

电子科技大学信息与软件工程学院

标准实验报告

(实验) 课程名称: 面向对象程序设计 C++

电子科技大学教务处制表

学生姓名：罗悦 学 号：2016220304022 指导教师：钱伟中

实验地点：信软学院实验室

实验时间：2018-10-17

一、实验室名称：信软学院软件实验室

二、实验项目名称： C++的函数及运算符重载程序设计

三、实验学时：2 学时

四、实验原理

本次实验需要设计复数类 MyComplex，通过运算符的重载，使复数运算具备+，-，*，/，<<，>>等功能。

五、实验目的：

通过实验练习，让学生理解类的成员函数重载的含义，理解构造函数的重载的意义，并掌握如何对构造函数重载；掌握函数重载的两种方式，分别通过成员函数和友元函数实现重载，理解两种实现方式的区别，最后，理解和掌握几种常用运算符重载的实现方法。

六、实验内容：

设计复数类 MyComplex，复数的实部和虚部均为 double 类型，并且放在 private 段；重载复数类的构造函数，及复制构造函数；重载复数运算+，-，*，/；重载复数流运算符<<,>>。同时实现返回复数实部、虚部、模的成员函数。编程设计实现下列函数：

```
myComplex();  
myComplex(int a);  
myComplex(int a,int b)  
myComplex( myComplex& v)  
double getReal(); //返回复数的实部  
double getImaginary (); //返回复数的虚部  
double getModulus (); //返回复数的模  
myComplex& operator=( myComplex& rhs); //类对象的赋值  
myComplex& operator+=( myComplex& rhs);
```

```
myComplex& operator-=( myComplex& rhs);  
myComplex& operator*=( myComplex& rhs);  
myComplex& operator/=( myComplex& rhs);  
friend myComplex operator+(myComplex m,myComplex n)  
friend myComplex operator-(myComplex m,myComplex n)  
friend myComplex operator*(myComplex m,myComplex n)  
friend myComplex operator/(myComplex m,myComplex n)  
friend ostream& operator<<(ostream& os,myComplex c);  
friend istream& operator>>(istream& is,myComplex& c);
```

七、实验器材（设备、元器件）：

PC 计算机、Windows 系列操作系统 、Visual Studio2013 软件

八、实验步骤：

- 1) 创建工程；
- 2) 创建头文件；
- 3) 定义复数类；
- 4) 实现成员函数；
- 5) 实现运算符重载函数；
- 6) 编写主程序，验证重载后的运算符功能。
- 7) 编译链接，并运行程序。

九、实验程序及结果分析：

myComplex.h 头文件

```
#include <iostream>  
  
#include <cmath>  
  
#include "math.h"  
  
#include "stdlib.h"  
  
using namespace std;  
  
class myComplex
```

```
{
private:
    double Real;          //定义实部
    double Imaginary;     //定义虚部
    double Modulus;       //定义模
public:
    myComplex(){           //构造函数
        Real=5;
        Imaginary=6;
    }
    myComplex(double a){   //重载构造函数
        Real=a;
        Imaginary=6;
    }
    myComplex(double a,double b){ //重载构造函数
        Real=a;
        Imaginary=b;
    }
    myComplex(const myComplex& v){ //复制构造函数
        Real=v.Real;
        Imaginary=v.Imaginary;
        Modulus=v.Modulus;
    }
    double &getReal(){      //返回实部
        return Real;
    }
    double &getImaginary(){ //返回虚部
        return Imaginary;
    }
}
```

```

double &getModulus(){           //返回模
    Modulus=sqrt(Real*Real+Imaginary*Imaginary);
    return Modulus;
}

myComplex& operator=(const myComplex& r){//类对象的赋值
    Real=r.Real;
    Imaginary=r.Imaginary;
    return *this;
}

myComplex& operator+=(const myComplex& r){//重载运算符"+="
    Real+=r.Real;
    Imaginary+=r.Imaginary;
    return *this;
}

myComplex& operator-=(const myComplex& r){//重载运算符 "-="
    Real-=r.Real;
    Imaginary-=r.Imaginary;
    return *this;
}

myComplex& operator*=(const myComplex& r){//重载运算符 "*="
    double k;                //定义一个记录原始 Real 值的量
    k=Real;
    Real=Real*r.Real-Imaginary*r.Imaginary;
    Imaginary=k*r.Imaginary+r.Real*Imaginary;
    return *this;
}

myComplex& operator/=(const myComplex& r){//重载运算符 "/="
    double k;                //定义一个记录原始 Real 值的量
    k=Real;

