**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。

### 算法分析，程序结果：

1.#include<iostream>

using namespace std;

class Time // 定义Time类

{

private: // 数据成员为私有的

int hour;

int minute;

int sec;

public:

void settime()

{

cout << "请输入设定的时间：" << endl;

cin >> Time::hour; //输入设定的时间

cin >> Time::minute;

cin >> Time::sec;

}

void showtime()

{

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

}

};

int main()

{

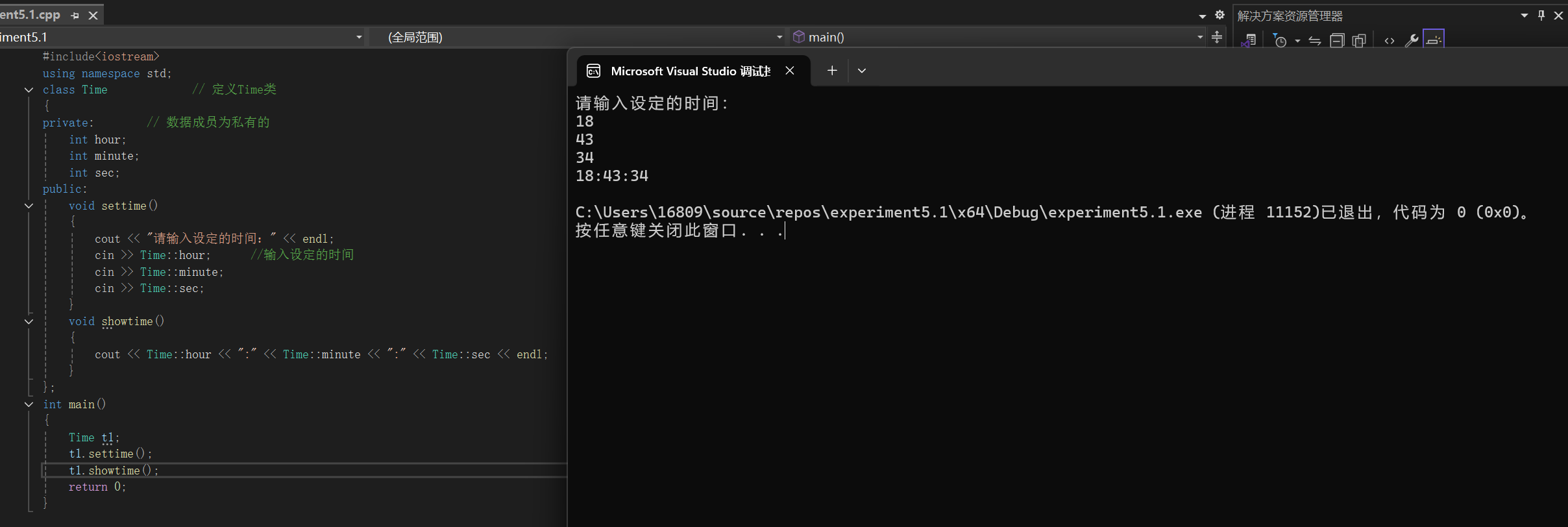
Time t1;

t1.settime();

t1.showtime();

return 0;

}



一般来说，那些需要被类外部的代码直接访问和操作的成员应该指定为公用的。

一般情况下，类中的数据成员如果不希望被类外部直接访问和修改，应该指定为私有的。

简短的、与类的内部数据紧密相关的函数最好放在类中定义。

当函数体比较复杂、函数功能相对独立于类的内部实现细节，或者函数需要在多个类中复用的时候，最好在类外定义。

2.student.h：#pragma once

#include<string>

//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();

void set\_value(int num,const char \* name,char sex);

private:

int num;

char name[20];

char sex;

};

student.cpp：//student.cpp 在此文件中进行函数的定义

#include <iostream>

#include<string>

#include"student.h" //不要漏写此行，否则编译通不过

using namespace std;

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

{

cout << "num：" << num << endl;

cout << "name：" << name << endl;

cout << "sex：" << sex << endl;

}

void Student::set\_value(int num,const char \* name,char sex)

{

Student::num = num;

strcpy\_s(Student::name, name);

Student::sex = sex;

}

main.cpp：//main.cpp 主函数模块

#include <iostream>//将类声明头文件包含进来

#include<string>

#include "student.h"

using namespace std;

int main()

