西安交通大学实验报告

课程 计算机程序设计 实验名称 继承 第 1 页 共 19 页

系 别 工科试验班(钱学森实验班) 实 验 日 期 2017 年5 月 11 日

专业班级 钱学森62班 组别 无 实 验 报 告 日 期 2017 年5 月 15 日

姓 名 周宇晨 学号 2160405046 报 告 退 发 ( 订正 、 重做 ）

同 组 人 无 教 师 审 批 签 字

一、实验目的

1.熟练类的声明、定义和构造、析构。

2.辨明public，private和protected的区别，正确使用分类。

3.学习继承的用法，了解不同继承方式的异同。

二、实验题目

题目一 从类Person中派生出一个教师类，新增的属性有：专业、职称和主讲课程（一门），并为这些属性定义相应的方法。

1.要点分析

类的声明；派生类的声明。

2.程序源码

#include<iostream>

using namespace std;

class Person

{

private:

char Name[20];

char Sex;

int Age;

public:

void Register(char \*name, int age, char sex)

{

strcpy\_s(Name, name);

Age = age;

Sex = (sex == 'm' ? 'm' : 'f');

}

void ShowMe()

{

cout << "姓名：" << Name << endl;

cout << "年龄：" << Age << endl;

cout << "性别：" << Sex << endl;

}

};

class Teacher :public Person

{

private:

char Department[25];

char Position[10];

char Lesson[20];

public:

void RegisterTch(char \*name, int age, char sex, char \*department, char \*position, char \*lesson)

{

strcpy\_s(Department, department);

strcpy\_s(Position, position);

strcpy\_s(Lesson, lesson);

Register(name, age, sex);

}

void showTch()

{

ShowMe();

cout << "专业：" << Department << endl;

cout << "职称：" << Position << endl;

cout << "主讲课程：" << Lesson << endl;

}

};

int main()

{

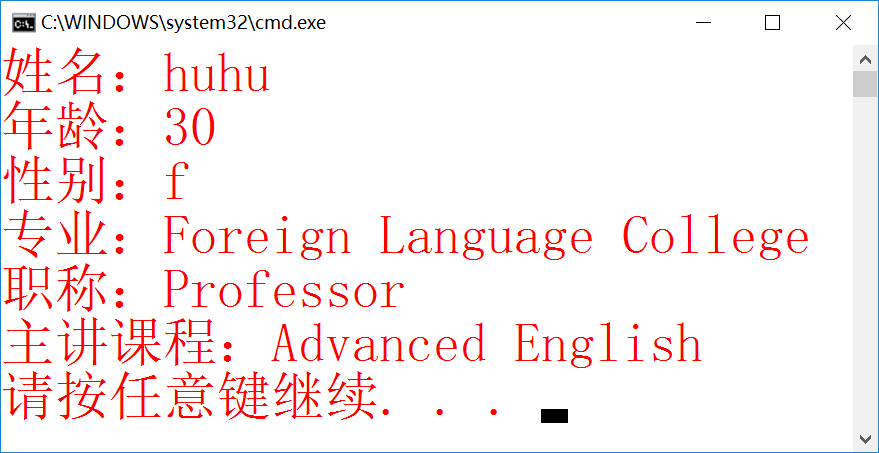
Teacher tch;

tch.RegisterTch("huhu", 30, 'f', "Foreign Language College", "Professor", "Advanced English");

tch.showTch();

return 0;

}



题目二 许多研究生既有学生的属性，又有教师的属性。试通过多重继承说明一个研究生类。

1.要点分析

略。

2.程序源码

#include<iostream>

using namespace std;

class Person

{

private:

char Name[20];

char Sex;

int Age;

public:

void Register(char \*name, int age, char sex)

{

strcpy\_s(Name, name);

Age = age;

Sex = (sex == 'm' ? 'm' : 'f');

}

void ShowMe()

{

cout << "姓名：" << Name << endl;

cout << "年龄：" << Age << endl;

cout << "性别：" << Sex << endl;

}

};

class Student :public Person

{

private:

char Number[11];

public:

void RegisterStu(char \*name, int age, char sex, char \*number)

