西安交通大学实验报告

课程 计算机程序设计 实验名称 多态 第 1 页 共 28 页

系 别 工科试验班(钱学森实验班) 实 验 日 期 2017 年5 月 18 日

专业班级 钱学森62班 组别 无 实 验 报 告 日 期 2017 年5 月 22 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.理解多态概念。

2.学会使用虚函数。

3.了解运算符重载。

二、实验题目

题目一 定义一个类base，该类含有虚函数display，然后定义他的两个派生类FirstB和SecondB，这两个类均含有公有成员函数display。在主程序中，定义指向基类base的指针变量ptr，并分别定义base、FirstB、SecondB的对象b1、f1、s1的起始地址，然后执行这些对象的成员函数display。

1.要点分析

定义类及虚函数；定义派生类。

2.程序源码

#include<iostream>

using namespace std;

class base

{

public:

virtual void display()

{

cout << "假猪套天下第一！" << endl;

}

};

class FirstB :public base

{

public:

void display()

{

cout << "大师兄说得对！" << endl;

}

};

class SecondB :public base

{

public:

void display()

{

cout << "二师兄说得对！" << endl;

}

};

int main()

{

base b1, \*ptr;

FirstB f1;

SecondB s1;

b1.display();

ptr = &f1;

ptr->display();

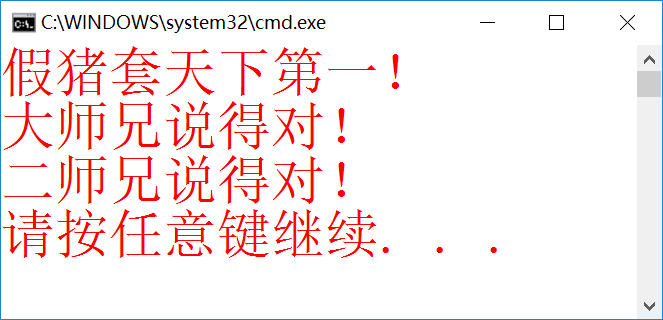
ptr = &s1;

ptr->display();

return 0;

}

3.实验结果



题目二 扩充例9-5，从中派生出一个正方形类（一个顶点和边长）和圆柱体类（圆和高），写一个测试程序，输出正方形的面积和圆柱体的体积。

1.要点分析

增加顶点边长；构造形体线度虚函数；定义派生类。

2.程序源码

#include<iostream>

using namespace std;

class point

{

int x, y;

public:

point(int a, int b)

{

setprint(a, b);

}

void setprint(int a, int b) { x = a; y = b; };

int getx()

{

return x;

}

int gety()

{

return y;

}

virtual void print()

{

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

}

virtual double area() { return 0; };

};

class circle :public point

{

double r;

public:

circle(int a = 0, int b = 0, double c = 0) :point(a, b) { r = c; }

void setbanjin(double s)

{

r = (s >= 0 ? s : 0);

}

double getr()

{

return r;

}

virtual double area()

{

return 3.14159\*r\*r;

}

void print()

{

cout << " 圆心坐标";

point::print();

cout << "； 半径为:" << r << endl;

cout << " 面积为：" << area() << endl;

}

};

class square : public point

{

int a;

public:

square(int aa = 0, int bb = 0, int c = 2) :point(aa, bb)

{

a = c;

}

void print()

{

cout << " 正方形的边长为：" << a << " 顶点为：";

point::print();

cout << " 正方形面积为：" << area() << endl;

}

void setbian(int b)

{

a = b;

}

int getbian()

{

return a;

}

double area()

{

return a\*a;

}

};

class yuanzhu :public circle

{

int h;

public:

yuanzhu(int a = 0, int b = 0, int c = 0, int d = 0) :circle(a, b, c)

{

h = d;

}

void set(int f)

{

h = f;

}

void print()

{

cout << " 圆柱的顶面的圆心为:";

circle::print();

cout << " 圆柱的高为：" << h;

cout << endl;

cout << " 圆柱的体积为：" << area()\*h;

}

};

int main()

{

point A(2, 3), \*p;

square square1(0, 0, 3);

yuanzhu yuanzhu1(0, 0, 3, 2);

p = &square1;

p->print();

cout << " 圆柱的信息为：" << endl;

