Overloading Unary Operators- Overloading binary Operators

Overloading Unary Operator: Let us consider to overload (-) unary operator. In unary operator function, no arguments should be passed. It works only with one class objects. It is a overloading of an operator operating on a single operand.

Example:

Assume that class Distance takes two member object i.e. feet and inches, create a function by which Distance object should decrement the value of feet and inches by 1 (having single operand of Distance Type).

```
/ C++ program to show unary operator overloading
#include <iostream>
using namespace std;
class Distance {
public:
    // Member Object
    int feet, inch;

    // Constructor to initialize the object's value
    Distance(int f, int i)
    {
        this->feet = f;
        this->inch = i;
    }

    // Overloading(-) operator to perform decrement
    // operation of Distance object
    void operator-()
    {
        feet--;
        inch--;
        cout << "\nFeet & Inches(Decrement): " << feet << "'" << inch;
    }
};

// Driver Code
int main()
{
    // Declare and Initialize the constructor
    Distance d1(8, 9);

    // Use (-) unary operator by single operand
    -d1;
    return 0;
}
```

Output:

```
Feet & Inches(Decrement): 7'8
```

Binary Operator Overloading in C++:

An operator which contains two operands to perform a mathematical operation is called the Binary Operator Overloading. It is a polymorphic compile technique where a single operator can perform various functionalities by taking two operands from the programmer or user. There are multiple binary operators like +, -, *, /, etc., that can directly manipulate or overload the object of a class.
Syntax of the Binary Operator Overloading

Following is the Binary Operator Overloading syntax in the C++ Programming language.

```
return_type :: operator binary_operator_symbol (arg)
{
// function definition
}
```

Here, **return_type:** It defines the return type of the function.

operator: It is a keyword of the function overloading.

binary_operator_symbol: It represents the binary operator symbol that overloads a function to perform the calculation.

arg: It defines the argument passed to the function.

Program to perform the addition and subtraction of two complex numbers using the binary (+) and (-) operator

Let's create a program to calculate the addition and subtraction of two complex numbers by overloading the '+' and '-' binary operators in the C++ programming language.

```
/* use binary (+) operator to add two complex numbers. */
#include <iostream>
using namespace std;
class Complex_num
{
    // declare data member or variables
    int x, y;
    public:
        // create a member function to take input
        void inp()
        {
            cout << " Input two complex number: " << endl;
            cin >> x >> y;
        }
        // use binary '+' operator to overload
        Complex_num operator + (Complex_num obj)
        {
            // create an object
            Complex_num A;
            // assign values to object
            A.x = x + obj.x;
            A.y = y + obj.y;
            return (A);
        }
        // overload the binary (-) operator
        Complex_num operator - (Complex_num obj)
        {
            // create an object
            Complex_num A;
            // assign values to object
            A.x = x - obj.x;
            A.y = y - obj.y;
            return (A);
        }
        // display the result of addition
        void print()
        {
            cout << x << " + " << y << "i" << "\n";
        }

        // display the result of subtraction
        void print2()
        {
            cout << x << " - " << y << "i" << "\n";
        }
};
int main ()
{
Complex_num x1, y1, sum, sub; // here we created object of class Addition i.e x1 and y1
    // accepting the values
    x1.inp();
    y1.inp();
    // add the objects
    sum = x1 + y1;
    sub = x1 - y1; // subtract the complex number
    // display user entered values
    cout << "\n Entered values are: \n";
    cout << " \t";
    x1.print();
    cout << " \t";
    y1.print();
    cout << "\n The addition of two complex (real and imaginary) numbers: ";
    sum.print(); // call print function to display the result of addition
    cout << "\n The subtraction of two complex (real and imaginary) numbers: ";
    sub.print2(); // call print2 function to display the result of subtraction
    return 0;
}
```

Output

```
Input two complex numbers:
5
7
Input two complex numbers:
3
5
Entered values are:
        5 + 7i
        3 + 5i
The addition of two complex (real and imaginary) numbers: 8 + 12i
The subtraction of two complex (real and imaginary) numbers: 2 - 2i
```