ASSIGNMENT 2

# Aim:

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

1. Constructor (including a default constructor which creates the complex number (0+0i).

2. Overloaded operator ‘+’ to add two complex numbers.

3. Overloaded operator ‘\*’ to multiply two complex numbers.

4. Overloaded <<and >> to print and read Complex Numbers.

# OBJECTIVES:

To create a user defined data type as Complex by implementing class.Also perform operator overloading on arithmetic operators ‘+’, ‘-‘ on complex numbers.

# THEORY:

A **class in C++** is the building block , that leads to **Object**-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that **class**. A **C++ class** is like a blueprint for an **object**. Here we are creating user defined data type as complex and also performing arithmetic operations on complex data type by using operator overloading.

**Operator Overloading:**

In C++, we can make operators to work for user defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading.

**Syntax:**

Classname operator ‘operator\_to\_be\_overloaded’ (object of class);

Algorithm:

# SOURCECODE:

#include<iostream>

using namespace std;

class complex

{

private:

int real,imag;

public:

complex()

{

real = 0;

imag = 0;

}

void input()

{

cout<<"Enter real part of complex number : ";

cin>>real;

cout<<"Enter the imaginary part of complex number : ";

cin>>imag;

}

complex operator + (complex c2)

{

complex temp;

temp.real = real + c2.real;

temp.imag = imag + c2.imag;

return temp;

}

complex operator \* (complex c2)

{

complex temp;

temp.real = real \* c2.real - imag \* c2.imag;

temp.imag = real \* c2.imag + imag \* c2.real;

return temp;

}

void display()

{

if(imag<0)

{

cout<<"\nThe result complex number is "<<real<<imag<<"i\n";

}

else

{

cout<<"\nThe result complex number is "<<real<<" + "<<imag<<"i\n";

}

}

};

int main()

{

complex c1,c2,result;

int choice;

p:cout<<"\nEnter the first complex number\n";

c1.input();

cout<<"\nEnter the second complex number\n";

c2.input();

cout<<"\nFor Addition ~press 1 \nFor Multiplication ~press 2\n";

cin>>choice;

switch(choice)

{

case 1:

result = c1 + c2;

result.display();

break;

case 2:

result = c1 \* c2;

result.display();

break;

default:

cout<<"\nEnter valid choice\n"

;goto p;

}

char ch;

cout<<"Do you want to continue (~press Y/N)\n";

cin>>ch;

if(ch=='y'||ch=='Y')

{

goto p;

}

return 0;

}

# OUTPUT:

/\*

Enter the first complex number

Enter real part of complex number : 2

Enter the imaginary part of complex number : 3

Enter the second complex number

Enter real part of complex number : 4

Enter the imaginary part of complex number : 2

For Addition ~press 1

For Multiplication ~press 2

1

The result complex number is 6 + 5i

Do you want to continue (~press Y/N)

n

--------------------------------

Process exited after 6.38 seconds with return value 0

Press any key to continue . . .

\*/

# CONCLUSION:

By using class , we can create a user defined data type as we want. Operator overloading helps to do arithmetic operations on such user defined data types.