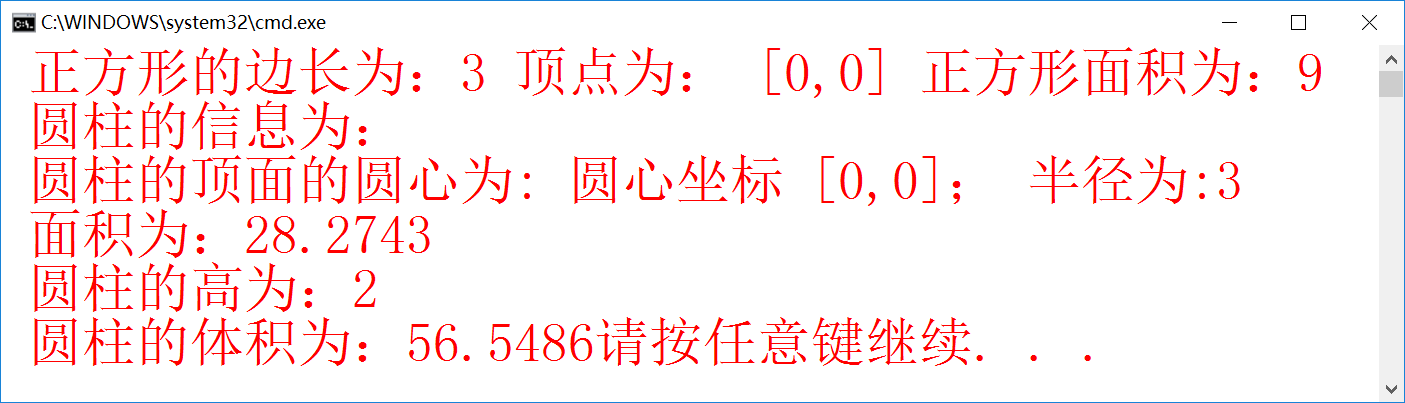
p = &yuanzhu1;

p->print();

return 0;

}

3.实验结果



题目三 扩充实例编程中的日期类，为Date类增加一个个成员函数，可以判断一个日期是否是系统当前日期，，从键盘输入你的生日，如果今天是你的生日，则显示：“Happy Birthday！”，否则显示“还有xx天是你的生日”或“你的生日已经过去了xx天，明年的生日还要再等yy天”。

1.要点分析

重载运算符+、-，用来进行日期比较。

2.程序源码

#include <iostream>

#include <ctime>

using namespace std;

class date {

int day, month, year;

void incday();

static const int days[];

public:

int daycalc() const;

date(int y, int m, int d);

date(int m, int d);

date();

void systemdate();

void setdate(int yy, int mm, int dd);

void setdate(int mm, int dd);

bool isleapyear(int yy)const;

bool isendofmonth() const;

friend bool panduan(date d1, date d2);

void print\_ymd()const;

void print\_mdy()const;

const date &operator +(int days);

const date &operator +=(int days);

int operator -(const date& ymd)const;

};

const int date::days[] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };

date::date(int y, int m, int d) { setdate(y, m, d); }

date::date(int m, int d) { setdate(m, d); }

date::date() { systemdate(); }

void date::systemdate() {

tm \*gm;

time\_t t = time(NULL);

gm = gmtime(&t);

year = 1900 + gm->tm\_year;

month = gm->tm\_mon + 1;

day = gm->tm\_mday;

}

void date::setdate(int yy, int mm, int dd) {

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

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

if (month == 2 && isleapyear(year)) { day = (dd>1 && dd<29) ? dd : 1; }

else day = (dd >= 1 && dd <= days[month]) ? dd : 1;

}

void date::setdate(int mm, int dd) {

tm \*gm;

time\_t t = time(NULL);

gm = gmtime(&t);

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

year = 1900 + gm->tm\_year;

if (month == 2 && isleapyear(year)) {

day = (dd >= 1 && dd <= 29) ? dd : 1;

}

else day = (dd >= 1 && dd <= days[month]) ? dd : 1;

}

const date &date::operator +(int days) {

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

incday();

return \*this;

}

}

const date &date::operator +=(int days) {

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

incday();

return \*this;

}

}

int date::operator -(const date &ymd) const {

int days;

days = daycalc() - ymd.daycalc();

return days;

}

bool date::isleapyear(int y) const {

if (y % 400 == 0 || (y % 100 != 0 && y % 4 == 0)) return true;

return false;

}

bool date::isendofmonth() const {

if (month == 2 && isleapyear(year)) {

return day == 29;

}

else return day == days[month];

}

bool panduan(date d1, date d2) {//////////////////////////////////////////////////

if (d2.day == d1.day&&d2.month == d1.month&&d2.year == d1.year) {

return true;

}

else return false;

}

void date::incday() {

if (isendofmonth())

if (month == 12) {

day = 1, month = 1, year++;

}

else {

day = 1; month++;

}

else day++;

}

int date::daycalc() const {

int dd;

int yy = year - 1900;

dd = yy \* 365;

if (yy) dd += (yy - 1) / 4;

for (int i = 1; i<month; i++) dd += days[i];

if (isleapyear(year) && (month>2)) dd++;

dd += day;

return dd;

}

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

void date::print\_mdy() const {

char\*monthname[12] = { "January","February","March","April","May","June","July","August","September","October","November","December" };

cout << monthname[month - 1] << ' ' << day << "," << year << endl;

}

int main() {

date today;

cout << "今天的日期是：";

today.print\_ymd();

int myday, mymonth, myyear;

cout << "请输入你的生日(按照年月日格式)：";

cin >> myyear >> mymonth >> myday;

date birthday(myyear, mymonth, myday);

if (panduan(today, birthday)) {

cout << "Happy birthday!" << endl;

}

else {

if (today.daycalc()>birthday.daycalc()) {

cout << "你的生日已经过去了" << today - birthday << "天,明年的生日要再等";

int ddd = birthday - today;

ddd += 365;

cout << ddd << "天." << endl;

