EXPERIMENT-10

OBJECTIVE

Implement Operator overloading concepts.

PROGRAM

Operators Overloading: Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. Operator overloading is used to overload or redefines most of the operators available in C++. It is used to perform the operation on the user-defined data type. For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.

The advantage of Operators overloading is to perform different operations on the same operand.

Operator that cannot be overloaded are as follows:

- Scope operator (::)
- Sizeof
- member selector(.)
- member pointer selector(*)
- ternary operator(?:)

Syntax:

```
return_type class_name  : : operator op(argument_list)
{
    // body of the function.
}
```

Where the **return type** is the type of value returned by the function.

class_name is the name of the class.

operator op is an operator function where op is the operator being overloaded, and the operator is the keyword.

Rules for Operator Overloading:

- Existing operators can only be overloaded, but the new operators cannot be overloaded.

- The overloaded operator contains atleast one operand of the user-defined data type.
- We cannot use friend function to overload certain operators. However, the member function can be used to overload those operators.
- When unary operators are overloaded through a member function take no explicit arguments, but, if they are overloaded by a friend function, takes one argument.
- When binary operators are overloaded through a member function takes one explicit argument, and if they are overloaded through a friend function takes two explicit arguments.

Program to overload the unary operator ++.

```
1. #include <iostream>
2. using namespace std;
3. class Test
4. {
5.    private:
6.       int num;
7.    public:
8.       Test(): num(8){}
9.       void operator ++()       {
10.         num = num+2;
11.      }
12.      void Print() {
13.         cout<<"The Count is: "<<num;
14.      }
15. };
16. int main()
17. {
18.    Test tt;
19.    ++tt;  // calling of a function "void operator ++()"
20.    tt.Print();
```

```
21.     return 0;
22. }
```

Output:

The Count is: 10

Program to overload the binary operators:

```
1.  #include <iostream>
2.  using namespace std;
3.  class A
4.  {
5.
6.      int x;
7.        public:
8.        A(){}
9.      A(int i)
10.   {
11.     x=i;
12.   }
13.   void operator+(A);
14.   void display();
15. };
16.
17. void A :: operator+(A a)
18. {
19.
20.   int m = x+a.x;
21.   cout<<"The result of the addition of two objects is : "<<m;
22.
```

```
23. }
24. int main()
25. {
26.     A a1(5);
27.     A a2(4);
28.     a1+a2;
29.     return 0;
30. }
```

Output:

```
The result of the addition of two objects is : 9
```