**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。

**实验结果：**

## 程序修改

1.1 问题解答：

公用成员 vs 私有成员:

私有数据成员：hour, minute, sec。这些应该是私有的，因为它们是内部状态，不应该直接被外部访问或修改。

公用成员函数：setTime() 和 showTime()。这些成员函数应该是公用的，因为它们提供了与外部世界交互的接口，允许用户输入和查看时间。

类内 vs 类外函数定义:

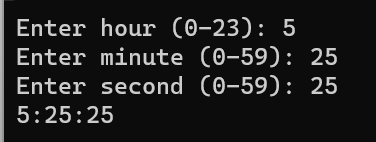
类内定义：对于较短、简单的函数，直接在类内定义是更好的选择，如showTime方法，由于方法内容不会修改类的任何成员变量，甚至可以用const修饰，这可以提高代码的可读性和组织性。

类外定义：对于更复杂或更长的函数，可以在类外定义，如setTime方法，这有助于保持类的定义简洁，同时将实现细节分离。

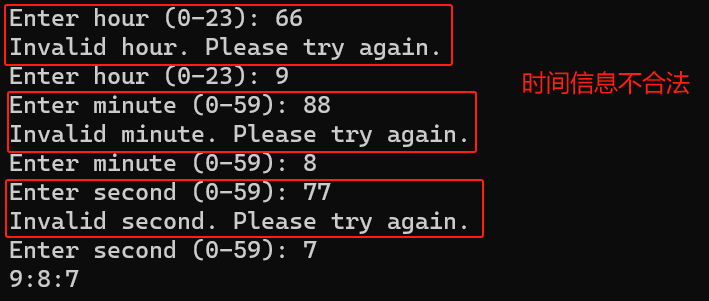
* 1. 运行结果：

由于代码的时间设置上没有相关约束，导致用户输入时间时，即使时间信息不合法，如hour=27，程序也会正常输出，所以我添加了时间的校验功能，在用户输入错误时，提醒用户重新输入时间。

1. 正常输入时间



1. 输入错误时间



* 1. 代码：

#include<iostream>

using namespace std;

class Time // 定义Time类

{

private: // 数据成员为公用的

int hour;

int minute;

int sec;

public:

// 构造函数

Time() : hour(0), minute(0), sec(0){}

void setTime();

void showTime() const {

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

}

};

void Time::setTime() {

// 输入小时数，检查有效性

while (true) {

cout << "Enter hour (0-23): ";

cin >> hour;

if (hour >= 0 && hour <= 23) {

break;

}

cout << "Invalid hour. Please try again.\n";

}

// 输入分钟数，检查有效性

while (true) {

cout << "Enter minute (0-59): ";

cin >> minute;

if (minute >= 0 && minute <= 59) {

break;

}

cout << "Invalid minute. Please try again.\n";

}

// 输入秒数，检查有效性

while (true) {

cout << "Enter second (0-59): ";

cin >> sec;

if (sec >= 0 && sec <= 59) {

break;

}

cout << "Invalid second. Please try again.\n";

}

}

int main()

{

Time t1; //定义t1为Time类对象

t1.setTime(); //输入设定的时间

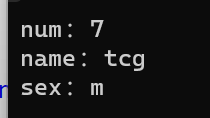
t1.showTime();

return 0;

}

## 完善程序

2.1 运行结果：



* 1. 代码：

// student.h

#pragma once

#ifndef STUDENT\_H

#define STUDENT\_H

class Student //类声明

{

public: //公用成员函数原型声明

void display();

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

private:

int num;

char name[20];

char sex;

};

#endif

// student.cpp

#include <iostream>

#include <cstring>

#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 n, const char\* nm, char s) {

num = n;

strncpy\_s(name, nm, 19); // 使用strncpy确保不会超过数组大小

name[19] = '\0'; // 确保字符串正确结束

sex = s;

}

// main.cpp

#include <iostream> //将类声明头文件包含进来

#include "student.h"

int main()