{

strcpy\_s(Number, number);

Register(name, age, sex);

}

void ShowStu()

{

ShowMe();

cout << "学号：" << Number << endl;

}

};

class Teacher :public Person

{

private:

char Department[25];

char Position[10];

public:

void RegisterTch(char \*name, int age, char sex, char \*department, char \*position)

{

strcpy\_s(Department, department);

strcpy\_s(Position, position);

Register(name, age, sex);

}

void ShowTch()

{

cout << "专业：" << Department << endl;

cout << "职称：" << Position << endl;

}

};

class Postgraduate :public Student, public Teacher

{

public:

void RegisterPg(char \*name, int age, char sex, char \*number, char \*department, char \*position)

{

RegisterStu(name, age, sex, number);

RegisterTch(name, age, sex, department, position);

}

void ShowPg()

{

ShowStu();

ShowTch();

}

};

int main()

{

Postgraduate pg;

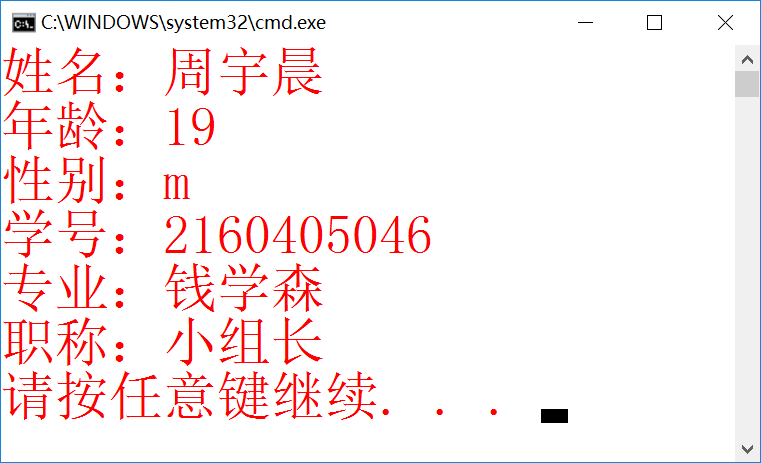
pg.RegisterPg("周宇晨", 19, 'm', "2160405046", "钱学森", "小组长");

pg.ShowPg();

return 0;

}

3.实验结果



题目三 修改例9-5，从Point类中派生出一个Line类。Line类增加一个数据成员EndPoint计算线的长度。试比较一下与直接使用Point类来构造Line类的不同之处。

1.要点分析

无。

2.程序源码

#include<iostream>

#include<cstring>

#include<cmath>

using namespace std;

class Point

{

public:

double x;

double y;

void setPoint(double a, double b)

{

x = a;

y = b;

}

int getx()

{

return x;

}

int gety()

{

return y;

}

void show()

{

cout << "[" << x << "," << y << "]";

}

};

class Line :public Point

{

double length;

double x1, x2, y1, y2;

Point point1, point2;

public:

Line()

{

point1.setPoint(0, 0);

point2.setPoint(0, 0);

}

Line(double a, double b, double c, double d)

{

SetLine(a, b, c, d);

}

void SetLine(double a, double b, double c, double d)

{

point1.setPoint(a, b);

point2.setPoint(c, d);

}

void EndPoint()

{

x1 = point1.getx();

y1 = point1.gety();

x2 = point2.getx();

y2 = point2.gety();

length = sqrt((x1 - x2)\*(x1 - x2)\*1.0 + (y1 - y2)\*(y1 - y2)\*1.0);

}

void show()

{

cout << "\n两个点之间的直线距离为" << length << endl;

}

};

int main()

{

double a, b, c, d;

Point p, q;

cout << "请输入第一个点的横纵坐标：";

cin >> a >> b;

cout << "请输入第二个点的横纵坐标：";

cin >> c >> d;

p.setPoint(a, b);

q.setPoint(c, d);

cout << "您输入的两个点为：";

p.show();

cout << "\t";

q.show();

Line f;

f.SetLine(a, b, c, d);