{

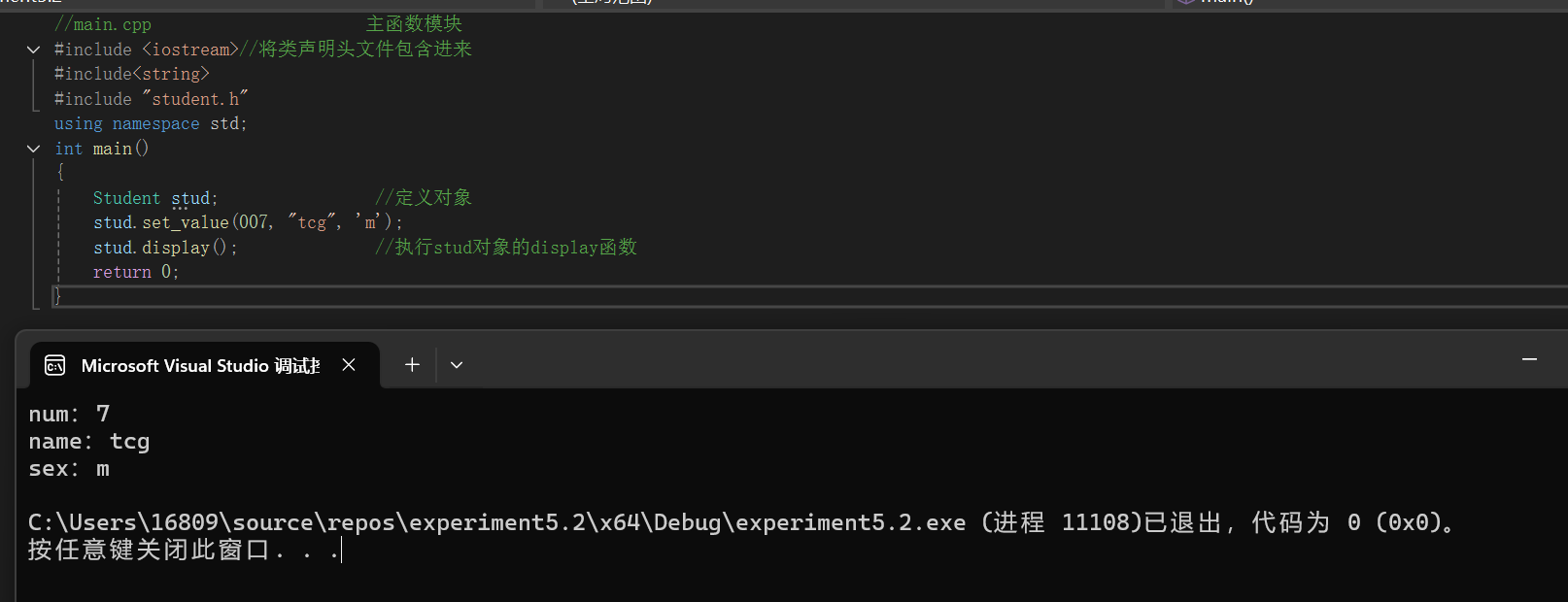
Student stud; //定义对象

stud.set\_value(007, "tcg", 'm');

stud.display(); //执行stud对象的display函数

return 0;

}



3.#include<iostream>

using namespace std;

class cuboid

{

public:

double length;

double width;

double height;

void set()

{

cout << "请输入长方体的长宽高：" << endl;

cin >> cuboid::length;

cin >> cuboid::width;

cin >> cuboid::height;

}

double volume()

{

double v = length \* width \* height;

return v;

}

void show()

{

double v = volume();

cout << "长方体的体积为：" << v << endl;

}

};

int main()

{

cuboid c1;

cuboid c2;

cuboid c3;

c1.set();

c1.volume();

c1.show();

c2.set();

c2.volume();