}

else

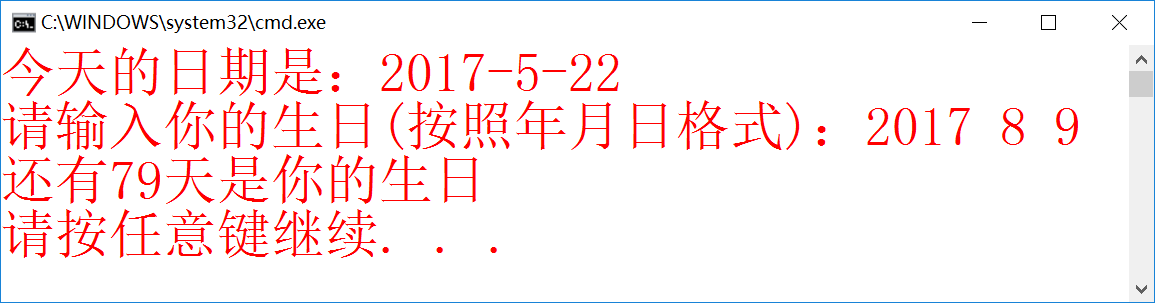
cout << "还有" << birthday - today << "天是你的生日" << endl;

}

return 0;

}

3.实验结果



题目四 完成上一章实验第5题红颜色的函数功能。

1.要点分析

重载>函数。

2.程序源码

#include<iostream>

#include<string.h>

using namespace std;

class cake

{

private:

int r, h;

char \*c;

public:

cake() { r = 0; h = 0; c = new char[strlen("XXXXXX") + 1]; strcpy(c, "XXXXXX"); }

void init(int r1, int h1, char \*c1)

{

c = new char[strlen(c1) + 1];

strcpy(c, c1);

r = r1;

h = h1;

}

float volumn()

{

float rr = r, hh = h; return 3.14\*rr\*rr\*hh;

}

float area()

{

float rr = r, hh = h; return 2 \* 3.14\*rr\*rr + 2 \* 3.14\*rr\*hh;

}

void output()

{

cout << " 祝福语：" << c << endl;

cout << " 圆柱形蛋糕信息如下：" << endl;

cout << " 半径：" << r << ' ' << " 高度：" << h << ' ' << " 体积：" << volumn() << endl;

}

~cake() { delete[]c; }

};

class yzcake :public cake

{

private:

int a, h2;

public:

yzcake() { a = 0; h2 = 0; }

void init(int a1, int h21)

{

a = a1; h2 = h21;

}

float volumn()

{

float aa = a, h2h = h2; return aa\*aa\*h2h;

}

void output()

{

cout << " 正方体蛋糕信息如下：" << endl;

cout << " 边长：" << a << ' ' << " 高度：" << h2 << ' ' << " 体积：" << volumn() << endl;

}

~yzcake() {}

};

class yzlcake :public yzcake

{

private:

int A, B, H;

public:

yzlcake() { A = 0; B = 0; H = 0; }

void init(int A1, int B1, int H1)

{

A = A1; B = B1; H = H1;

}

float volumn()

{

float AA = A, BB = B, HH = H; return AA\*BB\*HH / 2;

}

void output()

{

cout << " 菱形体蛋糕信息如下：" << endl;

cout << " 长轴长：" << A << ' ' << " 短轴长：" << B << ' ' << " 高度：" << H << ' ' << " 体积：" << volumn() << endl;

}

float aaa()

{

return volumn() + yzcake::volumn() + cake::volumn();

}

int operator>(yzlcake u)

{

if (aaa()>u.aaa())return 1;

if (aaa() == u.aaa())return 0;

if (aaa()<u.aaa())return -1;

}

~yzlcake() {}

};

int main()

{

cout << " 请输入蛋糕：祝词、半径、高、方形边长、高、菱形长轴、短轴、高" << endl;

char\*He, he[20], p[3] = { 'h','h','h' };

cin.get(he, 20);

int r1, h1, l, h2, a, b, h3;

cin >> r1 >> h1 >> l >> h2 >> a >> b >> h3;

He = new char[strlen(he) + 1];

strcpy(He, he);

cout << " 三层蛋糕信息如下：" << endl;

cake c1, c11;

c1.init(r1, h1, He);

c1.output();

yzcake c2, c22;

c2.init(l, h2);

c2.output();

yzlcake c3, c33;

c3.init(a, b, h3);

c3.output();

c11.init(3, 2, p);

c22.init(3, 2);

c33.init(2, 1, 2);

switch (c3>c33)

{

case 1:

cout << " 第一个蛋糕大" << endl;

break;

case 0:

cout << " 两个蛋糕一样大" << endl;

break;

case -1:

cout << " 第二个蛋糕大" << endl;