```

```

Real=(Real*r.Real+Imaginary*r.Imaginary)/(r.Real*r.Real+r.Imaginary*r.Imaginary);

Imaginary=(r.Real*Imaginary-k*r.Imaginary)/(r.Real*r.Real+r.Imaginary*r.Imaginary);

    return *this;

}

friend myComplex operator+(myComplex &m,myComplex &n);
friend myComplex operator-(myComplex &m,myComplex &n);
friend myComplex operator*(myComplex &m,myComplex &n);
friend myComplex operator/(myComplex &m,myComplex &n);
friend ostream& operator<<(ostream& os,myComplex& c);
friend istream& operator>>(istream& is,myComplex& c);

};

myComplex operator+(myComplex &m,myComplex &n){//重载运算符"+"
    myComplex temp(m.getReal()+n.getReal(),m.getImaginary()+n.getImaginary());
    return temp;
}

myComplex operator-(myComplex &m,myComplex &n){//重载运算符 "-"
    myComplex temp(m.getReal()-n.getReal(),m.getImaginary()-n.getImaginary());
    return temp;
}

myComplex operator*(myComplex &m,myComplex &n){//重载运算符 "*"
    myComplex
temp(m.getReal()*n.getReal()-m.getImaginary()*n.getImaginary(),m.getReal()*n.getI
maginary()+m.getImaginary()*n.getReal());
    return temp;
}

myComplex operator/(myComplex &m,myComplex &n){//重载运算符 "/"
    myComplex

```

```

temp((m.getReal()*n.getReal()+m.getImaginary()*n.getImaginary())/(n.getReal()*n.getReal()+n.getImaginary()*n.getImaginary()),(n.getReal()*m.getImaginary()-m.getReal()*n.getImaginary())/(n.getReal()*n.getReal()+n.getImaginary()*n.getImaginary()));
    return temp;
}
ostream& operator<<(ostream& os, myComplex& c){//重载运算符"<<"
    os <<'('<<c.getReal()<<','<<c.getImaginary()<<','<<c.getModulus()<<')';
    return os;
}
istream& operator>>(istream& is, myComplex& c){//重载运算符">>"
    is >>c.getReal()>>c.getImaginary()>>c.getModulus();
    return is;
}

```

如上程序所示，我定义了一个 myComplex.h 头文件，这个头文件主要包括了一个复数类 myComplex，这个类定义了复数的实部、虚部和模的值为私有变量，并且都为 double 类型。在这个类中，定义了一个无参数的构造函数，并且进行了重载，又定义了两个分别有一个参数和两个参数的构造函数。且定义了成员函数来访问私有变量，返回私有变量的值。同时重载了赋值运算符“=”，和单目运算符“+”“-”“*”“/”。并且也运用友元的方式，在类外定义了双目运算符的重载函数，分别有“+”“-”“*”“/”，同时定义了运算符“<<”与“>>”。

myComplex.cpp 主程序

```

#include <iostream>
#include <cmath>
#include "mycomplex.h"
using namespace std;

int main(){
    myComplex a1(9,10);    //定义一个对象 a1
    myComplex a2(7,8);    //定义一个对象 a2