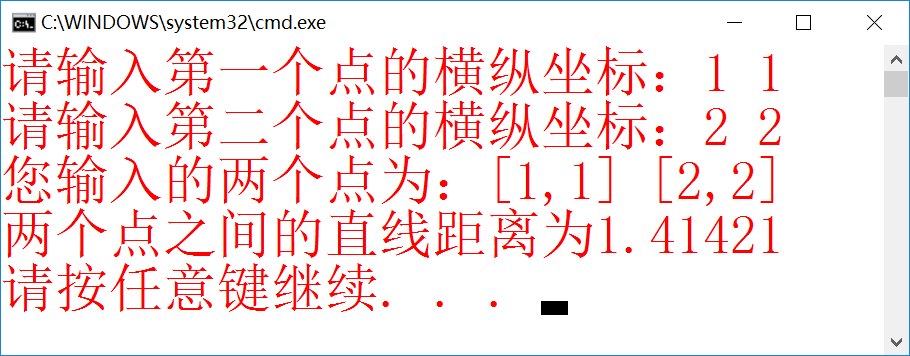
f.EndPoint();

f.show();

return 0;

}

3.实验结果



题目四 从Date类和Time类派生出一个DateAndTime类，修改相应的成员函数，当时间递增到新的一天时，能够修改日期值。

1.要点分析

2.程序源码

#include<iostream>

#include<cmath>

using namespace std;

class Date

{

protected:

int year, month, day;

public:

Date(int a, int b, int c) { year = a; month = b; day = c; }

void init(int yy, int mm, int dd)

{

month = (mm >= 1 && mm <= 12) ? mm : 0;

year = (yy >= 1900 && yy <= 2100) ? yy : 1900;

day = (dd >= 1 && dd <= 31) ? dd : 0;

}

void print\_ymd() { cout << year << "-" << month << "-" << day; };

};

class Time

{

protected:

int hour, minute, second;

public:

Time(int h, int m, int s) { hour = h, minute = m, second = s; }

void Register(int h = 0, int m = 0, int s = 0) { hour = h, minute = m, second = s; }

void print\_hms() { cout << "," << hour << "时" << minute << "分" << second << "秒" << endl; }

};

class DateAndTime :public Date, public Time

{

public:

DateAndTime(int a = 1900, int b = 1, int c = 1, int d = 0, int e = 0, int f = 0) :

Date(a, b, c), Time(d, e, f) {}

void input(int yy, int mm, int dd, int h, int m, int s)

{

hour = h, minute = m, second = s; month = mm; year = yy; day = dd;

}

void Normal()

{

int n;

if (second >= 60) { n = second / 60; second = second % 60; minute = minute + n; }

if (second<0) { n = second / 60 + 1; second = second % 60 + 60; minute = minute - n; }

if (minute >= 60) { n = minute / 60; minute = minute % 60; hour = hour + n; }

if (minute<0) { n = minute / 60 + 1; minute = minute % 60 + 60; hour = hour - n; }

if (hour >= 24) { n = hour / 24; hour = hour % 24; day = day + n; }

if (hour<0) { n = hour / 24 + 1; hour = hour % 24 + 24; day = day - n; }

if (day >= 30) { n = day / 30; day = day % 30; month = month + n; }

if (day<0) { n = day / 30 + 1; day = day % 30 + 30; month = month - n; }

if (month >= 12) { n = month / 12; month = month % 12; year = year + n; }

if (month<0) { n = month / 12 + 1; month = month % 12 + 12; year = year - n; }

}

void add(DateAndTime &d)

{

year = year + d.year;

month = month + d.month;

day = day + d.day;

hour = hour + d.hour;

minute = minute + d.minute;

second = second + d.second;

}

void sub(DateAndTime &d)

{

year = year - d.year;

month = month - d.month;

day = day - d.day;

hour = hour - d.hour;

minute = minute - d.minute;

second = second - d.second;

}

void show() { Normal(); print\_ymd(); print\_hms(); }

};

int main()