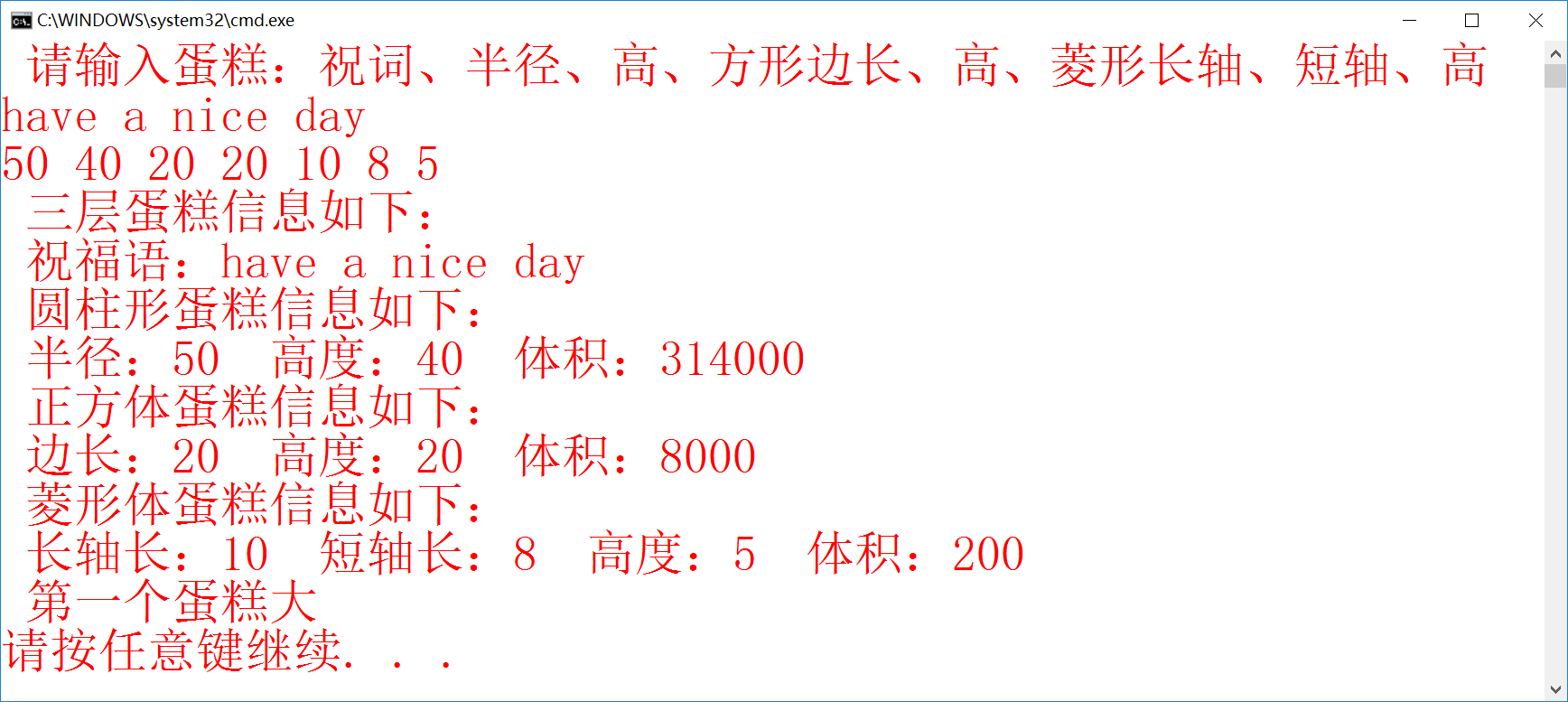
break;

}

return 0;

}

3.实验结果



题目五 定义一个一元二次方程类，通过继承方式定义一元三次方程类，再继承定义一元四次方程类。类中至少包含构造函数、求根函数、运算符 + 重载函数、运算符 - 重载函数、运算符==重载函数、输出方程的函数等6 个函数，并编写主函数测试各成员函数。提示： ① 两个一元三次方程对应相加仍然是一个一元三次方程； ② 求根方法采用迭代方法，迭代公式为： Xn+1=Xn － F(Xn)/F ’ (Xn) ，结束迭代的条件 |F(Xn+1)|<10-7 与 |Xn+1-Xn|<10-7 同时成立； ③ 一元三次方程的一般形式如下： F(X)=AX3+BX2+CX+D=0 。输出方程格式为： A\*X^3+B\*X^2+C\*X+D=0 ；④两个一元三次方程对应相加和对应相减仍然是一元三次方程。⑤假定类中的方程系数能求解出实根。不考虑方程存在虚根和无根的情况。求根函数应该有一个参数，该参数指明迭代初值。例如方程 2X3-4X2+3x-6 ＝ 0 在 1.5 附近的根。又例如方程 X3+12X2+48X+64=0 在 -4.5 附近的根。例如方程X4-10X3+35X2-50x+24＝0在4.5附近的根。又例如方程X4+12X3+54X2+108X+81=0在-1.5附近的根。因此类中数据成员除了系数外，还应考虑迭代初值作为数据成员。

1.要点分析

方程相加函数；输出格式控制。

2.程序源码

#include <iostream>

#include <cmath>

using namespace std;

class three {

protected:

float m\_third, m\_second, m\_first, m\_zero;

float x, m\_original;

float value, diffvalue;

public:

three(float third, float second, float first, float zero, float original) {

if (!third) {

m\_third = m\_second = m\_first = m\_zero = 0;

}

else {

m\_zero = zero;

m\_first = first;

m\_second = second;

m\_third = third;

m\_original = original;

