Experiment No.3

1. **Title:** Demonstrate use of operator overloading for Complex class.

# Problem Statement:

Implement a class Complex which represents the Complex Number data type. Implement the following operations:

* 1. Constructor (including default constructor which creates the complex number 0+0i).
  2. Overloaded << and >> to print and read Complex Numbers.
  3. Overloaded operator+ to add two complex numbers.
  4. Overloaded operator\* to multiply two complex numbers.

# Objectives:

1. To understand concept of operator overloading.
2. To demonstrate overloading of binary operator, insertion and extraction operator.

# Outcomes:

* Students will be able to demonstrate use of constructor
* Students will be able to demonstrate binary operator overloading
* Students will be able to demonstrate overloading of insertion and extraction operator using friend function.

# Software/Hardware/Other Requirements:

* Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
* Operating System – ubuntu/Fedora 64bit OS
* Software: G++ compiler/ GCC compiler, Code Editor

# Theory:

1. **Concept of Overloading:**

* C++ allows you to specify more than one definition for a function name or an operator in the same scope, which is called function overloading and operator overloading respectively.
* An overloaded declaration is a declaration that had been declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).
* When you call an overloaded function or operator, the compiler determines the most appropriate definition to use by comparing the argument types you used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called overload resolution.

# Operator Overloading:

* Operator overloading is one of the special features of C++. It also shows the extensibility of C++.
* C++ permits us to add two variables of user defined types with the same way that is applied with built in type data type.
* This refers to ability to provide special meaning for existing data type. This mechanism of giving such special meaning to an operator is known as Operator overloading.
* Operator overloading provides a flexible option for creation of new definition for most of the C++ operators. We can assign additional meaning to all existing C++ operators except following:
  1. Scope resolution operator ( : : )
  2. Size of operator (sizeof)
  3. Conditional operator (?:)
  4. Class member access operators ( . , .\*)

# Rules for Operator Overloading:

1. Existing operators can only be overloaded, but the new operators cannot be overloaded.
2. Overloaded operator must have at least one operand that is user-defined type.
3. We cannot use friend function to overload certain operators. However, the member function can be used to overload those operators.
4. We must remember that the precedence of operators cannot be changed by operator overloading. g. Multiplication operator has higher precedence than addition operator. We cannot change this meaning.
5. Binary arithmetic operators such as +, -, \*, and / must return a value.
6. Chart for Parameter passing count:

|  |  |  |
| --- | --- | --- |
| **Type of Function** | **Unary Operator**  **Overloading** | **Binary Operator**  **Overloading** |
| Member Function | 0 | 1 |
| Friend Function | 1 | 2 |

1. Operators that cannot be overloaded

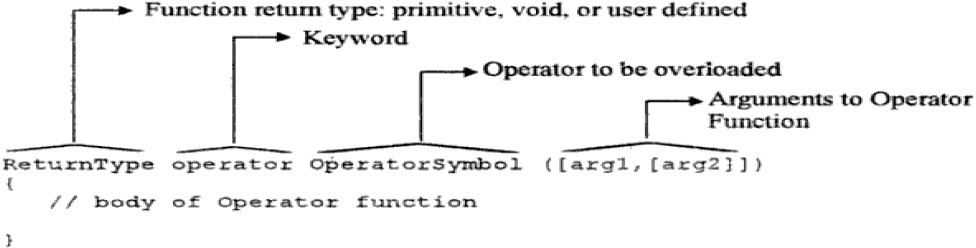
|  |  |
| --- | --- |
| Operator | Meaning |
| **sizeof** | Size operator |
| **.** | Membership operator |
| **.\*** | Pointer to member operator |
| **::** | Scope resolution operator |
| **?:** | Conditional operator |

1. Operators that cannot be overloaded with friend function

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| **=** | Assignment operator |
| **()** | Function call |
| **[]** | Subscripting operator |
| **->** | Class member access operator |

# How to Define Operator Overloading Function:

* To define an additional task to an operator, we must specify what it means in relation to class to which the operator is applied. This is done with the help of a special function, called operator function which describes the task.



* Operator function is no static member function or it may be friend function. A basic difference between them is that friend function will have only one argument for unary and binary operator whereas member function has no argument for unary operators and only one for binary operators. This is because the object used to invoke the member function is passé implicitly and therefore is available for member function. This is not the case with friend functions. Arguments may be passed either by value or by reference.

