**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2301班1

学 号： 8209230110

姓 名： 肖峰

**实验报告成绩：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验** | **实验一** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |  |

**批阅教师：**

1. **算法分析，程序结果**
2. **代码：**

#include<iostream>

using namespace std;

class Time {

public:

void sethour() {

cin >> this->hour;

}

void setminute() {

cin >> this->minute ;

}

void setsec() {

cin >> this->sec;

}

int showhour() {

return this->hour;

}

int showminute() {

return this->minute;

}

int showsec() {

return this->sec;

}

private:

int hour;

int minute;

int sec;

};

int main()

{

Time tl;

tl.sethour();

tl.setminute();

tl.setsec();

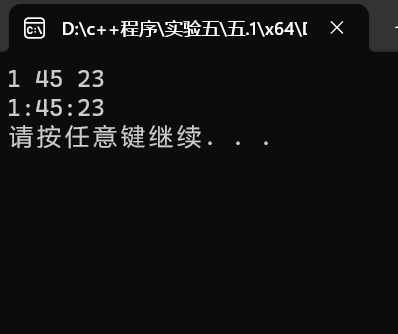
cout << tl.showhour() << ":" << tl.showminute() << ":" << tl.showsec() << endl;

system("pause");

return 0;

}

结果：

问题回答：

在C++中，公有成员应该指定为公有的，这样外部代码可以直接访问和使用它们。私有成员应该指定为私有的，以防止外部代码直接访问和修改它们，从而确保数据的安全性。

一般来说，与类的数据成员直接相关的函数最好放在类中定义，这样可以更好地封装类的数据和行为，提高代码的可维护性和可重用性。而与类的数据成员无关的函数，或者与类的数据成员关系不太密切的函数，最好在类外定义。

总的来说，将与类的数据成员直接相关的函数放在类中定义，将与类的数据成员无关或关系不太密切的函数放在类外定义，可以更好地组织和管理代码，提高代码的可读性和可维护性。

1. 代码：

Student.h:

#pragma once

class Student

{

public:

void display();

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

private:

int num;

char name[20];

char sex;

};

Student.cpp：

#include <iostream>

#include<cstring>

using namespace std;

#include"student.h"

void Student::display()

{

cout << "num：" << num << endl;

cout << "name：" << name << endl;

cout << "sex：" << sex << endl;

}

void Student::set\_value(int num, const char name[], char sex) {

this->num = num;

int i = 0;

strcpy\_s(this->name, sizeof(this->name), name);

this->sex = sex;

}

5.2.cpp:

#include<iostream>

#include"student.h"

using namespace std;

int main() {

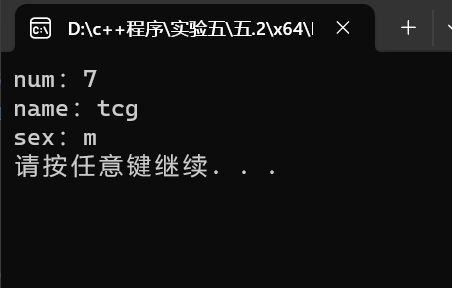
Student stud;

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

stud.display();

system("pause");

return 0;}

结果：  


3.

代码：

#include<iostream>

using namespace std;

class cube {

public:

friend istream& operator>>(istream& cin, cube& c);

double volume() {

double temp;

temp = length \* width \* height;

return temp;

}

private:

double length, width,height;

};

istream& operator>>(istream& cin, cube& c) {

cin >> c.length >> c.width >> c.height;

return cin;

}

ostream& operator<<(ostream& cout, cube& c) {

cout << c.volume();

return cout;

}

int main() {

cube c[3];

cout << "请分别输入三个长方体的长宽高" << endl;

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

cin >> c[i];

}

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

cout<<"第" <<i << "个长方体的体积为" << c[i] << endl;

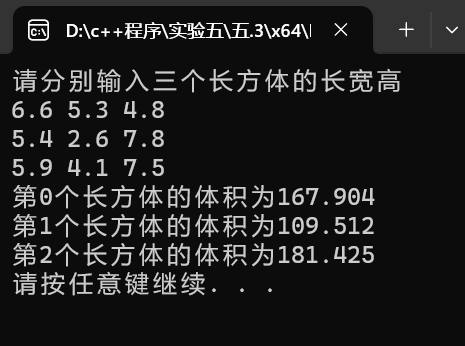
}

system("pause");

return 0;

}

结果：



4.

