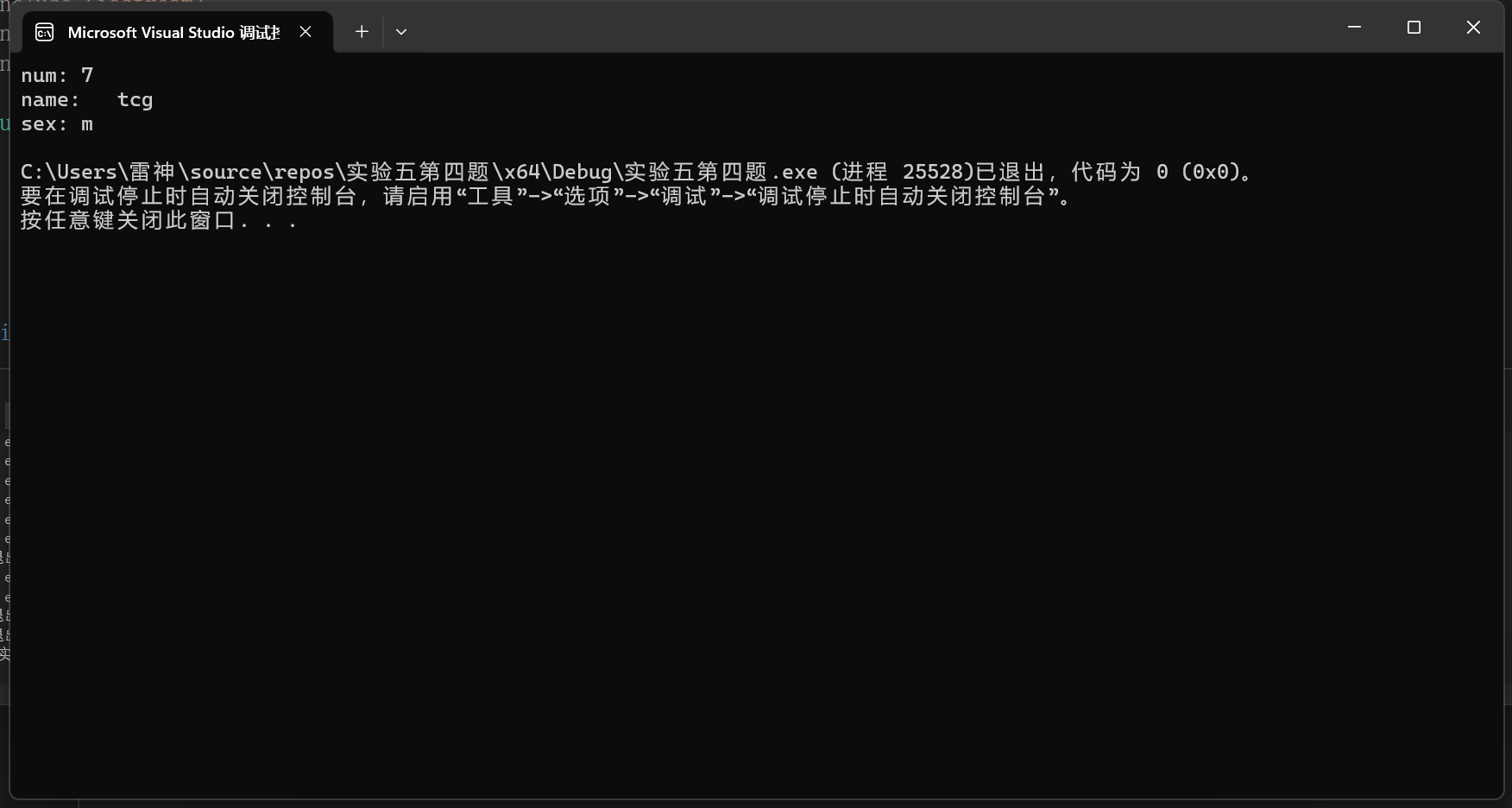
Student stud1(007, "tcg", 'm');

stud1.display(); // 调用 display 方法

return 0;

}

运行结果截图：



3.

代码如下：

#include <iostream>

using namespace std;

// 设置一个长方体的类

class Cuboid {

public:

double length; // 定义长方体的长

double width; // 定义长方体的宽

double height; // 定义长方体的高

double volume; // 存储体积

// 设置一个输入函数

void inputcuboid() {

cin >> length >> width >> height;

}

// 计算体积的函数

void caculatevolume() {

volume = length \* width \* height;

}

// 输出计算所得的体积

void showvolume() {

cout << "长方体的体积为 " << volume << "立方米" << endl;

}

};

int main() {

Cuboid cuboids[3]; // 创建一个包含3个长方体的数组

// 输入每个长方体的长宽高

for (int i = 0; i < 3; i++) {

cout << "请输入第 " << i + 1 << " 个长方体的长、宽、高：";

cuboids[i].inputcuboid();

cuboids[i].caculatevolume(); // 计算体积

}

// 输出每个长方体的体积

for (int i = 0; i < 3; i++) {

cuboids[i].showvolume();

}

return 0;

}

运行结果截图：



4.