{

DateAndTime d1, d2;

cout << "初始化：" << endl; d1.show(); d2.show();

d1.input(2017, 4, 1, 14, 14, 14);

d2.input(1998, 8, 9, 20, 18, 55);

cout << "第一个时间："; d1.show();

cout << "第二个时间："; d2.show();

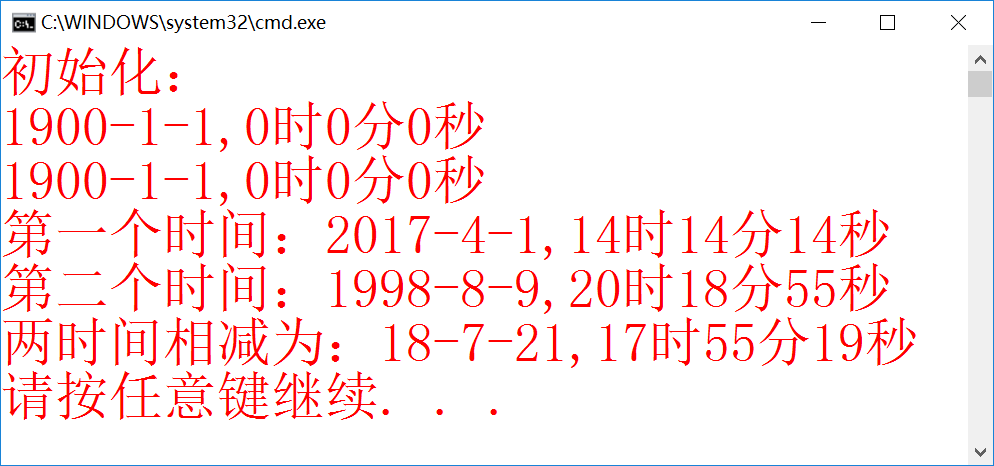
cout << "两时间相减为：";

d1.sub(d2); d1.show();

return 0;

}

3.实验结果



题目五 一元四次方程

1.要点分析

无。

2.程序源码

#include<iostream>

#include<cmath>

using namespace std;

class sanci

{

public:

double a, b, c, d, y;

double f1, f2;

int l = 0; //设定好循环迭代的最多次数

void setsan(double A, double B, double C, double D, double Y)

{

a = A;

b = B;

c = C;

d = D;

y = Y;

}

void qiugen()

{

f1 = d + c\*y + b\*y\*y + a\*pow(y, 3); //f1为函数值

f2 = c + 2 \* b\*y + 3 \* a\*y\*y; //f2为导数值

while (fabs(f1) >= 0.0000001)

{

y = y - f1 / f2; //求根

f1 = d + c\*y + b\*y\*y + a\*y\*y\*y;

f2 = c + 2 \* b\*y + 3 \* a\*y\*y;

l++; //次数+1

if (l>1000000) //次数达到1000000次，停止

{

cout << "找不到您要的解，令根为0\n";

y = 0;

f1 = 0;

}

}

}

void show1()

{

{

if (a>0)

{

if (a == 1)

cout << "x^3+";

if (a != 1)

cout << a << "x^3+";

}

if (a<0)

{

if (a == -1)

cout << "-x^3+";

if (a != -1)

cout << a << "x^3+";

}

}

{

if (b>0)

{

if (b == 1)

cout << "x^2+";

if (b != 1)

cout << b << "x^2+";

}

if (b<0)

{

if (b == -1)

cout << "\b" << "x^2+";

if (b != -1)

cout << "\b" << b << "x^2+";

}

}

{

if (c>0)

{

if (c == 1)

cout << "x+";

if (c != 1)

cout << c << "x+";

}

if (c<0)

{

if (c == -1)

cout << "\b" << "x+";

if (c != -1)

cout << "\b" << c << "x+";

}

}

{

if (d>0)

cout << d << "=0";

if (d == 0)

{

if (c == 0 && b == 0 && a == 0)

cout << "\b0=0";

else

cout << "\b=0";

}

if (d<0)

cout << "\b" << d << "=0";

}

}

void show2()

