**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。

5、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。

**二、算法分析，程序结果**  
1.数据成员应该设为 private，以实现封，防止外部直接修改对象的内部状态（如 hour, minute, sec）。成员函数需要提供外部访问接口的函数（如 inputTime()、outputTime()）应设为 public，供外部调用。简单、短小的函数（如 inputTime()、outputTime()）可以放在类内，便于提高效率并简化代码。复杂、长时间执行的函数最好放在类外，减少类体积并提高可读性。

代码：#include <iostream>

using namespace std;

class Time { // 定义Time类

private:

int hour; // 私有数据成员

int minute;

int sec;

public:

void inputTime() {

cin >> hour >> minute >> sec;

}

void outputTime() const {

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

}

};

int main() {

Time t1;

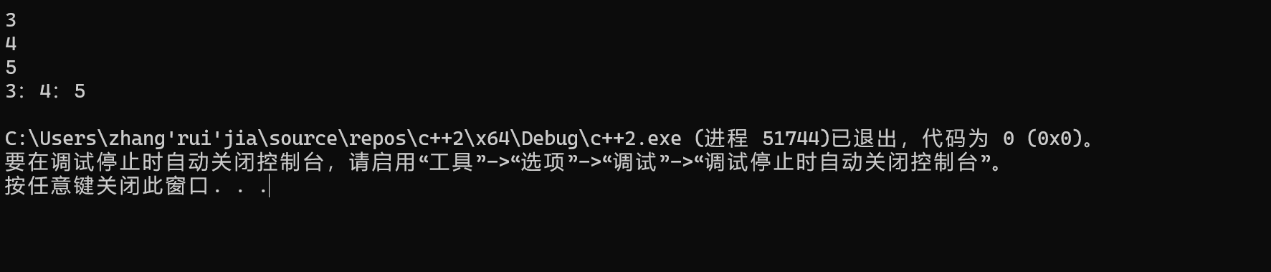
t1.inputTime();

t1.outputTime();

return 0;

}

结果：



2.

Main.cpp

Ik8l,

#include <iostream> //将类声明头文件包含进来

#include "student.h"

#include <string>

using namespace std;

int main()

{

Student stud; //定义对象

Student stud1;

stud1.setvalue(007, "tag", 'm');

stud1.display(); //执行stud对象的display函数

return 0;

}

头文件：

#pragma once

class Student //类声明

{private:

int num;

char name[20];

char sex;

public:

void display();//公用成员函数原型声明

void setvalue(int num,const char\* na,char s);

};

Student.cpp:

#include <iostream>

#include "student.h"

#include <string>

#include <cstring>

using namespace std;//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;

}

void Student::setvalue(int n,const char\* na,char s) {

num = n, strcpy(name, na), sex = s;

}.

3.代码：

#include <iostream>

using namespace std;

class cuboid {

private:

int length, width, height;

public:

cuboid(int length=0 , int width=0 , int height=0 ) {

this->length=length, this->width=width, this->height=height;

}

int v() {

return length \* width \* height;

}

~cuboid() {};

};

int main() {

int l1, w1, h1, l2, w2, h2, l3, w3, h3;

cin >> l1 >> w1 >> h1 >> l2 >> w2 >> h2 >> l3 >> w3 >> h3;

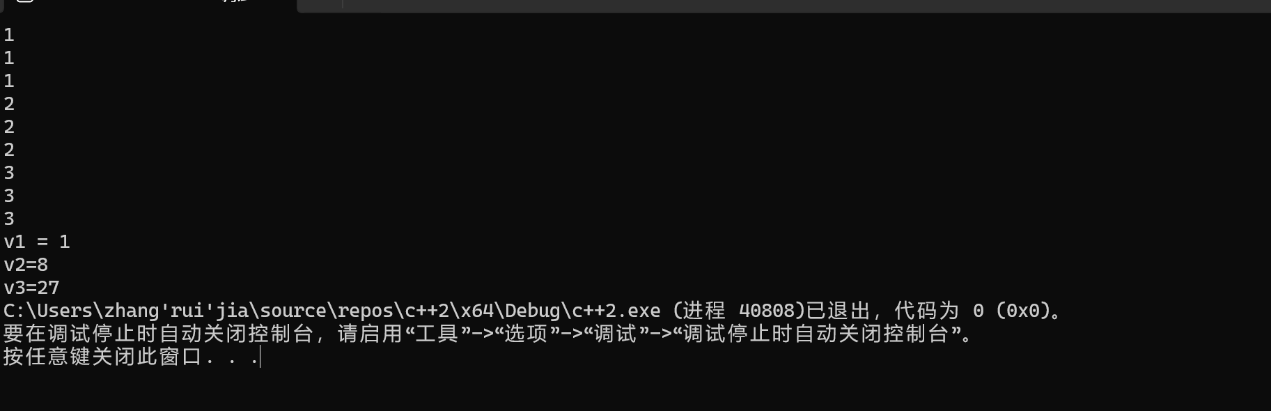
cuboid cub1(l1, w1, h1), cub2(l2, w2, h2), cub3(l3, w3, h3);

cout << "v1 = " << cub1.v() << endl << "v2=" << cub2.v() << endl << "v3=" << cub3.v();

return 0;

