Ekaba Bisong
Programming in C++
University of Calabar

Lesson Note #27
June 03, 2015

Adapted from C++ How To Program edited for our own purposes

Increment and Decrement Operators

In addition to the arithmetic assignment operators, C++ also provides two unary operators for adding 1 to or subtracting 1 from the value of a numeric variable. These are the unary increment operator, ++, and the unary decrement operator, --, which are summarized in the table below.

Operator	Called	Sample expression	Explanation
++	preincrement	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postincrement	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	predecrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postdecrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

A program can increment by 1 the value of a variable called c using the increment operator, ++, rather than the expression c = c + 1 or c += 1. An increment or decrement operator that's prefixed to (placed before) a variable is referred to as the prefix increment or prefix decrement operator, respectively. An increment or decrement operator that's postfixed to (placed after) a variable is referred to as the postfix increment or postfix decrement operator, respectively.

The following code sample demonstrates the difference between the prefix increment and postfix increment versions of the ++ increment operator. The decrement operator (--) works similarly.

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

int main()
{
   int c;

   // demonstrate postincrement
   c = 5; // assign 5 to c
   cout << c << endl; // print 5
   cout << c++ << endl; // print 5 then postincrement
   cout << c << endl; // print 6

   cout << endl; // skip a line

   // demonstrate preincrement
   c = 5; // assign 5 to c
   cout << c << endl; // print 5
   cout << ++c << endl; // preincrement then print 6
   cout << c << endl; // print 6
} // end main
```

```
5
5
6

5
6
6
```