{

if (d != 0 && c == 0 && b == 0 && a == 0)

cout << "方程无解";

if (d == 0 && c == 0 && b == 0 && a == 0)

cout << "方程有无数组解";

else

cout << "方程的根为：" << y;

}

void plus1(sanci&sa)

{

a = a + sa.a;

b = b + sa.b;

c = c + sa.c;

d = d + sa.d;

}

};

class sici :public sanci //公有继承

{

public:

double e;

void setsi(double A, double B, double C, double D, double E, double Y)

{

setsan(A, B, C, D, Y);

e = E;

}

void plus2(sici&si)

{

e = e + si.e; //加法，系数相加减

a = a + si.a;

b = b + si.b;

c = c + si.c;

d = d + si.d;

}

void qiugen()

{

l = 0;

f1 = d + c\*y + b\*y\*y + a\*y\*y\*y + e\*pow(y, 4);

f2 = c + 2 \* b\*y + 3 \* a\*y\*y + 4 \* e\*pow(y, 3);

while (fabs(f1) >= 0.0000001)

{

y = y - f1 / f2;

f1 = d + c\*y + b\*y\*y + a\*pow(y, 3) + e\*pow(y, 4);

f2 = c + 2 \* b\*y + 3 \* a\*y\*y + 4 \* e\*pow(y, 3);

l++;

if (l>10000)

{

cout << "找不到您要的解，令根为0\n";

y = 0;

f1 = 0;

}

}

}

void show3()

{

{

if (e>0)

{

if (e == 1)

cout << "x^4+";

if (e != 1)

cout << a << "x^4+";

}

if (e<0)

{

if (e == -1)

cout << "-x^4+";

if (e != -1)

cout << e << "x^4+";

}

}

show1();

}

void show4()

{

if (d != 0 && c == 0 && b == 0 && a == 0 && e == 0)

cout << "方程无解";

if (e == 0 && d == 0 && c == 0 && b == 0 && a == 0)

cout << "方程有无数组解";

else

cout << "方程的根为：" << y;

}

};

int main()

{

double a, b, c, d, y, a1, b1, c1, d1, y1;

sanci haha1, haha2;

cout << "三次方程求解：\n";

cout << "请按照次数从高到低的顺序输入第一个方程各项系数：";

cin >> a >> b >> c >> d;

cout << "请输入希望与根最近的数字：";

cin >> y;

haha1.setsan(a, b, c, d, y);

haha1.qiugen();

haha1.show1();

haha1.show2();

cout << endl;

cout << "请按照次数从高到低的顺序输入第二个方程各项系数：";

cin >> a1 >> b1 >> c1 >> d1;

cout << "请输入希望与根最近的数字：";

cin >> y1;

haha2.setsan(a1, b1, c1, d1, y1);

haha2.qiugen();

haha2.show1();

haha2.show2();

cout << "\n两个方程相加：\n";

haha1.plus1(haha2);

haha1.qiugen();

haha1.show1();

haha1.show2();

cout << endl;

cout << "四次方程求解：\n";

double a2, b2, c2, d2, e2, y2, a3, b3, c3, d3, e3, y3;

sici hehe1, hehe2;

cout << "请按照次数从高到低的顺序输入第一个方程各项系数：";

cin >> e2 >> a2 >> b2 >> c2 >> d2;

cout << "请输入希望与根最近的数字：";

cin >> y2;

hehe1.setsi(a2, b2, c2, d2, e2, y2);

hehe1.qiugen();

hehe1.show3();

hehe1.show4();

cout << endl;

cout << "请按照次数从低到高的顺序输入第二个方程各项系数：";

cin >> e3 >> a3 >> b3 >> c3 >> d3;

cout << "请输入希望与根最近的数字：";

cin >> y3;

hehe2.setsi(a3, b3, c3, d3, e3, y3);

hehe2.qiugen();

hehe2.show3();

hehe2.show4();

cout << "\n两个方程相加：\n";

hehe1.plus2(hehe2);

hehe1.qiugen();

