**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?



#include<iostream>

using namespace std;

class Time {

private:

int hour;

int minute;

int sec;

public:

void settime(){

cin >> hour;

cin >> minute;

cin >> sec;

this->hour;

this->hour;

this->sec;

}

void showtime(){

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

}

};

int main() {

Time t1;

t1.settime();

t1.showtime();

return 0;

}

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()

{

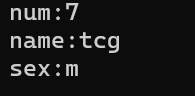
Student stud; //定义对象

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。

#include<iostream>

#include"student.h"

using namespace std;

void Student::display() {

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

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

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

} void set\_value(int Num, const char\* Name, char Sex) {

num=Num;

strcpy\_s(name, Name);

sex=Sex;

}

void display();

private:

int num;

char name[20];

char sex;

};

#include <iostream> //将类声明头文件包含进来

#include"student.h"

int main()

{

Student stud1;

stud1.set\_value(007,"tgc",'m');

stud1.display(); //执行stud对象的display函数

return 0;

}

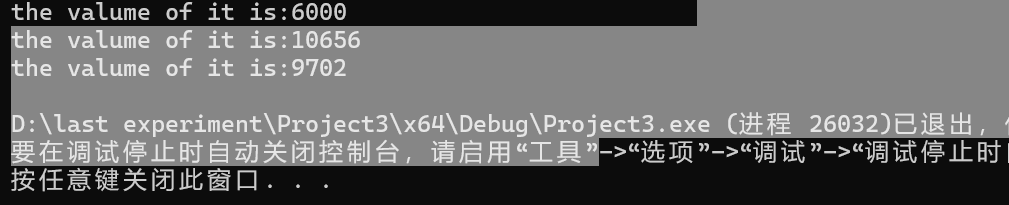
3、需要求3个长方柱的体积，请编一个基于对象的程序。数据成员包括length(长)、width(宽)、height(高)。要求用成员函数实现以下功能：

(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。



#include<iostream>

using namespace std;

class Cyboid {

private:

double length;

double width;

double height;

double v;

public:

Cyboid(double a, double b, double c) {

length = a;

width = b;

height=c;

}

void volume() {

v = length \* width \* height;

}

void show() {

cout << "the valume of it is:" << v << endl;

}

};

int main() {

Cyboid m(10, 20, 30);

m.volume();

m.show();

Cyboid n(12, 24, 37);

n.volume();

n.show();

Cyboid k(14, 21, 33);

k.volume();

k.show();

return 0;

}

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。

944da9dd0ade6176ab237ad0e27d1e7

#include<iostream>

using namespace std;

class Student {

private:

int id;

int grade;

public:

Student(int Id, int Grade) {

id = Id;

grade = Grade;

}

friend void max(Student\*arr);

};

void max(Student\* arr) {

int max\_grade = arr[0].grade;

int k = 0;

for (int i = 1; i < 5; i++) {

if (arr[i].grade > max\_grade) {

max\_grade = arr[i].grade;

k = i;

}

}

cout << "the greatest student is:" << arr[k].id << endl;

}

int main() {

Student stud[5] = {

Student(101,78),Student(102,85),Student(103,98),Student(104,100),Student(105,97)

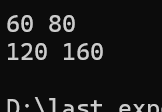
};

Student\* p = &stud[0];

max(p);

return 0;

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。



#include<iostream>

using namespace std;

class point {

private:

int x;

int y;

public:

point(int X, int Y) {

x = X;

y = Y;

}

void setpoint(int i, int j) {

x += i;

y+= j;

}

void display() {

cout << x << ' ' << y << endl;

}

};

int main() {

point p(60,80);

p.display();

p.setpoint(60,80);

p.display();

}

反思：一、成员指定为公用（public）的情况

接口函数：  
这些函数是类对外提供服务的途径，用于让外部代码能够操作类的对象、获取或设置对象的相关状态等。例如，对于一个表示图形的类，如果有计算面积、周长等功能，像getArea()、getPerimeter()这样的函数通常应设为公有，方便外部调用者获取该图形对象的关键属性信息。

需要外部直接访问的成员变量（在合理情况下）：  
虽然一般建议尽量将成员变量设为私有然后通过公有的访问器（getter）和修改器（setter）函数来操作，但在一些简单且逻辑清晰的场景下，某些成员变量可以直接设为公有。比如一个只包含简单数据且主要用于数据传递和临时存储的结构体类型的类，其成员变量设为公有可能更方便直接读写，不过这种情况相对较少，要谨慎使用，避免破坏封装性。

