# 实验五 运算符重载

姓 名：罗啸

学 号：2420173095

班 级：电子173

实验地点：电气学院412

指导老师：邝先验

# 实验五 运算符重载

## 5.1 实验目的

(1) 进一步了解运算符重载的概念和使用方法。

(2) 掌握几种常用的运算符重载的方法。

(3) 了解转换构造函数的使用方法。

(4) 了解在Vi s ual C++ 6.0 环境下进行运算符重载要注意的问题。

## 5.2 实验内容

### 5.2.1程序阅读

1．理解下面的程序，并在VC++6.0下运行查看结果，回答程序后面的问题。

#include <iostream>

using namespace std;

class CComplex

{

public:

CComplex()

{

real = 0;

imag = 0;

}

CComplex(int x,int y)

{

real = x;

imag = y;

}

int real;

int imag;

CComplex operator + (CComplex obj1)-----------------------------------------------①

{

CComplex obj2(real + obj1.real, imag + obj1.imag);

return obj2;}

};

void main()

{

CComplex obj1(100,30);

CComplex obj2(20, 30);

CComplex obj;

obj = obj1+obj2; ------------------------------------------------------------------②

cout << obj.real <<endl;

cout << obj.imag << endl;

}

问题一：①处的运算符重载，为什么该函数的返回值要设计成CComplex类型？

答：

重载后，+实现的功能是复数相加．

问题二：②处的运算符重载函数调用就相当于“obj=operator+(obj1,obj2);”，但是为什么CComplex类中的运算符重载函数只设计了一个参数？

答：

因为运算符重载是作为类的成员函数定义的，它默认了一个参数，即this．

2．理解下面的程序，并在VC++6.0下运行查看结果，回答程序后面的问题。

#include <iostream>

using namespace std;

class CComplex

{

public:

CComplex()

{

real = 0.0;

imag = 0.0;

}

CComplex(float x, float y)

{

real = x;

imag = y;

}

CComplex operator + (CComplex &obj1, CComplex &obj2)

{

CComplex obj3(obj1.real + obj2.real, obj1.imag + obj2.imag);

return obj3;

}

CComplex &operator++(CComplex &obj)

{

obj.real += 1;

obj.imag +=1;

return obj;

}

void print()

{

cout<<real<<"+"<<imag<<"i"<<endl;

}

private:

float real;

float imag;

};

CComplex &operator--(CComplex &x)

{

x.real -= 1;

x.imag -= 1;

return x;

}

void main()

{

CComplex obj1(2.1,3.2);

CComplex obj2(3.6,2.5);

cout<<"obj1=";

obj1.print();

cout<<"obj2=";

obj2.print();

CComplex obj3 = obj1 + obj2;

cout<<"befor++, obj3=";

obj3.print();

++obj3;

cout<<"after++, obj3=";

obj3.print();

--obj3;

cout<<"after--, obj3=";

obj3.print();

CComplex obj4 = ++obj3;

cout<<"obj4=";

obj4.print();

}

问题一：以上程序中的三个运算符重载都有错误，试改正过来，并分析该程序的输出结果。

答：

更改后，该类为：

#include <iostream>

using namespace std;

class CComplex

{

public:

CComplex()

{

real = 0.0;

imag = 0.0;

}

CComplex(float x, float y)

{

real = x;

imag = y;

}

CComplex operator + (CComplex &obj2)

{

CComplex obj3(real + obj2.real, imag + obj2.imag);

return obj3;

}

CComplex &operator++()

{

real += 1;

imag += 1;

CComplex obj(real, imag);

return obj;

}

void print()

{

cout << real << "+" << imag << "i" << endl;

}

friend CComplex &operator--(CComplex &x);

private:

float real;

float imag;

};

CComplex &operator--(CComplex &x)

{

x.real -= 1;

x.imag -= 1;

return x;

}

输出结果为：

obj1=2.1+3.2i

obj2=3.6+2.5i

befor++, obj3=5.7+5.7i

after++, obj3=6.7+6.7i

after--, obj3=5.7+5.7i

obj4=6.7+6.7i

### 5.2.2 程序设计

1. 在以上复数类的基础上，增加重载运算符乘“\*”、除“/”，实现复数的乘除运算。

**两个重载函数如下，均设置为类的友元函数**

**在类中的定义为：**

friend CComplex &operator \*(CComplex &x, CComplex &y);

friend CComplex &operator /(CComplex &x, CComplex &y);

类

CComplex &operator \*(CComplex &x, CComplex &y)

{

CComplex y1;

y1.real = x.real\*y.real - x.imag\*y.imag;

y1.imag = x.imag\*y.real + x.real\*y.imag;

return y1;

}

CComplex &operator /(CComplex &y, CComplex &x)

{

CComplex x1,y1,z;

x1.real = x.real; x1.imag = -x.imag; // x的共轭

x1 = x1 \* x; //利用重载的乘号计算分母,计算后分母虚部为0，即为一个实数

y1 = y \* x1; //计算分子

z.real = y1.real / x1.real;

z.imag = y1.imag / x1.real;

return z;

}

2. 有两个矩阵a 和b ，均为2 行3 列。求两个矩阵之和。重载运算符" + " ，使之能用于矩阵相加. 如c=a+b， 本题是< C++面向对象程序设计》第4 章第4 题。

#include<iostream>

#include<iomanip>

using namespace std;

class Complex

{

private:

int i,j,n,a[2][3];

public:

Complex();

Complex operator+(Complex &c);

void display();

void input();

} t1;

Complex::Complex()

{

for(int i=0;i<2;i++)

for(int j=0;j<3;j++)

a[i][j]=0;

}

void Complex::input()

{

for(i=0;i<2;i++)

for(j=0;j<3;j++)

cin>>a[i][j];

}

void Complex::display()

{

n=1;

for(i=0;i<2;i++)

for(j=0;j<3;j++)

{

n++;

if(n%2==0)

cout<<endl;

cout<<setw(5)<<a[i][j];

}

cout<<endl;

}

Complex Complex::operator+(Complex &c)

{

for(int i=0;i<2;i++)

for(int j=0;j<3;j++)

t1.a[i][j]=a[i][j]+c.a[i][j];

return t1;

}

int main()

{

Complex t2,t3,t4;

cout<<"请输入2\*3个整数"<<endl;

t2.input();

cout<<endl;

cout<<"请输入2\*3个整数"<<endl;

t3.input();

cout<<endl;

cout<<"t2";

t2.display();

cout<<"t3";

t3.display();

t4=t2+t3;

cout<<"t4=t2+t3=";

t4.display();

return 0;

}

## 5.3思考题

１．定义CPoint类，有两个成员变量：横坐标（x）和纵坐标（y），对CPoint类重载“++”（自增运算符）、“--”（自减运算符），实现对坐标值的改变。

#include <iostream>

using namespace std;

class Point

{

private:

int x, y;

public:

Point() { x = 0; y = 0; }

Point(int x1,int x2) :x(x1),y(x2){}

void Show() { cout << "x = " << x << " y = " << y << endl; }

friend Point &operator++(Point &p);

friend Point &operator--(Point &p);

};

Point &operator++(Point &p)

{

Point p1;

p.x = p.x + 1;

p.y = p.y + 1;

return p;

}

Point &operator--(Point &p)

{

p.x = p.x - 1;

p.y = p.y - 1;

return p;

}

int main()

{

Point p(5, 6),p1;

++p;

p.Show();

return 0;

}