# How to Operator function can be called in C++ programming?

1. **Using Traditional Method :** This method is similar to call function which we are using for the other normal function.
   1. For unary operation we can call the operator function by::

# object-name . operator op(operand); // where op=operator

* 1. For binary operation we can call the operator function by:

# object-name . operator op(operand1, operand2);

1. **Using Expression Syntax**: This method is just like the normal expression we use for various arithmetic operations such as addition, subtraction, etc. (e.g. a + b)
   1. For unary operation we can call the operator function.

# op operand // here op = operator or

**operand op**

* 1. For binary operation we can call the operator function by:

# operand1 op operator2

1. **Unary operators overloading in C++:**

* The unary operators operate on a single operand and following are the examples of Unary operators:

1. The increment (++) and decrement (–) operators.
2. The unary minus (-) operator.
3. The logical not (!) operator.

* The unary operators operate on the object for which they were called and normally, this operator appears on the left side of the object, as in !obj, -obj, and ++obj but sometime they can be used as postfix as well like obj++ or obj–.

# Binary operator overloading:

* In overloading binary operator, a friend function will have two arguments, while a member function will have one argument.
* Overloading binary operators using friends: Friend functions may be used in the place of member functions for overloading a binary operator.
* The only difference being that a friend function requires two arguments to be explicitly passed to it while a member function requires only one.
* The same complex number program with friend function can be developed as friend complex operator +(complex, complex); and we will define this function as

complex operator + (complex a, complex b) { return complex ( c.x + b.x), (a.y + b.y) ;

}

in this case, the statement

c3 = c1 + c2; is equal to c3 = operator + (c1, c2)

# Overloading stream insertion (<<) & extraction (>>) operators:

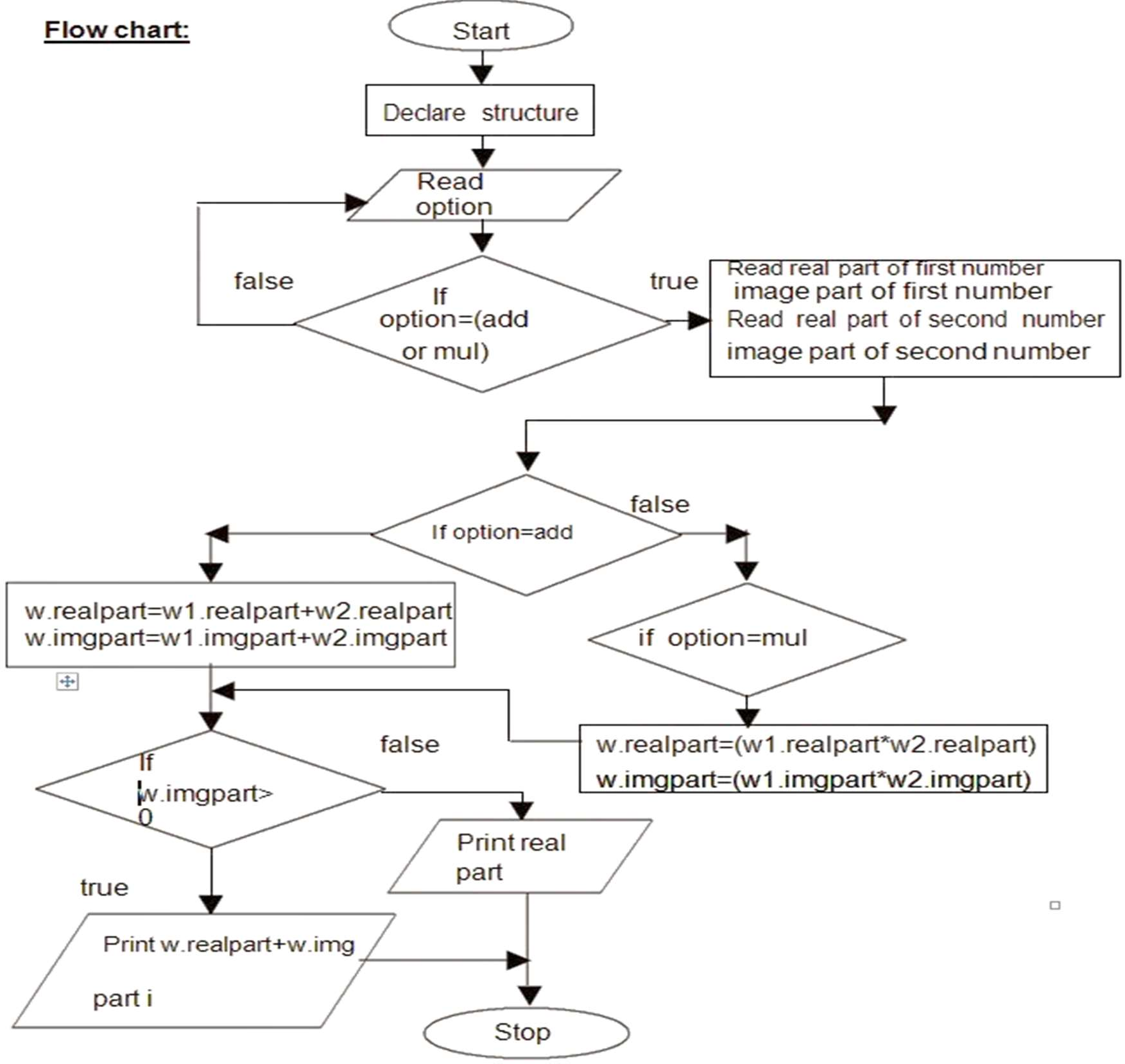
* In C++, stream insertion operator “<<” is used for output and extraction operator “>>” is used for input.
* We must know following things before we start overloading these operators.
  + cout is an object of ostream class and cin is an object istream class
  + These operators must be overloaded as a global function. And if we want to allow them to access private data members of class, we must make them friend.