}

}

void helpcheck(float m, int t) {

if (t>1) {

if (m>0 && m != 1) { cout << '+' << m << "x^" << t; }

else if (!m) {}

else if (m == -1) { cout << '-' << "x^" << t; }

else if (m == 1) { cout << "x^" << t; }

else { cout << m << "x^" << t; }

}

else if (t == 1) {

if (m>0 && m != 1) { cout << '+' << m << 'x'; }

else if (!m) {}

else if (m == -1) { cout << '-' << 'x'; }

else if (m == 1) { cout << '+' << 'x'; }

else { cout << m << 'x'; }

}

else if (t == 0) {

if (m>0) { cout << '+' << m << "=0"; }

else if (!m) { cout << "=0"; }

else if (m<0) { cout << m << "=0"; }

}

}

void outequalition() {

cout << "该方程为";

cout << m\_third << "x^3";

helpcheck(m\_second, 2);

helpcheck(m\_first, 1);

helpcheck(m\_zero, 0);

}

void plusequalition(three eq1) {

m\_third += eq1.m\_third;

m\_second += eq1.m\_second;

m\_first += eq1.m\_first;

m\_zero += eq1.m\_zero;

}

float fun1() {

value = m\_third\*m\_original\*m\_original\*m\_original + m\_second\*m\_original\*m\_original + m\_first\*m\_original + m\_zero;

return value;

}

float fun2() {

diffvalue = m\_third \* 3 \* m\_original\*m\_original + m\_second \* 2 \* m\_original + m\_first;

return diffvalue;

}

void calx() {

x = m\_original - fun1() / fun2();

}

bool condition1() {

if (abs(value)>.0000001) {

return true;

}

else {

return false;

}

}

bool condition2() {

if (abs(x - m\_original)>.0000001) {

return true;

}

else {

return false;

}

}

void exchange() {

while (condition1() || condition2()) {

m\_original = x;

calx();

}

cout << "在给定值附近的解为" << m\_original;

}

};

class four :public three {

private:

float m\_forth;

public:

four(float forth, float third, float second, float first, float zero, float original) :m\_forth(forth), three(third, second, first, zero, original) {};

void foutequalition() {

cout << "该方程为";

if (m\_forth == 1) {

cout << "x^4";

}

else if (m\_forth == -1) {

cout << "-x^4";

}

else {

cout << m\_forth << "x^4";

}

helpcheck(m\_third, 3);

helpcheck(m\_second, 2);

helpcheck(m\_first, 1);

helpcheck(m\_zero, 0);

}

void fplusequalition(four feq1) {

m\_forth += feq1.m\_forth;

m\_third += feq1.m\_third;

m\_second += feq1.m\_second;

m\_first += feq1.m\_first;

m\_zero += feq1.m\_zero;

}

float ffun1() {

value = fun1() + m\_forth\*m\_original\*m\_original\*m\_original\*m\_original;

return value;

}

float ffun2() {

diffvalue = fun2() + 4 \* m\_forth\*m\_original\*m\_original\*m\_original;

return diffvalue;

}

void fcalx() {

x = m\_original - ffun1() / ffun2();

}

bool fcondition1() {

if (abs(value)>.0000001) {

return true;

}

else {

return false;

}

}

bool fcondition2() {

if (abs(x - m\_original)>.0000001) {

return true;

}

else {

return false;

}

}

void fexchange() {

while (fcondition1() || fcondition2()) {

m\_original = x;

fcalx();

}

cout << "在给定值附近的解为" << m\_original;

}

};

int main() {

four ff1(1, 12, 54, 108, 81, -1.5);

ff1.foutequalition();

cout << "初始值为" << -1.5 << endl;

ff1.fcalx();

ff1.fexchange();

four ff2(1, -10, 35, -50, 24, 4.5);

cout << endl;

ff2.foutequalition();

cout << "初始值为" << 4.5 << endl;

ff2.fcalx();

ff2.fexchange();

ff1.fplusequalition(ff2);

cout << endl;

cout << "两个方程相加：";

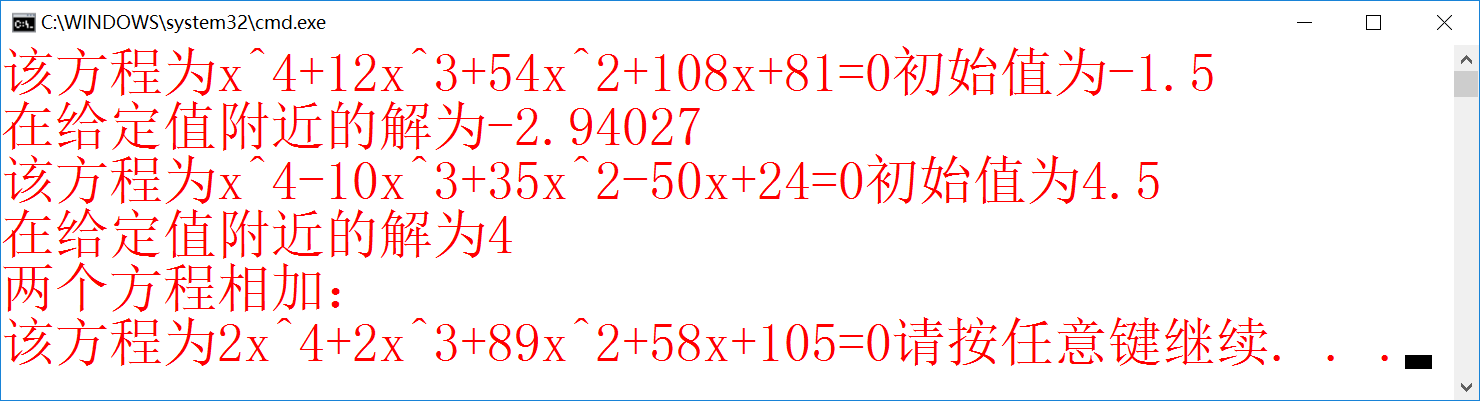
cout << endl;

ff1.foutequalition();

return 0;