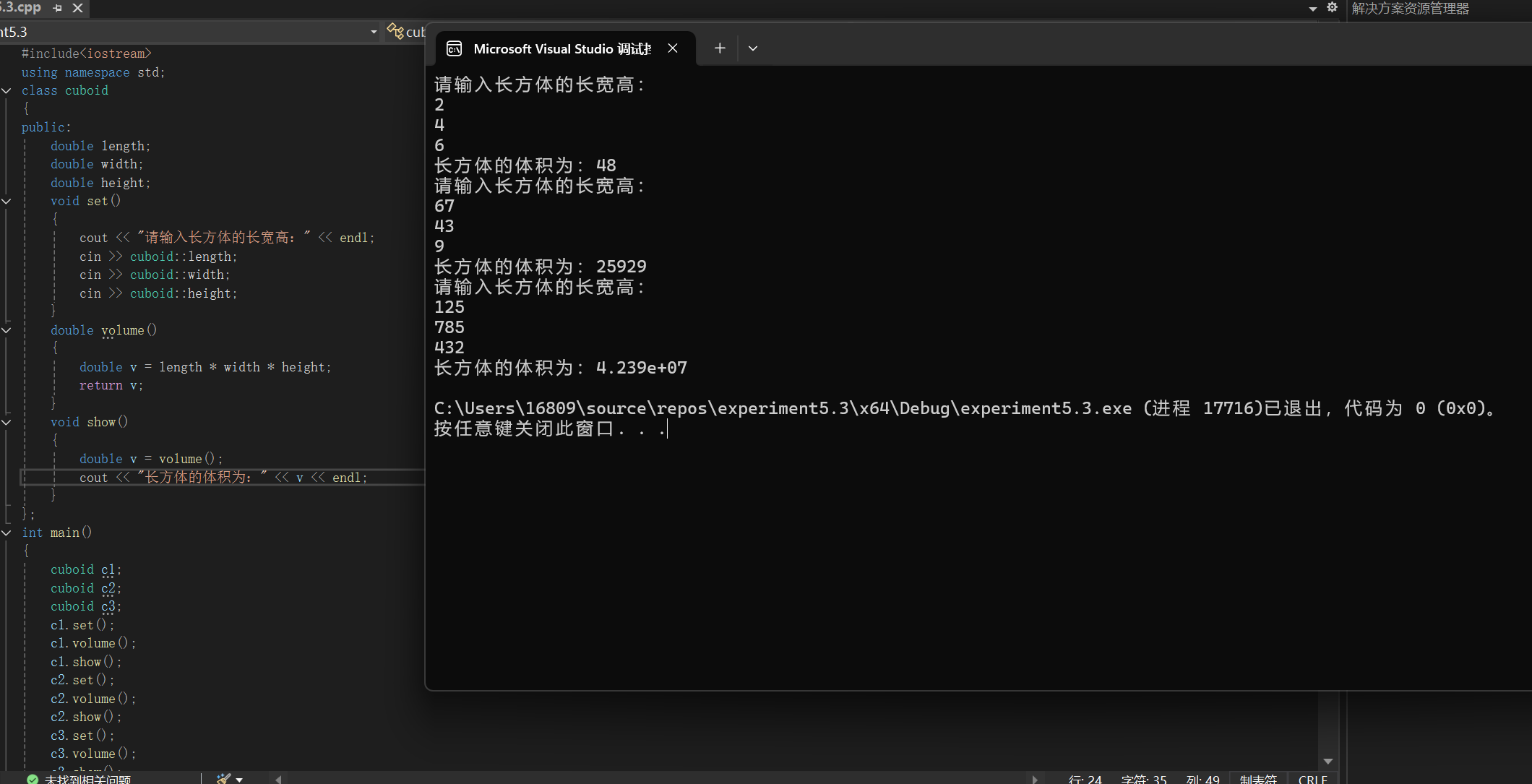
c2.show();

c3.set();

c3.volume();

c3.show();

}



4.#include<iostream>

using namespace std;

class Student

{

public:

int number;

float score;

void set()

{

cin >> number;

cin >> score;

}

};

void max(Student\* sz)

{

float m = sz[0].score;

int s = 0;

for (int i = 0; i < 5; i++)

{

if (sz[i].score > m)

{

m = sz[i].score;

s = i;

}

}

cout << "五个学生中成绩最高者的学号为：" << sz[s].number << " 他的成绩为：" << sz[s].score << endl;

}

int main()