# Why these operators must be overloaded as global?

* + In operator overloading, if an operator is overloaded as member, then it must be a member of the object on left side of the operator.
  + The operators ‘<<' and '>>' are called like 'cout << ob1' and 'cin >> ob1’.
  + So, if we want to make them a member method, then they must be made members of ostream & istream classes, which is not good option most of the time.
  + Therefore, these operators are overloaded as global functions with two parameters, cout and object of user defined class.

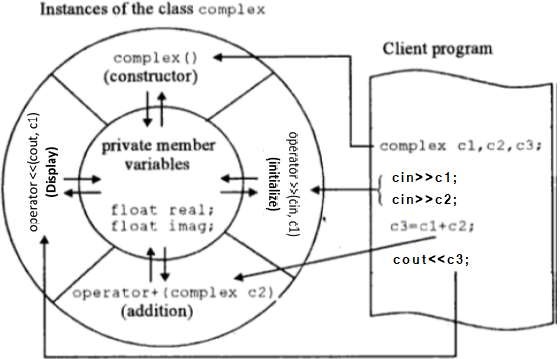
# Flowchart/Data Flow Diagram

**Fig: Operator Overloading in Complex Class**

****

**Fig: Flowchart Diagram for Complex Number using Operator Overloading Concept**

# Implementation:

****

#include<iostream> using namespace std; class Complex

{

float real,img; public:

Complex()

{

real=0.0f; img=0.0f;

}

Complex(float a,float b)

{

real=a; img=b;

}

//Addition of two complex numbers Complex &operator+(Complex c1)

{

Complex temp; temp.real=real+c1.real; temp.img=img+c1.img; return temp;

}

//Subtraction of two complex numbers Complex operator-(Complex c1)

{

Complex temp; temp.real=real-c1.real; temp.img=img-c1.img; return temp;

}

**C++ Program for Complex Number using Operator Overloading**

//Multiplication of two complex numbers Complex operator\*(Complex c1)

{

Complex temp; temp.real=(real\*c1.real)-(img\*c1.img); temp.img=(img\*c1.real)+(real\*c1.img); return temp;

}

//Division of two complex numbers Complex operator/(Complex c1)

{

Complex temp,c2; c2.img=-c1.img; float x;

temp.real=(real\*c1.real)-(img\*(c2.img)); temp.img=(real\*c1.real)+(real\*(c2.img)); x=(c1.real)\*(c1.real)+(c1.img)\*(c1.img); temp.real=temp.real/x; temp.img=temp.img/x;

return temp;

}

//overloaded insertion (<<) opertor for class Complex friend ostream &operator<<(ostream &out, Complex &c)

{

out << c.real<<" + "<< c.img<<" i";; return out;

}

//overloaded extraction (>>) opertor for class Complex friend istream &operator>>(istream &in, Complex &c)

{

}

};

int main()

{

in>> c.real>>c.img; return in;

Complex c1,c2,c3; int choice;

char ans; do

{

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*\*\n"; cout<<"\n\t1.Addition\n\t2.Subtraction\n\t3.Multiplication\n\t4.Division"; cout<<"\n\nEnter your choice: ";

cin>>choice;

cout<< "Enter real and img part of first complex number\n"; cin>>c1;

cout<< "Enter real and img part of second complex number\n"; cin>>c2;

switch(choice)