}

3.实验结果



题目六 给自定义虚数类增加运算符\*、/、++、--、==重载，注意++和--分前置和后置两种情况，还要增加运算符+重载，即虚数类对象+分数类对象。编写主函数加以测试。

1.要点分析

前置和后置重载的定义。

2.程序源码

#include <iostream>

using namespace std;

class Complex

{

private:

double real, imag;

public:

Complex(double r = 0, double i = 0) : real(r), imag(i) { }

Complex operator +(Complex&);

Complex operator +(double);

Complex operator \*(Complex);

Complex operator / (Complex);

Complex &operator ++();

Complex operator ++(int);

Complex &operator --();

Complex operator --(int);

void out();

bool operator ==(Complex);

~Complex() { };

};

Complex Complex::operator + (Complex &c)

{

Complex temp;

temp.real = real + c.real;

temp.imag = imag + c.imag;

return temp;

}

Complex Complex::operator + (double d)

{

Complex temp;

temp.real = real + d;

temp.imag = imag;

return temp;

}

Complex Complex::operator \*(Complex a)

{

Complex temp;

temp.real = real\*a.real - imag\*a.imag;

temp.imag = real\*a.imag + imag\*a.real;

return temp;

}

Complex Complex::operator /(Complex a)

{

Complex temp;

if (a == 0)

{

temp.real = 1;

temp.imag = 0;

return temp;

}

else

{

temp.real = (real\*a.real + imag\*a.imag) / (a.real\*a.real + a.imag\*a.imag);

temp.imag = -(real\*a.imag + imag\*a.real) / (a.real\*a.real + a.imag\*a.imag);

return temp;

}

}

Complex &Complex::operator ++()

{

real++;

imag++;

return \*this;

}

Complex Complex::operator ++(int)

{

Complex temp = \*this;

real++;

imag++;

return temp;

}

Complex &Complex::operator --()

{

real--;

imag--;

return \*this;

}

Complex Complex::operator --(int)

{

Complex temp = \*this;

real--;

imag--;

return temp;

}

bool Complex::operator ==(Complex c)

{

if (real == c.real && imag == c.imag)

return true;

else

return false;

}

void Complex::out()

{

cout << real;

if (imag < 0)

{

if (imag == -1)

cout << "-i";

else

cout << imag << "i";

}

if (imag > 0)

{

if (imag == 1)

cout << "+i";

else

cout << "+" << imag << "i";

}

}

int main()

{

Complex c1(3, 4), c2(5, 6), c3;

cout << "c1=";

c1.out();

cout << " c2=";

c2.out();

cout << endl;

cout << "测试两个复数相加" << endl;

c3 = c1 + c2;

c3.out();

cout << endl;

cout << "测试一个复数加一个分数" << endl;

c3 = c3 + 6.5;

c3.out();

cout << endl;

cout << "测试两个复数相乘" << endl;

c3 = c1\*c2;

c3.out();

cout << endl;

cout << "测试两个复数相除" << endl;

c3 = c1 / c2;

c3.out();

cout << endl;

cout << "测试复数右++" << endl;

c1++;

c1.out();

cout << endl;

cout << "测试复数左++" << endl;

++c1;

c1.out();

cout << endl;

cout << "测试复数右--" << endl;

c2--;

c2.out();

cout << endl;

cout << "测试复数左--" << endl;

--c2;

c2.out();

cout << endl;

cout << "测试相等" << endl;

if (c1 == c2)

cout << "两个复数相等";

else

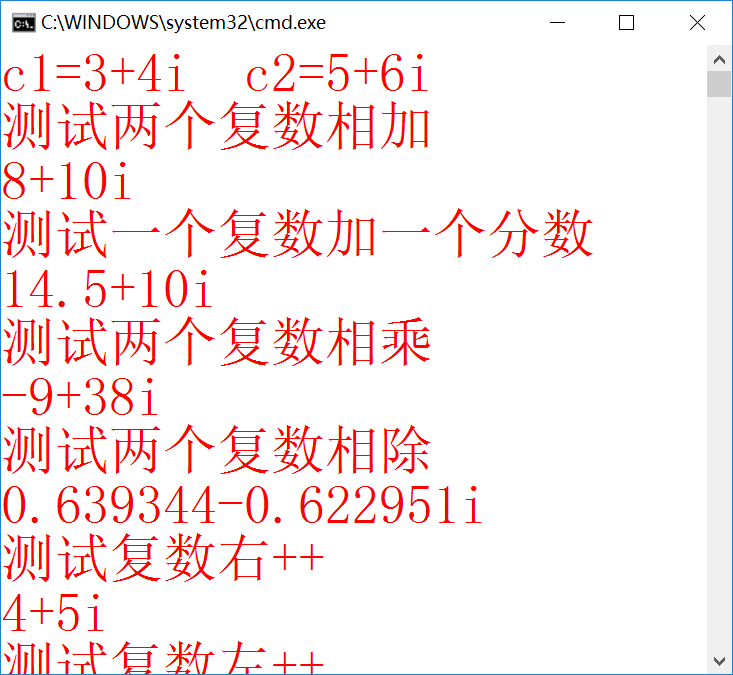
cout << "两个复数不相等";

