C++ Programming Practice Report

|  |  |
| --- | --- |
| Student ID | 18511510019 |
| Student Name | MOHAMED NUR |
| Practice No. |  |
| Practice Title |  |
| Date |  |
| Place |  |
| Mark |  |
| Checked by | Wingo WU |

# Overload operators

## Objects

1. To get a deep understanding of overloaded operators
2. To write a little complicate an practical class
3. To learn apply class in real situation

## Problems

### write complex class for complex in math

Read carefully the following program. In this program, it use a class named Complex to do complex operation. such as complex input(>>),complex output(<<), complex addition, subtraction, division and multiplication.

You are required to write the Complex class to finish this program, compile and run it and Insert the whole program and the results. You are not permitted to change this main program. But at first you may test your complex class in your own way. But Result you submit should use this main program.

int main()

{

Complex z1(3,5),z2(4.5,2.3),z3;

Complex z4;

Complex z5(z1);

Complex z6;

Complex z7;

Complex z8;

//double d;

cout<<"enter a complex(z1):";

cin>>z1;

cout<<"enter a complex(z2):";

cin>>z2;

cout<<" enter a complex(z3):";

cin>>z3;

cout<<"z1="<<z1<<endl;

cout<<"z2="<<z2<<endl;

cout<<"z3="<<z3<<endl;

z4=z1+z2;

cout<<"z1+z2="<<z4<<endl;

z5 = z1\*z2;

cout<<"z1\*z2:"<<z5<<endl;

z3+= (z2\*3)+2\*z1;

cout<<"after z3+=(z2\*3)+2\*z1 z3="<<z3<<endl;

z4= (z2/4.0)+5.0/z1;

cout<<"after z4= (z2/4.0)+5.0/z1 z4="<<z4<<endl;

z5= (z1/z2+5)/2;

cout<<"after z5= (z1/z2+5)/2 z5="<<z5<<endl;

if(z1==z5)

cout<<"equal"<<endl;

else

cout<<"not equal"<<endl;

z8=3\*z1+ (z1\*z2)\*3-(3\*z3/z4/2)+(z5+=z3)+(z4-z3)+(z5/z2);

cout<<z8<<endl;

return 0;

}

## Results

### 

[Your program in text format]

// p0.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include<iostream>

using namespace std;

class Complex {

private:

double m\_real;

double m\_imag;

public:

Complex(double real = 0.0, double imag = 0.0) :

m\_real(real), m\_imag(imag) {};

Complex(const Complex& c)

: m\_real(c.m\_real), m\_imag(c.m\_imag) { };

void SetReal(double real) { m\_real = real; };

double GetReal() { return m\_real; };

Complex operator -() const;

Complex& operator +=(const Complex&);

Complex& operator \*=(const Complex&);

Complex& operator /=(const Complex&);

Complex& operator -=(const Complex&);

Complex operator +(const Complex&) const;

Complex operator \*(const Complex&) const;

Complex operator /(const Complex&) const;

Complex operator -(const Complex&) const;

bool operator == (const Complex&) const;

friend istream& operator >>(istream& is, Complex&);

friend ostream& operator <<(ostream&, const Complex&);

friend Complex operator \*(const double ,const Complex&);

friend Complex operator /(const double, const Complex&);

friend Complex operator -(const double, const Complex&);

friend Complex operator +(const double, const Complex&);

};

istream& operator >>(istream & is, Complex& c) {

cout << "real imag"<< endl;

is >> c.m\_real >> c.m\_imag;

return is;

}

ostream& operator <<(ostream& os, const Complex& c) {

os << c.m\_real;

if (c.m\_imag < 0)

os << c.m\_imag << "i";

else if (c.m\_imag > 0)

os << "+" << c.m\_imag << "i";

return os;

}

Complex& Complex::operator +=(const Complex& c2) {

m\_real += c2.m\_real;

m\_imag += c2.m\_imag;

return \*this;

}

Complex Complex::operator+(const Complex& c2) const {

return Complex(\*this) += c2;

}

Complex& Complex::operator \*=(const Complex& c2) {

double real = m\_real \* c2.m\_real - m\_imag \* c2.m\_imag;

double imag = m\_real \* c2.m\_imag + m\_imag \* c2.m\_real;

m\_real = real;

m\_imag = imag;

return \*this;

}

Complex Complex::operator \*(const Complex& c2) const {

return Complex(\*this) \*= c2;

}

Complex operator \*(const double num, const Complex&c) {

return Complex(num, 0) \*= c;

};

Complex operator /(const double num, const Complex&c) {

return Complex(num, 0) /= c;

};

Complex& Complex::operator /=(const Complex& c2) {

Complex nm = Complex(\*this) \* Complex(c2.m\_real, -c2.m\_imag);

double dn = c2.m\_real \* c2.m\_real + c2.m\_imag \* c2.m\_imag;

m\_real = nm.m\_real / dn;

m\_imag = nm.m\_imag / dn;

return \*this;

}

Complex Complex::operator /(const Complex& c2) const {

return Complex(\*this) /= c2;

}

bool Complex::operator == (const Complex& c) const {

return (m\_real == c.m\_real) && (m\_imag == c.m\_imag);

}

Complex Complex::operator -() const {

return Complex(-m\_real, -m\_imag);

}

Complex& Complex::operator -=(const Complex& c2) {

return \*this += -c2;

}

Complex Complex::operator -(const Complex& c2) const {

return Complex(\*this) -= c2;

}

Complex operator -(const double num, const Complex&c) {

return Complex(num, 0) -= c;

};

Complex operator +(const double num, const Complex&c) {

return Complex(num, 0) += c;

};

int main()

{

Complex z1(3, 5), z2(4.5, 2.3), z3;

Complex z4;

Complex z5(z1);

Complex z6;

Complex z7;

Complex z8;

//double d;

cout << "enter a complex(z1):";

cin >> z1;

cout << "enter a complex(z2):";

cin >> z2;

cout << "enter a complex(z3):";

cin >> z3;

cout << "z1=" << z1 << endl;

cout << "z2=" << z2 << endl;

cout << "z3=" << z3 << endl;

z4 = z1 + z2;

cout << "z1+z2=" << z4 << endl;

z5 = z1\*z2;

cout << "z1\*z2:" << z5 << endl;

z3 += (z2 \* 3) + 2 \* z1;

cout << "after z3+=(z2\*3)+2\*z1 z3=" << z3 << endl;

z4 = (z2 / 4.0) + 5.0 / z1;

cout << "after z4= (z2/4.0)+5.0/z1 z4=" << z4 << endl;

if (z1 == z5)

cout << "equal" << endl;

else

cout << "not equal" << endl;

z8 = 3 \* z1 + (z1\*z2) \* 3 - (3 \* z3 / z4 / 2) + (z5 += z3) + (z4 - z3) + (z5 / z2);

cout << z8 << endl;

return 0;

}

[Your screen shot]

# 