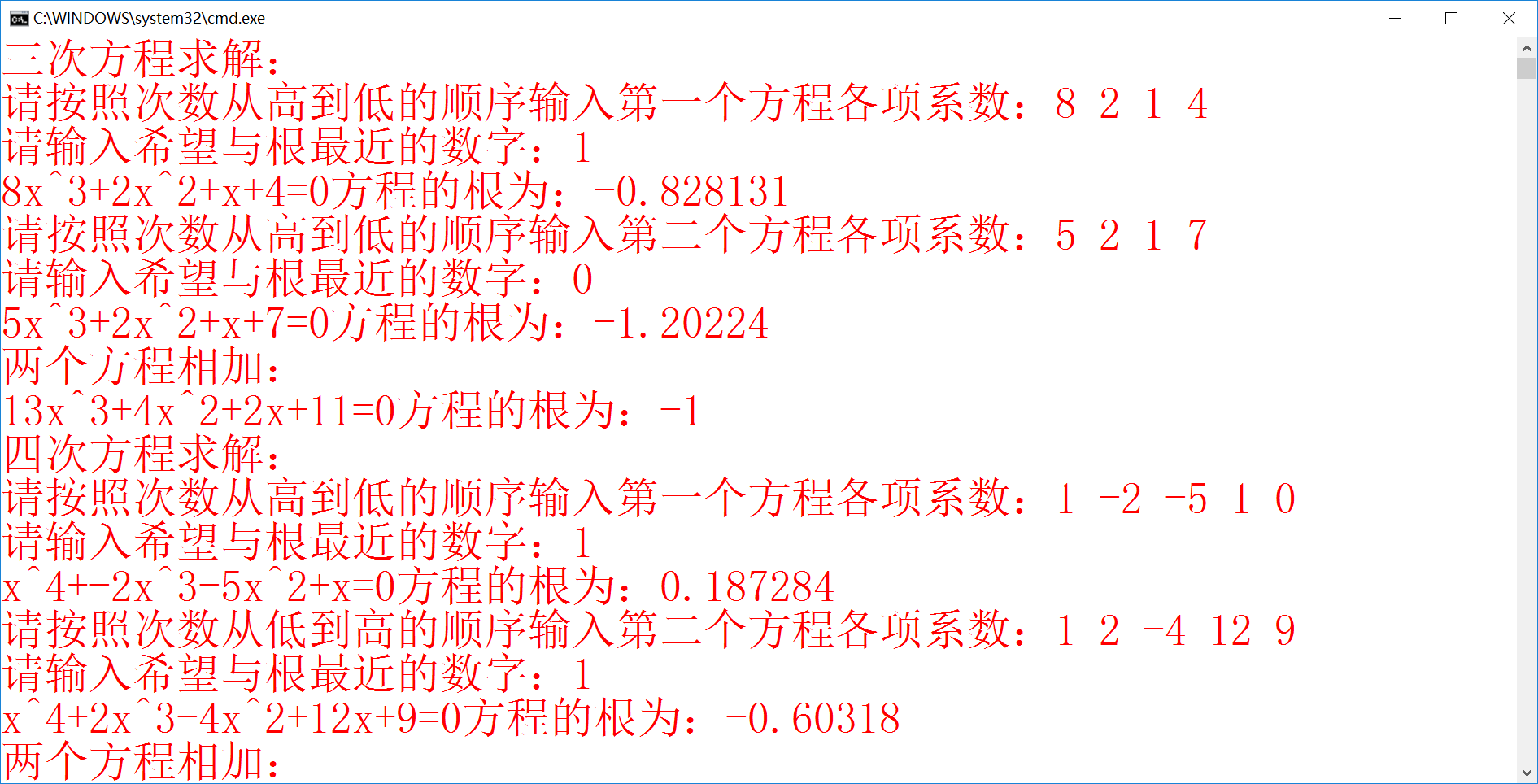
hehe1.show3();

hehe1.show4();

return 0;

}

3.实验结果



题目六 蛋糕

1.要点分析

重载函数；派生类的覆盖。

2.程序源码

#include<iostream>

#include<cstring>

#include<cmath>

using namespace std;

class yuan //定义圆类

{

protected:

double h1, r;

char zhuci[50];

public:

yuan(double H1, double R, char\*Zhuci)

