**广州大学学生实验报告**

开课学院及实验室：计算机科学与工程实验室电子楼412A室 2018年 3月27日

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 计算机科学与教育软件 | | 年级  专业班 | 软件171 | 姓名 | 谢金宏 | 学号 | 1706300001 |
| 实验课程名称 | | 面向对象程序设计实验 | | | | | 成绩 |  |
| 实验项目名称 | | 类与对象 | | | | | 指导老师 | 陶文正 |

**一、实验目的与要求**

1. 掌握类和对象的概念、定义和使用方法。
2. 掌握不同特性对象成员的访问方法，以及this指针的概念和使用方法。
3. 掌握构造函数和析构函数的使用方法。
4. 掌握静态成员的使用方法。
5. 掌握友元修饰符的使用方法。

学会使用VC6的Debug的功能跟踪构造函数、析构函数的执行顺序。

**二、实验设备与平台**

实验设备要求每个学生一台电脑，其中运行环境为VC++ 6.0系统。本报告采用G++7.2.0完成实验，对实验部分代码进行了调整，以符合较新的C++标准。

**三、实验内容及步骤**

(1) 定义一个Score类，其中包括私有数据成员和公有成员函数，即

num 学号

Math 高等数学成绩

English 英语成绩

Programming 程序设计成绩

inscore() 输入学号和各科成绩，并且计算平均成绩

showscore() 输出学号和各科成绩

使用score类，输入某班n（事先不能确定）个学生的学号和各科成绩，然后求各个学生的平均成绩，并列表输出学生的学号、各科成绩和平均成绩。

#include <iostream>

#include <string>

using namespace std;

class Score {

public:

void Inscore() {

cout << "Please input the information, \n";

cout << "Student Code: "; cin >> Serial;

cout << "Math: "; cin >> Math;

cout << "English: "; cin >> English;

cout << "Programming: "; cin >> Programming;

}

void ShowScore() {

cout << "Student #" << Serial << endl;

cout << "Math: " << Math << endl;

cout << "English: " << English << endl;

cout << "Programming: " << Programming << endl;

cout << "Average: " << (Math + English + Programming)/3.0 << endl;

}

private:

string Serial;

int Math, English, Programming;

};

int main() {

cout << "Please tell me the number of student: " << endl;

int n; cin >> n;

Score \*sc = new Score[n];

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

sc[i].Inscore();

cout << endl;

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

sc[i].ShowScore();

}

(2)写出下列程序的执行结果，然后上机进行验证。用VC6的Debug的功能跟踪构造函数、析构函数和成员函数的执行顺序。

#include <iostream>

using namespace std;

class TEST

{

public:

    TEST() {

cout<< "调用构造函数" << endl; x = 2; y = 50; z = 9;

}

    TEST(int a, int b, int c) {

        cout << "调用重载构造函数"<< endl;

        x = a; y = b; z = c;

    }

void display() {

        cout << "x=" << x << '\t' << "y=" << y <<'\t' << "z=" << z << endl;

    }

    int max(int a, int b) {

        if (a > b) return a; else return b;

    }

    ~TEST () {

        cout << "调用析构函数" << endl;

        cout << x << "," << y <<"和" << z << "最大值是:" << max( max( x,y ),z) << endl;

    }

    private:

    int x,y,z;

};

int main()

{

    TEST obj1;

    obj1.display();

    TEST obj2(33, 20, 80);

    obj2.display();

}

构造函数执行，然后是重载构造函数，然后是两次析构函数，析构顺序是obj2, obj1。

（3）假设某城市人人所得税的税基为2000元，而其征收个人所得税的规定如下：不超过税基免征税，按超过税基部分征收所得税。超过部分在2000元以下，征税5%；5000元以下，征税10%；10000元以下，征税15%；10000元以上，征税20%。建立一个Incomec类，共包含的数据成员和成员函数如下：

income 数据成员，表示职工的实际收

wage 数据成员，表示职工工资

setWage（double m） 成员函数，设置工资值

getTaxrate() 静态成员函数，计算税率

calIncome（） 静态成员函数，计算职工的实际收入

在主函数中，实例化不同的Incomec类对象，并计算它们的实际收入。

#include <iostream>

using namespace std;

class Incomec

{

private:

    double wage, income;

public:

    void setWage(double m) {

        wage = m;

    }

    static void calIncome(Incomec &p) {

        if (p.wage <= 2000) p.income = p.wage;

        else if (p.wage <= 4000) p.income = p.wage\*0.95;

        else if (p.wage <= 7000) p.income = p.wage\*0.90;

        else if (p.wage <= 12000) p.income = p.wage\*0.75;

        else p.income = p.wage\*0.80;

    }

    friend ostream& operator <<(ostream&, Incomec);

};

ostream& operator <<(ostream& out, Incomec ic) {

    out << "职工工资：" << ic.wage << " 税后收入：" << ic.income;

    return out;

}

int main()

{

    Incomec ia, ib, ic, id, ie;

    ia.setWage(1200); ib.setWage(2500);

    ic.setWage(4500); id.setWage(6000);

    ie.setWage(12000);

    Incomec::calIncome(ia); Incomec::calIncome(ib);

    Incomec::calIncome(ic); Incomec::calIncome(id);

    Incomec::calIncome(ie);

    cout << ia << endl; cout << ib << endl;

    cout << ic << endl; cout << id << endl;

    cout << ie << endl;

}

程序代码如上。

（4）建立一个复数类Imaginary，其私有数据成员x和y表示复数的实部和虚部，构造函数Imaginary用于对复数的实部和虚部初始化，友员函数add,sub,mul和div分别用于进行复数的加、减、乘和除法运算，静态函数show用于显示运算结果。在主函数中，实例化两个复数，并输入一个运算符，按运算符选择相应的友员函数进行复数运算，然后输出运算结果。

#include <iostream>

using namespace std;

class Imaginary

{

private:

    double x, y;

public:

    Imaginary(double x, double y) {

        this->x = x, this->y = y;

    }

    friend Imaginary add(const Imaginary &, const Imaginary&);

    friend Imaginary sub(const Imaginary &, const Imaginary&);

    friend Imaginary mul(const Imaginary &, const Imaginary&);

    friend Imaginary div(const Imaginary &, const Imaginary&);

    static void show(const Imaginary &i) {

        cout << i.x << '+' << i.y << 'i' << endl;

    }

};

Imaginary add(const Imaginary &a, const Imaginary &b) {

    return Imaginary(a.x+b.x, a.y+b.y);

}

Imaginary sub(const Imaginary &a, const Imaginary &b) {

    return Imaginary(a.x-b.x, a.y-b.y);

}

Imaginary mul(const Imaginary &a, const Imaginary &b) {

    return Imaginary(a.x\*b.x-a.y\*b.y, a.y\*b.x+a.x\*b.y);

}

Imaginary div(const Imaginary &a, const Imaginary &b) {

    return Imaginary((a.x\*b.x+a.y\*b.y)/(b.x\*b.x+b.y\*b.y), (a.y\*b.x-a.x\*b.y)/(b.x\*b.x+b.y\*b.y));

}

int main()

{

    Imaginary a(1, 2); Imaginary b(3, 4);

    Imaginary::show(add(a, b));

    Imaginary::show(sub(a, b));

    Imaginary::show(mul(a, b));

    Imaginary::show(div(a, b));

}

程序代码如上。