代码如下：

#include <iostream>

using namespace std;

// 定义学生类

class Student {

public:

double score; // 学生成绩

int number; // 学生学号

};

// 定义 max 函数,通过指针指向对象名称的方式来引入函数参数

void max(Student\* student, int size)

{

int maxIndex = 0;

// 查找最高分学生

for (int i = 1; i < size; i++) {

if (student[i].score > student[maxIndex].score) {

maxIndex = i;

}

}

// 输出最高分学生的学号和成绩

cout << "成绩最高的学生学号是: " << student[maxIndex].number

<< "，成绩是: " << student[maxIndex].score << endl;

}

int main() {

// 定义5个学生对象并初始化

Student students[5] =

{

{85.5, 2401},

{92.3, 2402},

{78.9, 2403},

{88.7, 2404},

{91.2, 2405}

};

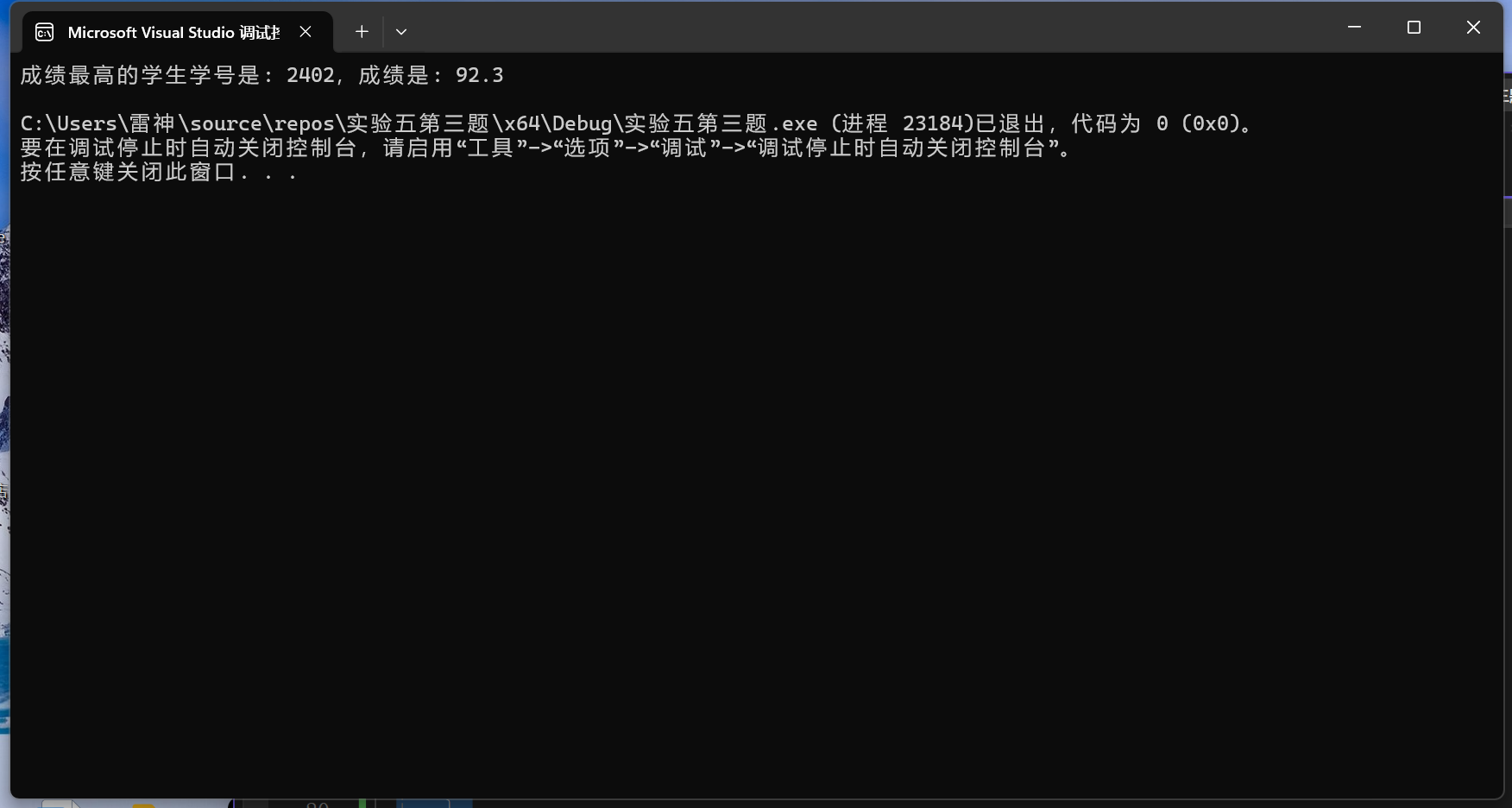
// 调用 max 函数

max(students, 5);

return 0;

}

运行结果截图：



5.