二、成员指定为私有（private）的情况

内部实现细节相关的成员变量：  
类内部用于保存对象状态、辅助实现各种功能的变量，通常不想让外部代码随意访问和修改，以防破坏类的内部逻辑一致性。例如在一个Stack（栈）类中，用于存储栈元素的数组以及表示栈顶位置的指针变量等，这些都是内部实现细节，应设为私有，外部代码没必要也不应该直接操作它们。

辅助实现公有接口的内部函数：  
有些函数只是在类的内部多个公有函数之间起到辅助作用，不应该被外部代码调用，将其设为私有可以隐藏实现细节，增强类的封装性。比如在上述Stack类中，可能有一个用于判断栈是否已满的isFull()函数，它只是在push（入栈）操作时内部调用用于判断能否入栈，外部用户不需要直接调用这个函数，所以应设为私有。

三、函数最好放在类中定义（内联函数情况居多）的情况

简单短小且频繁调用的函数：  
像访问器（getter）和修改器（setter）函数，如果它们的实现非常简单，只是返回或设置一个成员变量的值，将其定义在类内部声明为内联函数（在类定义中直接实现函数体），编译器有可能会进行内联展开，减少函数调用的开销。例如对于一个Person类，有getName()函数用于返回成员变量。

在将char数组传入过程发生不匹配现象，要用const char\*来做形参类型，并且使用strcpy\_s函数来实现复制。

在写第四题的时候，不知道如何访问私有变量，后面使用友元函数解决问题。

体会：第四题给我留下了深刻的体会

一、面向对象编程思维的体现

在这个题目中，首先定义了Student类来抽象表示学生这一概念，将学号和成绩这些与学生相关的属性封装在类内部，这很好地体现了面向对象编程中封装的特性。通过类的构造函数可以方便地创建学生对象，并且利用成员函数来获取对象的属性值，使得代码结构更加清晰、有条理，相比于单纯使用结构体或者普通变量来存储数据，面向对象的方式让数据和操作数据的逻辑结合得更为紧密，有助于提高代码的可维护性和可扩展性。

二、指针在对象操作中的运用

使用指向对象的指针作为函数参数传递给max函数，这展现了指针在面向对象编程里灵活操作对象的优势。通过指针数组，我们可以方便地管理多个学生对象，并且在max函数中，能够以统一的方式去访问每个对象的成员函数来比较成绩大小，而不用把整个对象进行复制传递（如果直接传递对象，对于复杂对象可能会带来较大的复制开销），提高了程序运行的效率，同时也体现了指针在处理多个同类对象集合时简洁高效的特点。

三、函数设计与代码复用性

max函数的设计具有一定的通用性，它只需要接收指向学生对象的指针数组以及数组的大小，就能完成找出最高成绩学生学号的功能。这种设计思路意味着如果后续需要处理更多学生对象或者类似的查找最大、最小等基于对象属性比较的需求，只要按照相同的规则传入合适的指针数组，函数就可以复用，不用重复编写相似的查找逻辑，这有助于提高代码的复用程度，减少代码冗余，让整个程序的代码结构更加简洁高效。

四、内存管理的重要性

在main函数中，我们使用new关键字动态分配内存来创建学生对象，与之对应的，在使用完这些对象后，需要通过delete操作符来释放内存，以避免出现内存泄漏的问题。这提醒我们在使用动态内存分配的编程场景中，一定要谨慎地做好内存管理工作，否则随着程序的运行，可能会不断消耗内存资源，导致程序出现异常甚至崩溃，所以养成良好的内存管理习惯是编写可靠程序的重要环节。

五、整体程序逻辑与调试

通过这个简单的程序，能够体会到编写程序时清晰的逻辑思路是很关键的。从创建对象数组、调用查找函数到最后的内存释放，每个环节都紧密相连，任何一个环节出现问题都可能导致程序无法正确运行。而且在调试过程中，比如检查是否正确获取到对象的成绩、指针是否正确指向了相应的对象等，需要我们仔细梳理代码逻辑，借助调试工具来排查可能出现的错误，这也锻炼了我们的代码调试能力以及对程序整体逻辑的把控能力。

总之，这道看似简单的题目涵盖了面向对象编程、指针运用、函数设计、内存管理以及程序逻辑等多方面的知识点，有助于加深对编程基础知识的理解和运用能力的提升。

何承祖

软工2402