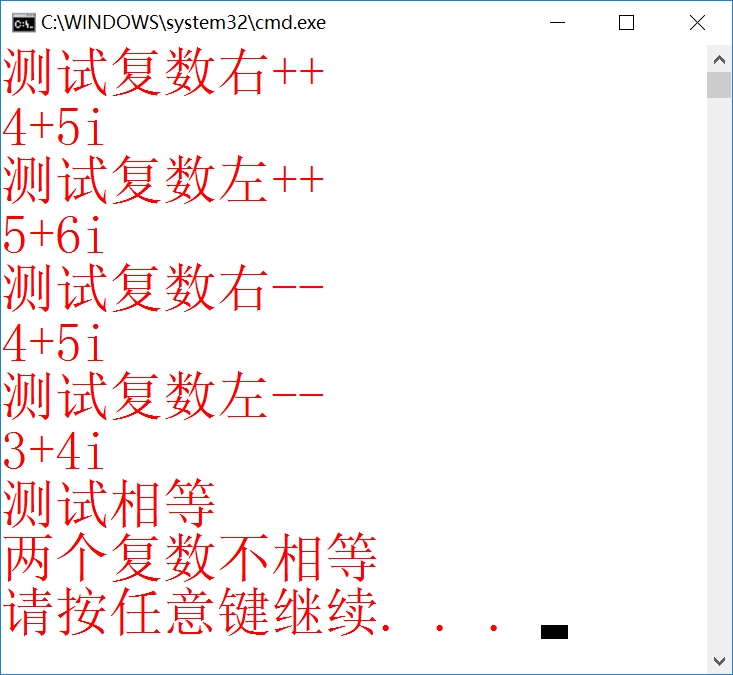
cout << endl;

return 0;

}

3.实验结果





题目七 完善上一章定义大整数类的题目，成员函数包括加、减以及判大小，注意用运算符重载，编写主函数加以测试。

1.要点分析

+、-符号的重载。

2.程序源码

#include<iostream>

#include<cstring>

using namespace std;

class number

{

int num[100];

char sign;

int s;

public:

number() {};

number(char\*, char);

number operator +(number);

number operator - (number);

void shuchu();

};

number::number(char \*str, char fuhao)

{

sign = fuhao;

s = strlen(str);

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

num[99 - i] = str[s - i - 1] - '0';

for (int i = s; i<100; i++)

num[99 - i] = 0;

}

number number::operator +(number numm)

{

if (sign == '+'&&numm.sign == '+' || sign == '-'&&numm.sign == '-')

{

for (int i = 99; i>0; i--)

{

int k = num[i] + numm.num[i];

num[i] = k % 10;

if (k>9)

num[i - 1]++;

}

}

else

{

if (s<numm.s)

{

for (int i = 99; i>0; i--)

{

numm.num[i] -= num[i];

if (numm.num[i]<0)

{

numm.num[i - 1]--;

numm.num[i] += 10;

}

num[i] = numm.num[i];

}

sign = numm.sign;

}

else if (s == numm.s)

{

int tmp = 1;

for (int i = 100 - s; i<100; i++)

{

if (num[i]<numm.num[i])

{

sign = numm.sign;

tmp = -1;

break;

}

}

if (tmp == 1)

{

for (int i = 99; i>0; i--)

{

num[i] -= numm.num[i];

if (num[i]<0)

{

num[i - 1]--;

num[i] += 10;

}

}

}

else

{

for (int i = 99; i>0; i--)

{

numm.num[i] -= num[i];

if (numm.num[i]<0)

{

numm.num[i - 1]--;

numm.num[i] += 10;

}

num[i] = numm.num[i];

}

sign = numm.sign;

}

}

else

{

for (int i = 99; i>0; i--)

{

num[i] -= numm.num[i];

if (num[i]<0)

{

num[i - 1]--;

num[i] += 10;

}

}

}

}

return \*this;

}

number number::operator -(number numm)

{

if (sign == '+'&&numm.sign == '-' || sign == '-'&&numm.sign == '+')

{

for (int i = 99; i>0; i--)

{

int k = num[i] + numm.num[i];

num[i] = k % 10;

if (k>9) { num[i - 1]++; }

}

}

else

{

if (s<numm.s)

{

for (int i = 99; i>0; i--)

{

numm.num[i] -= num[i];

if (numm.num[i]<0)

{

numm.num[i - 1]--;

numm.num[i] += 10;

