**Lab 04 – *Operator Overloading***

1. **Objectives**

The objective of this lab is to teach the students, the use of unary and binary operators for user defined classes i.e. operator overloading.

1. **Outcome** 
   1. At the end of this lab student will know the purpose of Operator Overloading.
   2. Student will be able to use unary and binary operators for user defined classes.
2. **Introduction**

One of the nice features of C++ is that you can give special meanings to operators, when they are used with user-defined classes. This is called operator overloading. You can implement C++ operator overloads by providing special member-functions on your classes that follow a particular naming convention. For example, to overload the + operator for your class, you would provide a member-function named operator+ on your class.

**The operator Keyword:**

The keyword operator is used to overload the ++ operator in this declarator:

**void operator ++ ()**

The return type (void in this case) comes first, followed by the keyword operator, followed by the operator itself (++), and finally the argument list enclosed in parentheses (which are empty here).

The following set of operators is commonly overloaded for user-defined classes:

1. **Examples**

|  |  |
| --- | --- |
| ++ -- (Increment and Decrement Operators) | Unary Operators |
| = (Assignment Operator)  + - \* (Binary Arithmetic Operators) |  |
| += -= \*= (Compound Assignment Operators)  == != <> (Comparison Operators) | Binary Operators |
| <<>> (Insertion and Extraction Operators) | Stream Operators |

* 1. **Example of Unary Operators Overloading**

The unary operators operate on a single operand and following are the examples of Unary operators:

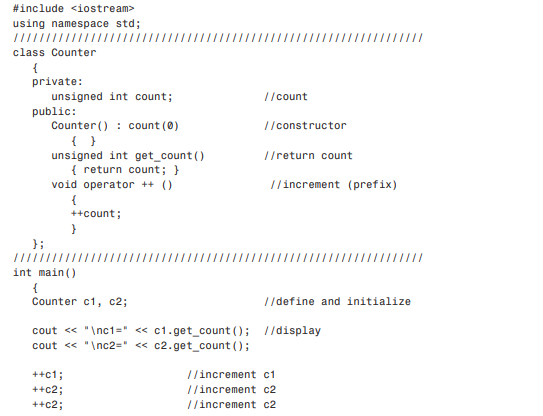
* + - The increment (++) and decrement (--) operators.
    - The unary minus (-) operator.
    - The logical not (!) operator.

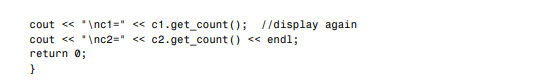
The unary operators operate on the object for which they were called and normally, this operator appears on the left side of the object, as in !obj, -obj, and ++obj but sometime they can be used as postfix as well like obj++ or obj--.

Following example explain how increment (++) operator can be overloaded for prefix as well as postfix usage. Similar way you can overload operator (--).

**Prefix Notation:**

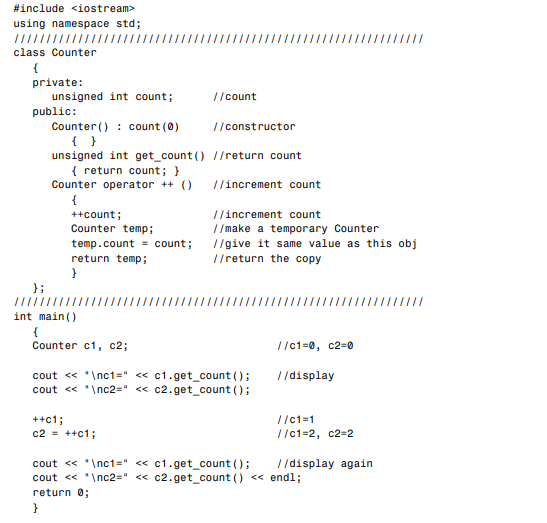
**Example #01**



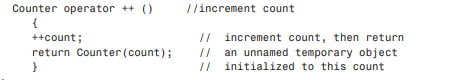


**Example #02**

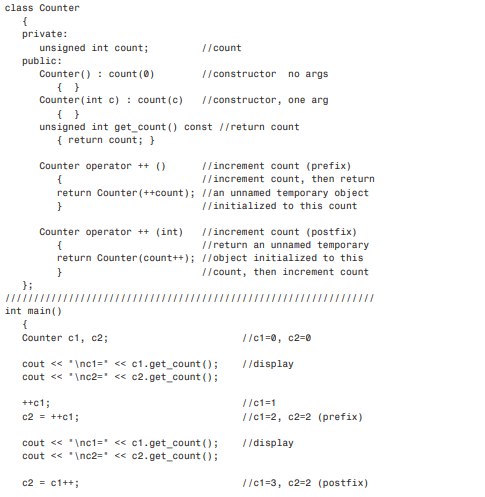
* **Method #01**

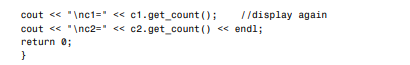


* **Method #02**



**Postfix Notation:**

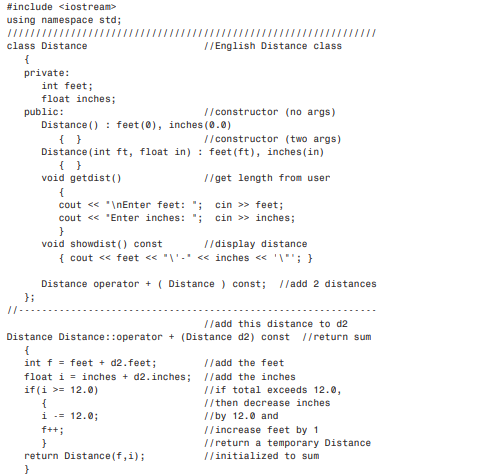


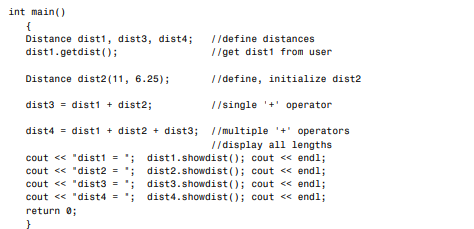


**4.2.Example of Binary Operators Overloading**

The binary operators take two arguments and following are the examples of Binary operators. You use binary operators very frequently like addition (+) operator, subtraction (-) operator and division (/) operator.

Following example explain how addition (+) operator can be overloaded. Similar way you can overload subtraction (-) division (/) or any other binary operators.





1. **In-Lab Tasks** 
   1. To the Distance class in the above given example in this chapter, add an overloaded - operator that subtracts two distances. It should allow statements like dist3= dist1-dist2;. Assume that the operator will never be used to subtract a larger number from a smaller one (that is, negative distances are not allowed).
   2. Write a program for class time which overloads the operator + to add two times instead of using function. The class time should contain hours, minutes and seconds as parameters.
2. **Home Tasks** 
   1. Create a class Int. Overload four integer arithmetic operators (+, -, \*, and /) so that they operate on objects of type Int. If the result of any such arithmetic operation exceeds the normal range of ints (in a 32-bit environment)— from 2,147,483,648 to –2,147,483,647—have the operator print a warning and terminate the program. Such a data type might be useful where mistakes caused by arithmetic overflow are unacceptable.

Hint: To facilitate checking for overflow, perform the calculations using type long double. Write a program to test this class.