{

init(H1, R, Zhuci);

}

void init(double H1, double R, char\*Zhuci)

{

h1 = H1;

r = R;

strcpy(zhuci, Zhuci);

}

double volume() //求体积

{

return 3.14\*r\*r\*h1;

}

double area()//求表面积

{

return 2 \* 3.14\*r\*r + 3.14 \* 2 \* r\*h1;

}

double V() //求总体积

{

return 3.14\*r\*r\*h1;

}

void output()

{

cout << "蛋糕助词为：" << zhuci;

cout << "\n圆柱形蛋糕信息如下：\n";

cout << "半径：" << r << "\t高：" << h1 << "\t体积：" << volume() << "\t表面积：" << area();

cout << "\n总体积：" << V();

}

};

class chang :public yuan //长方继承圆类

{

protected:

double a, h2;

public:

chang(double H1, double R, char\*Zhuci, double A, double H2) :yuan(H1, R, Zhuci)

{

a = A;

h2 = H2;

}

void init(double H1, double R, char\*Zhuci, double A, double H2)

{

yuan::init(H1, R, Zhuci); //同名函数，利用yuan：：来调用

a = A;

h2 = H2;

}

double volume()

{

return a\*a\*h2;

}

double area()

{

return a \* 4 \* h2 + 2 \* a\*a;

}

double V()

{

return a\*a\*h2 + 3.14\*r\*r\*h1;

}

void output()

{

yuan::output();

cout << "\n方柱形蛋糕信息如下：\n";

cout << "正方形边长：" << a << "\t高:" << h2 << "\t体积：" << volume() << "\t表面积：" << area();

cout << "\n总体积：" << V();

}

};

class ling :public chang

{

protected:

double b, c, h3;

public:

ling(double H1, double R, char\*Zhuci, double A, double H2, double B, double C, double H3) :chang(H1, R, Zhuci, A, H2)

{

b = B;

c = C;

h3 = H3;

}

void init(double H1, double R, char\*Zhuci, double A, double H2, double B, double C, double H3)

{

chang::init(H1, R, Zhuci, A, H2);

b = B;

c = C;

h3 = H3;

}

double volume()

{

return h3\*0.5\*b\*c;

}

double area()

{

return b\*c + sqrt(0.25\*b\*b + 0.25\*c\*c) \* 4 \* h3;

}

double V()

{

return h3\*0.5\*b\*c + a\*a\*h2 + 3.14\*r\*r\*h1;

}

void output()

{

chang::output();

cout << "\n菱柱形蛋糕信息如下：\n";

cout << "菱形长轴：" << b << "\t短轴：" << c << "\t体积：" << volume() << "\t表面积：" << area();

cout << "\n总体积：" << V();

}

};

int main()

{

char Zhuci[50];

double H1, R, A, H2, B, C, H3;

cout << "请输入蛋糕的祝词：\n";

cin.get(Zhuci, 50);

cout << "请依次输入圆柱形蛋糕的高度、半径;方柱形蛋糕的正方形边长、高度;菱柱形蛋糕的菱形长轴、短轴、高度：\n";

cin >> H1 >> R >> A >> H2 >> B >> C >> H3;

ling dangao(0, 0, "0", 0, 0, 0, 0, 0);