}

num[i] = numm.num[i];

}

sign = (numm.sign == '-') ? '+' : '-';

}

else if (s == numm.s)

{

int tmp = 1;

for (int i = 100 - s; i<100; i++)

{

if (num[i]<numm.num[i])

{

sign = numm.sign;

tmp = -1;

break;

}

}

if (tmp == 1)

{

for (int i = 99; i>0; i--)

{

num[i] -= numm.num[i];

if (num[i]<0)

{

num[i - 1]--;

num[i] += 10;

}

}

}

else

{

for (int i = 99; i>0; i--)

{

numm.num[i] -= num[i];

if (numm.num[i]<0)

{

numm.num[i - 1]--;

numm.num[i] += 10;

}

num[i] = numm.num[i];

}

sign = (numm.sign == '-') ? '+' : '-';

}

}

else

{

for (int i = 99; i>0; i--)

{

num[i] -= numm.num[i];

if (num[i]<0)

{

num[i - 1]--;

num[i] += 10;

}

}

}

}

return \*this;

}

void number::shuchu()

{

cout << sign;

for (int i = 0; i<100; i++)

{

if (num[i] != 0)

{

s = 100 - i;

break;

}

else

s = 0;

}

if (s == 0)

cout << "\b0";

else

{

for (int i = 100 - s; i<100; i++)

cout << num[i];

}

cout << endl;

}

int main()

{

char a[100], b[100], c[100], p, q, r;

cout << "请输入100位以内的正整数a(带上正负号)：";

cin >> p >> a;

number num1(a, p);

cout << "数字a为：";

num1.shuchu();

cout << "请输入100位以内的正整数b(带上正负号)：";

cin >> q >> b;

number num2(b, q);

cout << "数字b为：";

num2.shuchu();

number num3 = num1 + num2;

cout << "c=a+b=";

num3.shuchu();

cout << "请输入100位以内的正整数e(带上正负号)：";

cin >> r >> c;

number num5(c, r);

cout << "数字e为：";

num5.shuchu();

number num4 = num3 - num5;

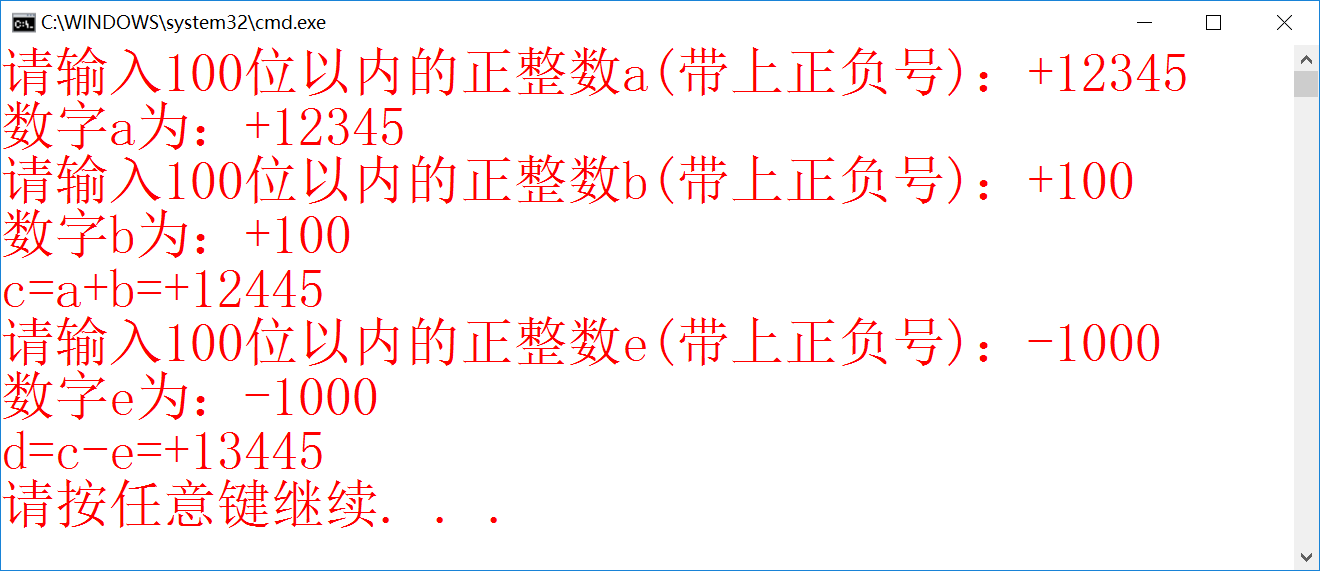
cout << "d=c-e=";

num4.shuchu();

return 0;

}

3.实验结果



三、实验小结

本周学习了多态的概念和相关应用。多态概念并不复杂，但要想透彻理解仍有不小的难度。经过若干题目的实践，据我目前的理解，多态的含义就是接口重用，其作用就在于降低升级或改动程序时的工作量。

同时阅读了上周宋睿同学写的的优秀实验报告。其优势主要体现在实验结果将各种讨论情况充分反馈，展现方式也较为多样（除了图片还包括录制的动画）；实验要点分析较为详尽，将题目结构拆解科学，没有矫情的凑字数步骤。