```

cout<<"定义了两个复数类对象 a1、a2 如下所示，括号中从左到右分别为实部、虚部、模： "<<endl;

cout<<a1<<endl;

cout<<a2<<"\n"<<endl;

myComplex a3; //定义一个对象 a3

cout<<"定义了一个复数类对象 a3： "<<endl;

a3=a1+a2; //检测运算符"+"

cout<<"运算 a3=a1+a2 后，a3 的实部、虚部和模分别为： "<<endl;

cout<<a3<<"\n"<<endl;

a3=a1-a2; //检测运算符 "-"

cout<<"运算 a3=a1-a2 后，a3 的实部、虚部和模分别为： "<<endl;

cout<<a3<<"\n"<<endl;

a3=a1*a2; //检测运算符 "*"

cout<<"运算 a3=a1*a2 后，a3 的实部、虚部和模分别为： "<<endl;

cout<<a3<<"\n"<<endl;

a3=a1/a2; //检测运算符 "/"

cout<<"运算 a3=a1/a2 后，a3 的实部、虚部和模分别为： "<<endl;

cout<<a3<<"\n"<<endl;

a1+=a2; //检测运算符 "+="

cout<<"运算 a1+=a2 后，a1 的实部、虚部和模分别为： "<<endl;

cout<<a1<<"\n"<<endl;

a1-=a2; //检测运算符 "-="


```

cout<<"运算 a1-=a2 后，a1 的实部、虚部和模分别为："<<endl;
cout<<a1<<"\n"<<endl;

a1*=a2;                //检测运算符"*="
cout<<"运算 a1*=a2 后，a1 的实部、虚部和模分别为："<<endl;
cout<<a1<<"\n"<<endl;

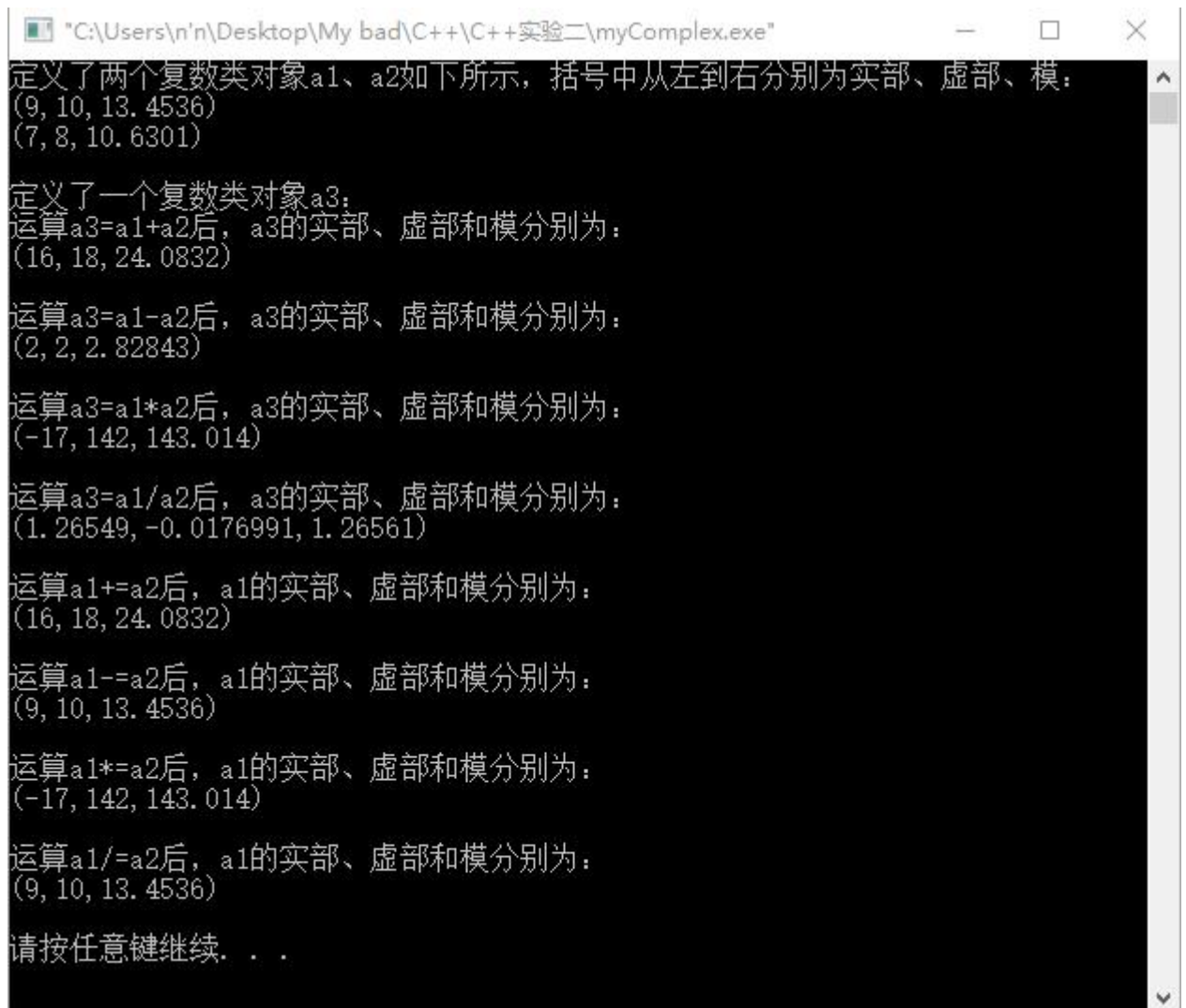
a1/=a2;                //检测运算符"/="
cout<<"运算 a1/=a2 后，a1 的实部、虚部和模分别为："<<endl;
cout<<a1<<"\n"<<endl;

return 0;

}