}

结果：



4. #include <iostream>

using namespace std;

class student {

private:int number, score;

public:

void show() {

cout << number << " " << score << endl;

}

student (int number =0,int score=0){

this->number = number, this->score = score;

}

int sco() {

return score;

}

};

int max(student\* a[]) {

int s = 0, b, c;

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

if (a[i]->sco() >= s)s = a[i]->sco();

}

return s;

}

int main() {int a[5], b[5];

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

cin >> a[i] >> b[i];

}

student stu[5]= { student(a[0],b[0]),

student(a[1],b[1]),

student(a[2],b[2]),

student(a[3],b[3]),

student(a[4],b[4]),

};

student\* p[5];

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

p[i] = &stu[i];

}

int s = max(p);

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

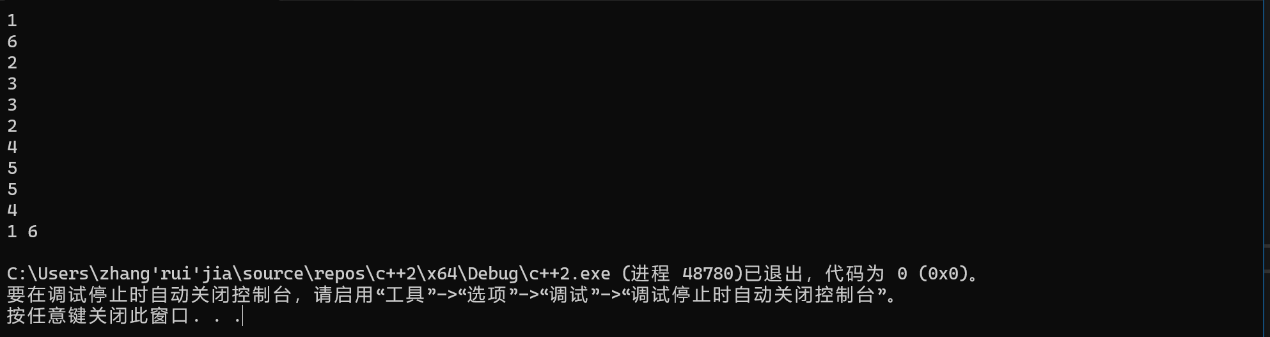
if (p[i]->sco() == s)p[i]->show();

}

return 0;

}

结果：



5.代码：

#include <iostream>

using namespace std;

class point {

private :

int x, y;

public:

point(int x = 0, int y = 0) {

this->x = x, this->y = y;

}

void setpoint(int i, int j) {

this->x += i, this->y += j;

}

void display() {

cout << "x:" << x << "y:" << y;

}

};

int main() {

point poi1(60, 80);

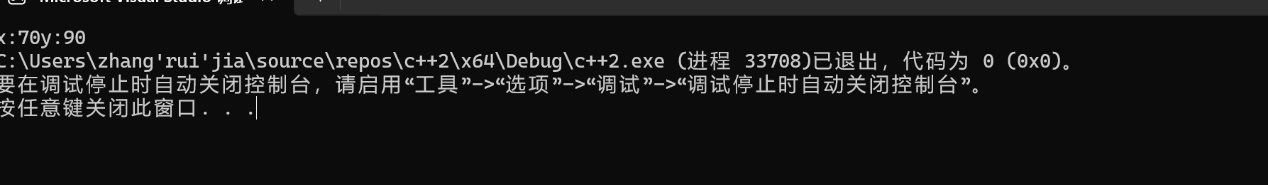
poi1.setpoint(10, 10);

poi1.display();

return 0;

}

结果：



三．遇到的困难和解决方法

**遇到的困难：**  
在这次C++实验中，我遇到了一个困难，就是在使用类的成员函数时，常常忘记对类成员进行初始化，导致程序运行时出现了不可预料的行为。例如，某些成员变量没有初始化就被访问，结果引发了逻辑错误。

**解决方法：**  
我通过逐步调试和检查每个成员变量的初始化过程，确保每个变量在使用之前都有明确的初值。此外，我也学习了如何使用初始化列表来简化成员变量的初始化，并保证它们始终处于有效状态。

**四．体会**  
通过这次实验，我更加理解了类内部成员变量初始化的重要性。及时初始化成员变量能够有效避免程序中的潜在错误，增强代码的稳定性和可维护性。同时，我也体会到良好的编程习惯对编写高质量代码的重要性，尤其是在面向对象编程中，管理好类的状态对于整个程序的正确运行至关重要。