{

case 1: c3=c1+c2;

cout<<"\n\nAddition is: "; cout<<c3;

|  |  |
| --- | --- |
| } | break; case 2: c3=c1-c2;  cout<<"\n\nSubtraction is: "; cout<<c3;  break; case 3: c3=c1\*c2;  cout<<"\n\nMultiplication is: "; cout<<c3;  break; case 4: c3=c1/c2;  cout<<"\n\nDivision is: "; cout<<c3;  break; default:cout<<"\nWrong choice";  }  cout<<"\nDo you want to continue?(y/n): "; cin>>ans;  }while(ans=='y' || ans=='Y'); return 0; |
| **Output:** | |
| \*\*\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*\* | |
| 1.Addition | |
| 2.Subtraction | |
| 3.Multiplication | |
| 4.Division | |
| Enter your choice: 1 | |
| Enter real and img part of first complex number | |
| 4.2 2.1 | |
| Enter real and img part of second complex number | |
| 2.1 4.2 | |
| Addition is: 6.3 + 6.3 i | |
| Do you want to continue? (y/n): y | |
| \*\*\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*\* | |
| 1.Addition | |
| 2.Subtraction | |
| 3.Multiplication | |
| 4.Division | |
| Enter your choice: 3 | |
| Enter real and img part of first complex number | |
| 4.1 2.2 | |
| Enter real and img part of second complex number | |
| 1.5 3.2 | |
| Multiplication is: -0.890001 + 16.42 i | |
| Do you want to continue? (y/n): n | |

# Conclusion:

Hence, we have studied, used and demonstrated use of binary operator overloading and insertion-extraction operator overloading using friend function.

# Review Questions & Exercises:

1. **Fill in the Blanks**
   1. An operator may be overloaded to work with a specific class.
   2. When overloading the postfix increment/postfix decrement operator, its function must have a dummy parameter.
   3. Overloading an operator allows operators to be used with a user defined types(classes)
   4. The dot(.) operator can be used with a new class without defining this binary function.
   5. The array subscript operator [], when overloaded, cannot \_ take multiple values inside (eg [4, 5,6]).

# Answer the following:

* 1. What is the significance of operator overloading?

Operator overloading is an important concept in c++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type.

E.g. we can overload an + operator for complex class(created by us) to perform addition operation on complex numbers.

* 1. List down the operators that cannot be overloaded as a friend.

Following operators that cannot be overloaded as a friend :-

1) Assignment operator =

2) Function call operator ()

3) Subscript operator []

4) class member access operator ->

* 1. Write any one step for the process of overloading.

Overloaded operators cannot have default arguments except the function call operator () which can have default arguments.

* 1. What is type conversion?

C++ allows to convert data of one type to that of another.

Eg :- conversion from int to long long,float to double,etc.

* 1. State three different type conversion

a) Implicit Type Conversion

b) Explicit Type Conversion

c) Converting by assignment

# Algorithm Assignments:

* 1. Assume a class named Bird exists. Write the member function definition that overloads the = operator for that class.

Bird Bird::operator = (Bird const &b1);

* 1. The class Stuff has both a copy constructor and an overloaded = operator. Assume that blob and clump are both instances of the Stuff class. For each statement below, indicate whether the copy constructor or the overloaded = operator will be called.
     1. Stuff blob = clump; (copy constructor )
     2. clump = blob; (overloaded =)
     3. blob.operator=(clump); (overloaded =)
     4. showValues(blob); // blob is passed by value. (copy constructor)

# Programming Assignments:

* 1. Overload new & delete operators to manipulate objects of the Student class. The Student class must contain data members such as char \*name, int roll\_no, char

\*branch, etc. the overloaded new & delete operators must allocate memory for the student class object & its data members.

* 1. Write an interactive operator overloaded program for manipulating rational numbers. Overload operators such as +, -, \*, /, >>, <<, >, <, ==.
  2. Write an interactive operator overloaded program to manipulate the Quadratic Equation. Overload operators such as >>, <<, +, -. Also write functions for:
     1. To computes the value of a polynomial for a given value of x.
     2. To computes the two solutions of the equation ax2+bx+c=0

# References:

1. E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-

Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990

1. K. R. Venugopal, B. Rajkumar, and T. RaviShankar Mastering C++ by K. R. Venugopal; with 2nd edition. McGraw-Hill Publication, ISBN13: 9781259029943.