{

Student sz[5] =

{ Student{57,56},

Student{65,38},

Student{89,45},

Student{48,78},

Student{75,65}

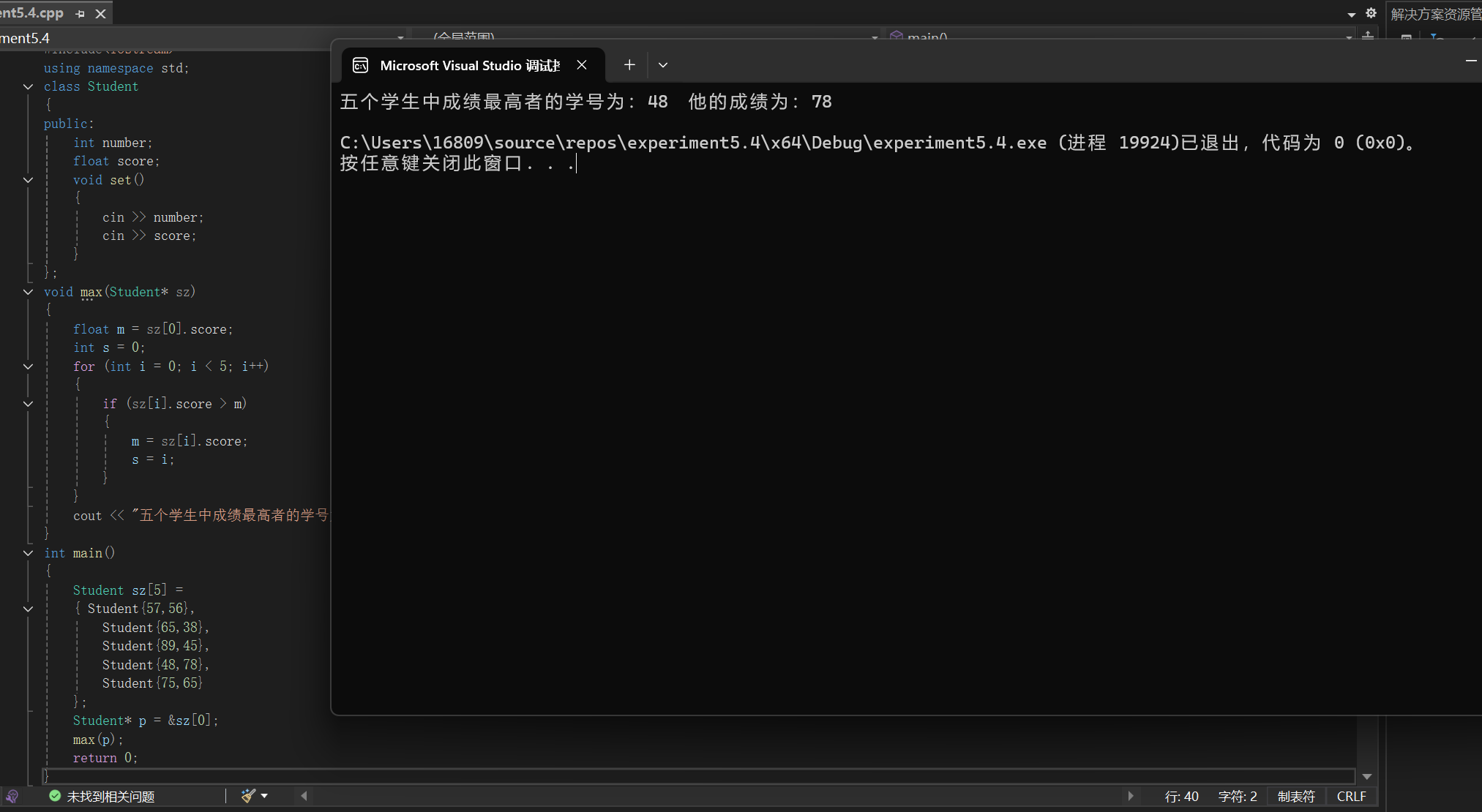
};

Student\* p = &sz[0];

max(p);

return 0;

}



5.#include <iostream>

using namespace std;

class Point

{

private:

int x;

int y;

public:

Point(int x,int y)

{

Point::x = x;

Point::y = y;

cout << "修改前的坐标为：（" << x << "," << y << ")" << endl;

}

void setPoint(int i, int j)

{

Point::x += i;

Point::y += j;

}

void display()

{

cout << "修改后的坐标值为：(" << x << "," << y << ")" << endl;

}

};

int main()

{

Point p(60, 80);

p.setPoint(43, 76);

p.display();

return 0;

}