代码如下：

#include<iostream>

using namespace std;

//建立一个点类

class Point

{

public:

//建立一个构造函数给x与y进行赋值

Point()

{

x = 60;

y = 80;

}

// 设置坐标值：将 (x, y) 改为 (60 + i, 80 + j)

void setPoint(int i, int j) {

x = 60 + i;

y = 80 + j;

}

//构造一个输出坐标值的函数

void test01()

{

cout << "该点的坐标值为： （" << x << " , " << y<<")" << endl;

}

//设立私有变量x和y来表示坐标

private:

int x;

int y;

};

int main()

{

//创建一个Point类的对象p

Point p;

cout << "初始坐标：";

p.test01(); // 输出初始坐标

// 修改坐标，假设 i = 10, j = 20

p.setPoint(10, 20);

cout << "修改后的坐标：";

p.test01(); // 输出修改后的坐标

return 0;

}

运行结果截图：



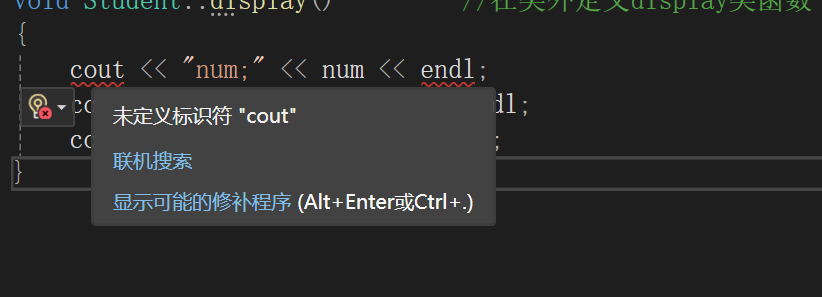
## 遇到的问题及解决方法

1. 在第四题当中，在设置这个对象数组的时候，可能会发生数组越界的情况，导致程序不能够正常运行。

解决方法：

需要确保循环的范围是 i < size，而不是 i <= size，因为数组索引是从0开始的，size 是数组的元素个数，而 size - 1 是最后一个有效索引。

1. 在第二题当中，当将代码完整写在编译器之后，出现了以下错误，导致程序不能够正常运行，出现报错



在本题中，定义在student.cpp的文件中，void函数中第一行cout语句显示“未定义表示符cout”的问题

原因：在头文件中加入了using namespace std;的命名空间，导致其他源文件受到了影响，不能够正常的进行输入输出。

解决方法:直接在main.cpp中加入using namespace std；

1. 总结和体会

在学习 C++ 中的类与对象时，我深刻体会到这部分知识是面向对象编程（OOP）的核心，也是理解 C++ 强大功能的基础。通过类与对象的概念，我逐渐明白了如何将现实世界中的事物抽象为程序中的实体，并通过封装、继承和多态等机制，使得程序更加灵活、可扩展。

首先，类（Class）是 C++ 中的基本构建块，它是对现实世界中事物的抽象。例如，通过 Student 类，我可以定义学生的基本信息，如学号、姓名和性别，这使得程序能够有条理地管理与学生相关的数据。类不仅仅是数据的集合，它还能包含一些操作这些数据的方法（成员函数），从而实现封装（Encapsulation）。通过类，程序员能够把数据与数据的操作封装在一起，避免外部直接访问或修改数据，确保数据的安全性和一致性。

其次，对象（Object）是类的实例化。通过类，我能够创建不同的学生对象，每个学生对象都有自己的属性和行为。这让我更容易理解如何通过具体实例化对象来模拟现实生活中的复杂情况，增强了程序的灵活性和可复用性。

在学习过程中，我发现了构造函数和析构函数的重要性。构造函数用于对象的初始化，确保在创建对象时自动完成必要的设置；析构函数则是在对象销毁时，自动释放资源，防止内存泄漏。这让我对内存管理有了更深的理解。

总之，学习 C++ 中的类与对象使我更加认识到面向对象编程的重要性，它不仅提升了程序的可读性、可维护性和可扩展性，还为我提供了一种更贴近现实世界的思考方式，使得我能够更加高效地设计和开发复杂的程序。