**Experiment No-07:** Operator Overloading in C++.

### Objectives

- Understand operator overloading in C++.
- Implement operator overloading using the Friend function.

**Example 1:** A C++ program to find the sum of two complex numbers using binary operator overloading.

---

```
// C++ program to demonstrate the working of operator overloading

#include <iostream>
using namespace std;

class Complex {
   private:
    float real;
    float imag;

   public:
    // Constructor to initialize real and imag to 0
    Complex() {
        real = 0;
        imag = 0;

    }

    void input() {
        cout << "Enter real and imaginary parts respectively: ";
        cin >> real;
        cin >> imag;
    }

    // Overload the + operator
    Complex operator + (Complex c) {
        Complex temp;
        temp.real = real + c.real;
        temp.imag = imag + c.imag;
        return temp;
    }

    void output() {
        if (imag < 0)
            cout << "Output Complex number: " << real << imag << "i";
        else
            cout << "Output Complex number: " << real << "+" << imag <<
                "i";
```

```
    }
};

int main() {
    Complex c1, c2, result;

    cout << "Enter first complex number:\n";
    c1.input();

    cout << "Enter second complex number:\n";
    c2.input();

   // c1 calls the operator function
   // c2 is passed as an argument to the function
    result = c1 + c2; // c1.add(c2)
    result.output();

    return 0;
}
```

---

**Example 2:** ++ Operator (Unary Operator) Overloading.

---

```
// Overload ++ when used as prefix and postfix

#include <iostream>
using namespace std;

class Count {
   private:
    int value;

   public:

    // Constructor to initialize count to 5
    Count() {
        value = 8;
    }


    // Overload ++ when used as prefix
    void operator ++ () {
        ++value;
    }


    // Overload ++ when used as postfix
    void operator ++ (int) {
        value++;
    }
```

```
    void display() {
        cout << "Count: " << value << endl;
    }
};

int main() {
    Count c1;

    // Call the "void operator ++ (int)" function
    c1++;
    c1.display();

    // Call the "void operator ++ ()" function
    ++c1;

    c1.display();
    return 0;
}
```

---

**Example 3 :** Return Value from Operator Function (++ Operator).

---

```
#include <iostream>
using namespace std;

class Count {
   private:
    int value;

   public
       :
    // Constructor to initialize count to 5
    Count(){
    value= 2;
    }

    // Overload ++ when used as prefix
    Count operator ++ () {
        Count temp;

        // Here, value is the value attribute of the calling object
        temp.value = ++value;

        return temp;
    }

    // Overload ++ when used as postfix
    Count operator ++ (int) {
        Count temp;
```

```
        // Here, value is the value attribute of the calling object
        temp.value = value++;

        return temp;
    }

    void display() {
        cout << "Count: " << value << endl;
    }
};

int main() {
    Count c1, result;

    // Call the "Count operator ++ ()" function
    result = ++c1;
    result.display();

    // Call the "Count operator ++ (int)" function
    result = c1++;
    result.display();

    return 0;
}
```

**Example 4 :** Operator overloading using Friend Function.

```
#include<iostream>
using namespace std;

class PrePost
{
int a;
public:
// constructor to initialize a
PrePost()
{
    a = 0;
}
void show()
{
cout<<a<<endl;
}
// friend functions
friend PrePost operator++(PrePost x);
friend PrePost operator++(PrePost x, int);
};
```

```
PrePost operator++(PrePost x)
{
    PrePost z;
    cout<<"Prefix "<<endl;
    z.a= ++x.a;
    return z;
}
PrePost operator++(PrePost x, int v)
{
    PrePost z;
    cout<<"Postfix "<<endl;
    z.a = x.a++;
    return z;
}
int main()
{
    PrePost p,x;
    x.show(); p.show();
    x=++p;
    x.show(); p.show();
    x=p++;
    x.show(); p.show();
}
```

---

*** For better understanding please feel free to search on internet because it is the best source of learning. ***

## Practice Exercise

1. Define a class **Distance** with distances in feet and inch and with a print function to print the distance.
a) overload < operator to compare two distances using member function.
b) overload + operator to add two Distances using friend function.
2. Write a C++ program to Overloaded − operator to subtract two complex number.

[Resource Link 1]
[Resource Link 2]