```

如上主程序所示，我定义了一个对象 a1，实部为 9，虚部 10，一个对象 a2，实部为 7，虚部 8，并分别打印了出来。之后又定义了一个对象 a3 用来进行双目运算符的重载测试工作，在之后分别进行了“a3=a1+a2”“a3=a1-a2”“a3=a1*a2”“a3=a1/a2”四个操作并且将计算后的 a3 的实部虚部和模的值打印出来。然后又进行了单目运算符的重载测试工作，分别进行了“a1+=a2”“a1-=a2”“a1*=a2”“a1/=a2”四个操作，将计算后的 a1 的实部虚部和模的值打印出来。实验结果如下图所示：



```
"C:\Users\n'n\Desktop\My bad\C++\C++实验二\myComplex.exe"
定义了两个复数类对象a1、a2如下所示，括号中从左到右分别为实部、虚部、模：
(9, 10, 13.4536)
(7, 8, 10.6301)

定义了一个复数类对象a3：
运算a3=a1+a2后，a3的实部、虚部和模分别为：
(16, 18, 24.0832)

运算a3=a1-a2后，a3的实部、虚部和模分别为：
(2, 2, 2.82843)

运算a3=a1*a2后，a3的实部、虚部和模分别为：
(-17, 142, 143.014)

运算a3=a1/a2后，a3的实部、虚部和模分别为：
(1.26549, -0.0176991, 1.26561)

运算a1+=a2后，a1的实部、虚部和模分别为：
(16, 18, 24.0832)

运算a1-=a2后，a1的实部、虚部和模分别为：
(9, 10, 13.4536)

运算a1*=a2后，a1的实部、虚部和模分别为：
(-17, 142, 143.014)

运算a1/=a2后，a1的实部、虚部和模分别为：
(9, 10, 13.4536)

请按任意键继续. . .
```

实验步骤和打印结果如上图所示。

十、实验结论：

本次实验了解到了单目运算符的重载函数可以在类中定义并且书写实现，而双目运算符的重载函数需要用友元的方式在类外书写，因为如果在类中书写，运算符重载为成员函数，第一个参数必须是本类的对象，因为默认使用第一个参数的对象进行调用该成员函数。一般的运算符的重载函数都能书写，但是如.，*，::，?:，这几个运算符不能被重载，且运算符函数的参数至少有一个必须是类的对象或者类的对象的引用。

十一、总结及心得体会：

本次实验让我学习到了很多知识，对C++的函数和语言又得到了更深的理解。我了解到了重载不能改变运算符的优先级。重载也不能改变运算符的结合律。

重载也不能改变运算符操作数的个数。比如+需要两个操作数，则重载的+也必须要有两个操作数。且在实验中复数的乘法和除法并不是简单的实部相乘(相除)和虚部相乘(相除)，而是需要做一个简单的运算才能得到正确的结果。

十二、对本实验过程及方法、手段的改进建议：

我认为该实验还可以添加“+”“-”“++”“--”等运算符来丰富实验内容，且这几个运算符的操作也比较特殊和重要。

报告评分：

指导教师签字：