代码：

#include<iostream>

#include<string>

using namespace std;

class Student {

public:

friend void max(Student s[]);

friend istream& operator>>(istream& cin, Student& s);

private:

string id;

int score;

};

istream& operator>>(istream& cin, Student& s) {

cin >> s.id >> s.score;

return cin;

}

void max(Student s[]) {

for (int i = 0; i < 4; ++i) {

if (s[i].score > s[i + 1].score) {

Student temp=s[i];

s[i] = s[i + 1];

s[i + 1] = temp;

}

}

cout << "这五个学生中成绩最高者学号为：" << s[4].id << endl;

}

int main() {

Student s[5];

cout << "请输入5个学生的学号和成绩" << endl;

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

cin >> s[i];

}

max(s);

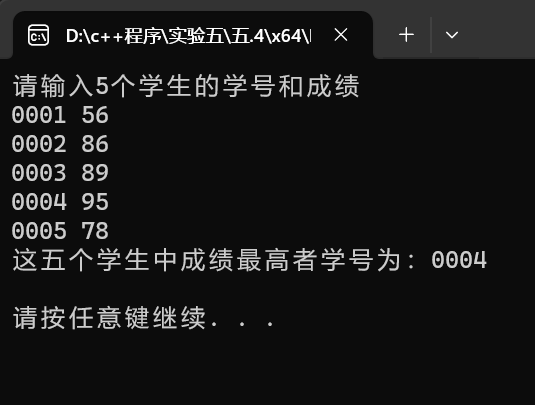
cout << endl;

system("pause");

return 0;

}

结果：



1. 代码：

#include<iostream>

using namespace std;

class point {

public:

point(int x, int y) {

this->x = x;

this->y = y;

}

void setPoint(int i, int j) {

this->x += i;

this->y += j;

}

void display() {

cout << "该点坐标为：" << "(" << this->x << "," << this->y << ")" << endl;

}

private:

int x;

int y;

};

int main() {

point p(60, 80);

p.setPoint(7, 9);

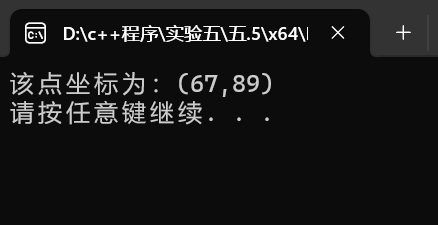
p.display();

system("pause");

return 0;

}

结果：



1. **遇到的问题与解决方法**
2. **const char\*类型的实参与char\*类型的形参不兼容----在函数的char\*类型形参前加const,使形参类型与传入的实参类型一样**

**2.当两个char字符组大小不同时，strcpy无法使用---使用strcpy\_s函数**

**三、体会**

**在学习C++中的类与对象的实验过程中，我掌握了声明类的方法，类和类的成员的概念以及定义对象的方法，我也深刻体会到了面向对象编程的重要性和优势。通过实验，我学会了如何创建类、定义对象以及使用构造函数和析构函数等，这些都是面向对象编程的基础知识。在实验中，我发现面向对象编程能够更好地组织和管理代码，使得程序更加模块化、可维护性更高。同时，通过类的封装特性，我可以隐藏类的实现细节，从而提高了代码的安全性。**

**在实验过程中，我也遇到了一些问题，比如对类和对象的概念理解不够深入、构造函数和析构函数的使用方法等。但通过反复实验和查阅资料，我逐渐克服了这些困难，对C++中的类与对象有了更加深入的理解。**

**总的来说，通过这次实验，我对C++中的类与对象有了更清晰的认识，也体会到了面向对象编程的优势。我相信这些知识和经验对我的编程能力和软件开发能力都会有所帮助。**