myComplex Class II

Assignment

Define a class for complex numbers. In addition to the member functions that sets and gets the member attributes, the class should have following member functions:

1. Constructors
   a. Default constructor
   b. Another constructor uses default parameters.
   c. Copy constructor
2. Mutators: setReal(), setImaginary()
3. Assessors: getReal(), getImaginary()
4. Overload output operators (<<) for `myComplex` class.
5. Overload input operator (>>) for `myComplex` class.
6. Overload assignment operator (=) for `myComplex` class
7. A member function to calculate the absolute value of the complex number
8. Overload arithmetic operators +, –, *, and /.
9. Overload comparison operators == and !=.

Write a driver program using complex class. Every member functions and operators should be called or activated to verify that they are working properly.

** Use UML to design the class (see Ch 13.15)

Objectives

The objective of this lab is to overload operators.

- Define and implement a complex class
- Define and implement constructors of the complex class
- Overload arithmetic operators: +, - , *, /
- Overload input/output operators: >>, <<
- Overload comparison operators: == and !=

An Example of Program Output

```
+--------------------------------------------+
|  This program tests complex class operations |
+--------------------------------------------+
0. exit
1. addition
2. subtraction
3. multiplication
4. multiply by a factor
5. division
 Selection: 1

Enter a complex number(separate by a space): 1 2
Enter another complex number(separate by a space): 3 4
 (1 + 2i)  +  (3 + 4i)  =  (4 + 6i)

0. exit
1. addition
```

```
2. subtraction
3. multiplication
4. multiply by a factor
5. division
 Selection: 3
Enter a complex number(separate by a space): 1 2
Enter another complex number(separate by a space): 3 4
 (1 + 2i)  *  (3 + 4i)  =  (-5 + 10i)
```

Design / Define a class for complex numbers

To design a complex class, we will list the information (i.e. data) of a complex number and actions that a complex number can do.

A complex number:
A complex number has the form of a + bi, where a and b are real numbers and i is the square root of -1. We refer to a as the real part and b as the imaginary part of the complex number. For example: 2.5 + 3 i is a complex number. 2.5 is the real part and 3 is the imaginary part of 2.5 + 3 i.

Member attributes:
The class should have two member attributes to represent the real and imaginary parts of a complex number.

Additional member function absolute_value( )

We can define a member function that calculates the absolute value of a complex number. The absolute value of a complex number, a + b *i*, is defined as:

$$|a+bi| = \sqrt{a^2+b^2}$$

On a complex number coordinate system, this value is the distance from origin to the complex point.

Operator Overload

We can add, subtract, multiply and divide complex numbers. The operation is defined as:

$$(a+bi)+(c+di) = (a+c)+(b+d)i$$

$$(a+bi)-(c+di) = (a-c)+(b-d)i$$

$$(a+bi)(c+di) = (ac-bd)+(ad+bc)i$$

$$\frac{(a+bi)}{(c+di)} = \frac{(ac+bd)}{c^2+d^2} + \frac{(bc-ad)}{c^2+d^2}i$$

myComplex Class II

Overload arithmetic operators

Operators are overloaded as non-member functions. Arithmetic operators +, -, *, / are all binary operators. This means they operate on two complex numbers and return the result as a third complex numbers. The function prototype for addition is:

```
myComplex operator + (const myComplex &c1, const myComplex &c2);
```

It takes two arguments and returns an object of myComplex class.

Overload input/output operators

Input operator (>>) is defined as a friend function of class myComplex. By doing so, the input operator can modify private member attributes. The function prototype is:

```
friend std::istream& operator >> (std::istream ins, myComplex &c);
```

The output operator (<<) is defined as a non-member function of class myComplex. The function prototype takes the similar form as that of the input operator.

Overload comparison operators: == and !=

Comparison operators return boolean values

Submissions

Submit following 3 files, including your test results:

1. UML of myComplex class design
2. myComplex.h – header file (with operator overload)
3. myComplex.cpp – implementation file (with operator overload
4. Your driver program (lab04.cpp file)
5. your test result: result.txt