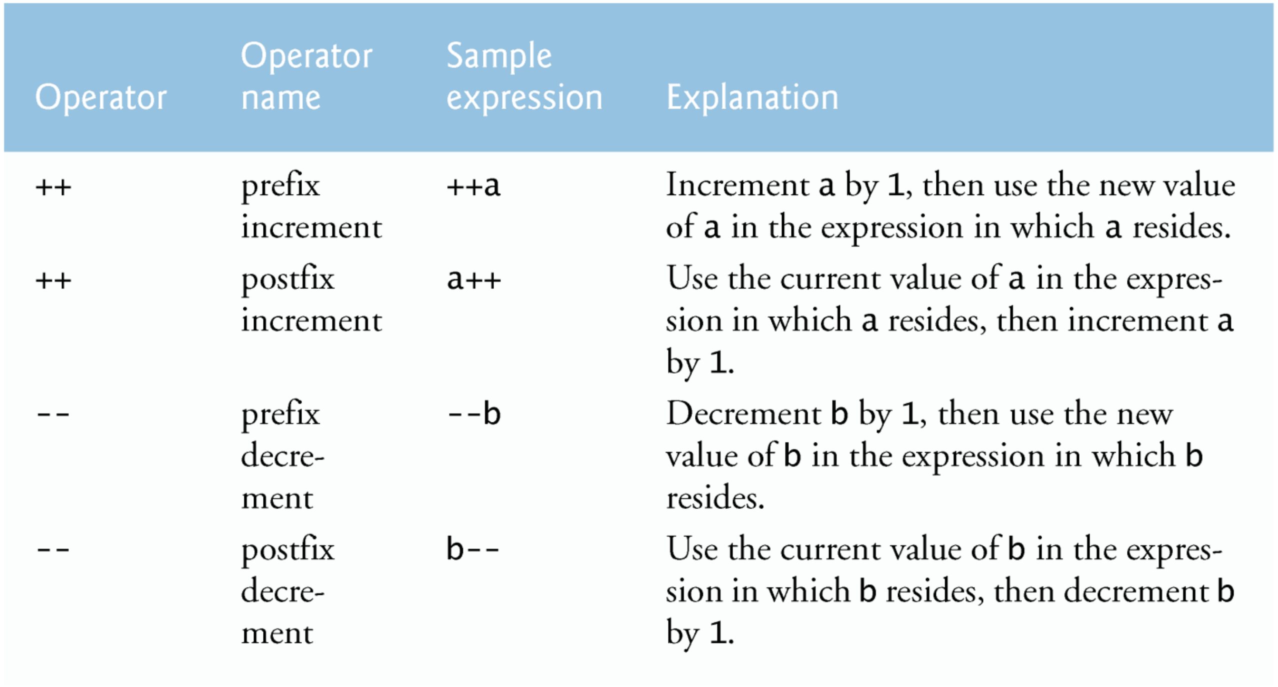
C++: Increment/Decrement Operators, Switch and Repetition Statements (while, do-while and for loop)

“increment” and decrement operator



Operator	Operator name	Sample expression	Explanation
++	prefix increment	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postfix increment	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	prefix decrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postfix decrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Fig. 3.12 | Increment and decrement operators.

© 1992-2015 by Pearson Education, Inc. All Rights Reserved.

Exercise: What’s the output of this program?

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

int main()
{
  int quantity = 10 ;
  cout << quantity++ << endl;
  cout << ++quantity << endl;
  cout << quantity   << endl;

  int sum = 100 + quantity++ ;
  int total = 100 + ++quantity ;
  cout << sum        << endl;
  cout << total      << endl;
  cout << quantity;

  return 0;
}
```

“Switch” statement

Perform actions based on the possible values of a variable or expression. The test must be of an integral value (byte, character, short or integer)

```
#include <iostream>
using namespace std;
int main()
{
  char theOperator ;
  cout << "Please enter the operator: ";
  cin >> theOperator;

  switch (theOperator)
  {
    case '+':  cout << "Addition"       << endl; break;
    case '-':  cout << "Subtraction"    << endl; break;
    case '*':  cout << "Multiplication" << endl; break;
    case '/':  cout << "Division"       << endl; break;
    case '%':  cout << "Modulo"         << endl; break;
  }
  return 0;
}
```

Syntax:

```
switch (expression)
{
    case  value-1:
    case value-2:
    …
    default:
}
```

“While” repetition statement

Repeat an action while a condition remains true

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

int main()
{
   char theOperator = ' ' ;

   while (theOperator != 'x')
   {
      cout << "Please enter the operator: ";
      cin >> theOperator;

      cout << "Operator: " << theOperator << endl;
   }
   return 0;
}
```

Syntax:

```
while (test-condition)
{
}
```

“do-while” repetition statement

Tests the loop-continuation condition after executing the loop’s body; therefore, the body always executes at least once.

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

int main()
{
  char theOperator = ' ' ;

  do
  {
    cout << "Please enter the operator: ";
    cin >> theOperator;

    cout << "Operator: " << theOperator << endl;
  } while (theOperator != 'x');
  return 0;
}
```

Syntax:

```
do
{
} while (test-condition);
```

“for” repetition statement

Syntax:

```
for (initialization; loopContinuationCondition; increment)
  statement;
```

- the initialization expression names the loop’s control variable and optionally provides its initial value.
- loopContinuationCondition determines whether the loop should continue executing
- increment modifies the control variable’s value, so that the loop-continuation condition eventually becomes false.
- The two semicolons in the for header are required.
- All three expressions in a for header are optional.

Repetition/Looping Summary

Repetition is implemented in one of three ways:

- **“while” statement**
- **“do-while” statement**
- **“for” statement**

Any repetition can be implemented using any of these 3 statements: while, do-while and for.