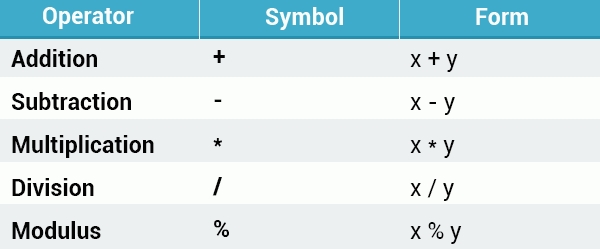
**Arithmetic Operators**

C++ supports these arithmetic operators.



The addition operator adds its operands together.

You can use multiple arithmetic operators in one line.

The subtraction operator subtracts one operand from the other.

The multiplication operator multiplies its operands.

The division operator divides the first operand by the second. Any remainder is dropped in order to return an integer value.

If one or both of the operands are floating point values, the division operator performs floating point division.  
Dividing by 0 will crash your program.

The modulus operator (%) is informally known as the remainder operator because it returns the remainder after an integer division.

**Operator Precedence**

Operator **precedence**determines the grouping of terms in an expression, which affects how an expression is evaluated. Certain operators take higher precedence over others; for example, the multiplication operator has higher precedence over the addition operator.

Ex

Int x =5+2\*2;

Cout<<x;

The program above evaluates 2\*2 first, and then adds the result to 5.  
As in mathematics, using **parentheses**alters operator precedence.

Parentheses force the operations to have higher precedence. If there are parenthetical expressions nested within one another, the expression within the innermost parentheses is evaluated first.

If none of the expressions are in parentheses, **multiplicative**(multiplication, division, modulus) operators will be evaluated before **additive**(addition, subtraction) operators.

**Assignment Operators**

The simple **assignment**operator (=) assigns the right side to the left side.  
  
C++ provides shorthand operators that have the capability of performing an operation and an assignment at the same time.

**For example:**

int x = 10;   
**x += 4**; // equivalent to x = x + 4   
**x -= 5**; // equivalent to x = x – 5

Assignment operator (=) assigns the right side to the left side.

The same shorthand syntax applies to the multiplication, division, and modulus operators.

x \*= 3; // equivalent to x = x \* 3   
x /= 2; // equivalent to x = x / 2   
x %= 4; // equivalent to x = x % 4

The same shorthand syntax applies to the multiplication, division, and modulus operators.

The **increment**operator is used to increase an integer's value by one, and is a commonly used C++ operator.

**x++;** //equivalent to x = x + 1

The increment operator is used to increase an integer's value by one.

The increment operator has two forms, **prefix**and **postfix**

++x; // prefix   
x++; // postfix

**Prefix**increments the value, and then proceeds with the expression.  
**Postfix**evaluates the expression and then performs the incrementing.  
  
**Prefix example:**

x = 5;   
y = **++x**;   
// x is 6, y is 6

**Postfix example:**

x = 5;   
y = **x++**;   
// x is 6, y is 5

The **prefix**example increments the value of x, and then assigns it to y.  
The **postfix**example assigns the value of x to y, and then increments it.

The **decrement**operator (--) works in much the same way as the increment operator, but instead of increasing the value, it decreases it by one.

--x; // prefix   
x--; // postfix

The decrement operator (--) works in much the same way as the increment operator.