{

Student stud; //定义对象

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

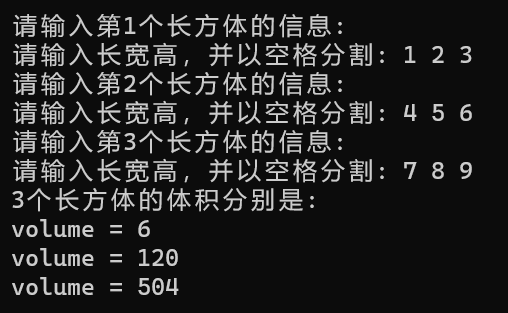
stud.display(); //执行stud对象的display函数

return 0;

}

## 编写长方体

3.1 运行结果：



* 1. 代码：

#include <iostream>

using namespace std;

class Cuboid {

private:

int length, width, height;

public:

void setInfo() {

cout << "请输入长宽高，并以空格分割: ";

cin >> length >> width >> height;

}

int volume() {

return length \* width \* height;

}

void display() {

cout << "volume = " << volume() << endl;

}

};

int main()

{

Cuboid cuboid1, cuboid2, cuboid3;

cout << "请输入第1个长方体的信息:\n";

cuboid1.setInfo();

cout << "请输入第2个长方体的信息:\n";

cuboid2.setInfo();

cout << "请输入第3个长方体的信息:\n";

cuboid3.setInfo();

cout << "3个长方体的体积分别是:\n";

cuboid1.display();

cuboid2.display();

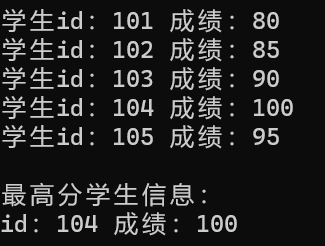
cuboid3.display();

return 0;

}

## 对象数组

4.1 运行结果：



* 1. 代码：

#include <iostream>

using namespace std;

#define SIZE 5

class Student {

private:

int id, score;

public:

Student() : id(0), score(0){}

Student(int id, int score) : id(id), score(score) {}

int getId() {

return id;

}

int getScore() {

return score;

}

};

void max(Student\* student) {

if (SIZE <= 0)return;

int maxIndex = 0;

int maxScore = student[0].getScore();

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

if (maxScore < student[i].getScore()) {

maxIndex = i;

maxScore = student[i].getScore();

}

}

cout << "最高分学生信息：" << endl;

cout << "id：" << student[maxIndex].getId() << " 成绩：" << student[maxIndex].getScore() << endl;

}

int main()

{

Student students[SIZE] = {

Student(101, 80),

Student(102, 85),

Student(103, 90),

Student(104, 100),

Student(105, 95)

};

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

cout << "学生id：" << students[i].getId() << " 成绩：" << students[i].getScore() << endl;

}

cout << endl;

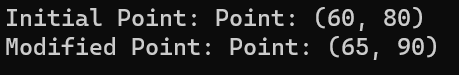
max(students);

return 0;

}

## 坐标表示

5.1 运行结果：



* 1. 代码：

#include <iostream>

using namespace std;

#define SIZE 5

class Point {

private:

int x, y;

public:

Point() :x(60), y(80){}

void setPoint(int i, int j) {

x += i;

y += j;

}

void display() {

cout << "Point: (" << x << ", " << y << ")" << endl;

}

};

int main()

{

Point point; // 默认构造函数将坐标初始化为(60,80)

cout << "Initial Point: ";

point.display(); // 显示初始坐标

point.setPoint(5, 10); // 修改坐标为(65,90)

cout << "Modified Point: ";

point.display(); // 显示修改后的坐标

return 0;

}

**实验心得：**

本次实验主要是对类和对象的使用，实验内容较为简单，通过代码的编写，我更加深刻理解了面向对象编程(OOP)的基本原则，如封装、数据隐藏和成员函数的使用。

5个OOP的编程体现了类设计是面向对象编程的核心，而构造函数的使用（如在 Point 类中所见）是初始化对象状态的有效方式。第二题还包括将类定义和实现分开（如使用头文件和源文件），这使得代码更容易维护和重用。第四题对象数组也涉及到了将类、数组、指针结合起来的方法，让我更加理解了指针和动态内存管理在C++中的应用。

实验的过程不仅加深了我对C++语言特性的理解，也有助于将理论知识与实际编程技能结合起来，这是成为一名有效的软件开发者的关键部分。