Operator overloading in C++

## Simple Addition in C++ Binary Operator Overloading Program

// Header Files

#include<iostream>

#include<conio.h>

//Standard namespace declaration

using namespace std;

class overloading {

int value;

public:

void setValue(int temp) {

value = temp;

}

overloading operator+(overloading ob) {

overloading t;

t.value = value + ob.value;

return (t);

}

void display() {

cout << value << endl;

}

};

//Main Functions

int main() {

overloading obj1, obj2, result;

int a, b;

cout << "Enter the value of Complex Numbers a,b:";

cin >> a>>b;

obj1.setValue(a);

obj2.setValue(b);

result = obj1 + obj2;

cout << "Input Values:\n";

obj1.display();

obj2.display();

cout << "Result:";

result.display();

getch();

return 0;

}

## Sample Output

Enter the value of Complex Numbers a,b:10

5

Input Values:

10

5

Result:15

## Definition

To write a program to find the complex numbers using unary operator overloading.

## Unary operators:

* Increment (++) Unary operator.
* Decrement (--) Unary operator.
* The minus (-) unary.
* The logical not (!) operator.

## Unary Operator Overloading Algorithm/Steps:

* Step 1: Start the program.
* Step 2: Declare the class.
* Step 3: Declare the variables and its member function.
* Step 4: Using the function getvalue() to get the two numbers.
* Step 5: Define the function operator ++ to increment the values
* Step 6: Define the function operator - -to decrement the values.
* Step 7: Define the display function.
* Step 8: Declare the class object.
* Step 9: Call the function getvalue()
* Step 10: Call the function operator ++() by incrementing the class object and call the function display.
* Step 11: Call the function operator - -() by decrementing the class object and call the function display.
* Step 12: Stop the program.

## Simple Program for Unary Operator Overloading Program

#include<iostream.h>

#include<conio.h>

class complex {

int a, b, c;

public:

complex() {

}

void getvalue() {

cout << "Enter the Two Numbers:";

cin >> a>>b;

}

void operator++() {

a = ++a;

b = ++b;

}

void operator--() {

a = --a;

b = --b;

}

void display() {

cout << a << "+\t" << b << "i" << endl;

}

};

void main() {

clrscr();

complex obj;

obj.getvalue();

obj++;

cout << "Increment Complex Number\n";

obj.display();

obj--;

cout << "Decrement Complex Number\n";

obj.display();

getch();

}

## Sample Output

Enter the two numbers: 3 6

Increment Complex Number

4 + 7i

Decrement Complex Number

3 + 6i

## Aim

To write a program to add two complex numbers using binary operator overloading.

## Binary Operator Overloading Algorithm/Steps:

* Step 1: Start the program.
* Step 2: Declare the class.
* Step 3: Declare the variables and its member function.
* Step 4: Using the function getvalue() to get the two numbers.
* Step 5: Define the function operator +() to add two complex numbers.
* Step 6: Define the function operator –()to subtract two complex numbers.
* Step 7: Define the display function.
* Step 8: Declare the class objects obj1,obj2 and result.
* Step 9: Call the function getvalue using obj1 and obj2
* Step 10: Calculate the value for the object result by calling the function operator + and     operator -.
* Step 11: Call the display function using obj1 and obj2 and result.
* Step 12: Return the values.
* Step 13: Stop the program.

## Binary Operator Overloading Example Program

#include<iostream.h>

#include<conio.h>

class complex {

int a, b;

public:

void getvalue() {

cout << "Enter the value of Complex Numbers a,b:";

cin >> a>>b;

}

complex operator+(complex ob) {

complex t;

t.a = a + ob.a;

t.b = b + ob.b;

return (t);

}

complex operator-(complex ob) {

complex t;

t.a = a - ob.a;

t.b = b - ob.b;

return (t);

}

void display() {

cout << a << "+" << b << "i" << "\n";

}

};

void main() {

clrscr();

complex obj1, obj2, result, result1;

obj1.getvalue();

obj2.getvalue();

result = obj1 + obj2;

result1 = obj1 - obj2;

cout << "Input Values:\n";

obj1.display();

obj2.display();

cout << "Result:";

result.display();

result1.display();

getch();

}

## Sample Output:

Enter the value of Complex Numbers a, b

4 5

Enter the value of Complex Numbers a, b

2 2

Input Values

4 + 5i

2 + 2i

Result

6 + 7i

2 + 3i