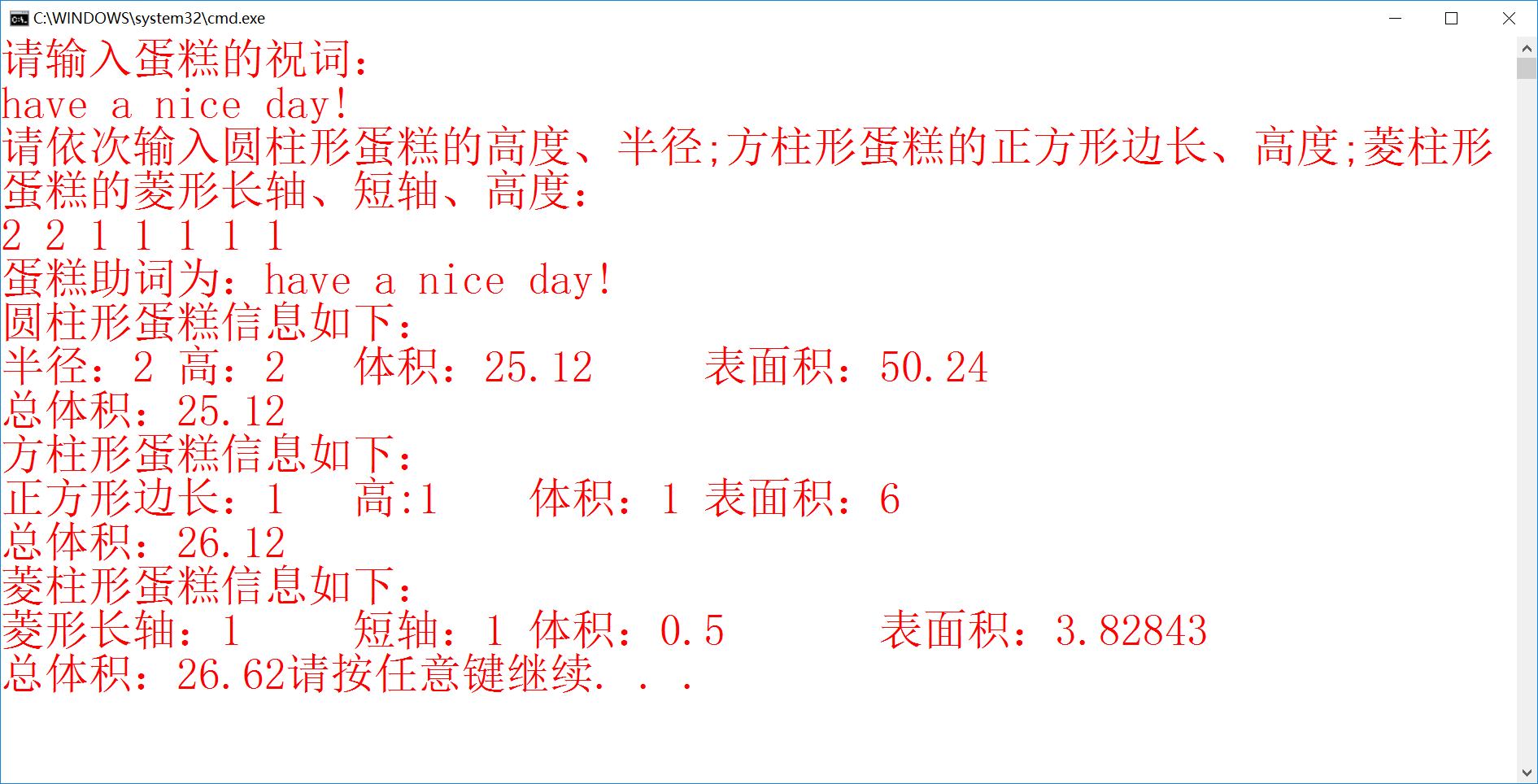
dangao.init(H1, R, Zhuci, A, H2, B, C, H3);

dangao.output();

return 0;

}

3.实验结果



三、个人小结

本周学习了继承。至此，面向对象的语言特色也基本展现了它的全部特点。虽然我不能完全理解这其中的精髓，但我相信掌握方法，一定终会逐步逼近核心。

我拜读了曾泓泰同志的上周的实验报告。所谓优秀难，找优秀的茬更难。事实上，每周我的分析对比都最后变成鸡蛋里挑骨头，从大佬的实验报告里找点不尽人意的地方实在是种折磨。而看到曾泓泰报告的分析，我服气了。论其字数，整整一页半；论其思想境界，实在是做到了重走一遍高年级学长的心路历程，从程序到方法，条条分析，优劣得所。

我喜欢c++，时日无几，好好珍惜。