# Simple C++ Maths

This lesson introduces the basic operators used in C++ like subtraction, addition, division, and multiplication

**We'll cover the following**

Math in C++ is very simple. Keep in mind that C++ mathematical operations follow a particular order much the same as high school math.

For example, multiplication and division take precedence over addition and subtraction. The *order* in which these operations are evaluated can be changed using *parentheses*.

## Operations in C++ #

The arithmetic operators in C++ are listed below.

| Symbols | Arithmetic operators |
|---|---|
| + | add |
| - | subtract |
| / | divide |
| * | multiply |
| % | modulus division |

| | |
|---|---|
| += | add and assign |
| -= | subtract and assign |
| /= | divide and assign |
| *= | multiply and assign |
| %= | mod and assign |

## Example #

### Adding, Subtracting, Multiplying and Dividing

Let’s take a look at how to use these operations while coding in C++.

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

int main() {
    int myInt = 100;

    myInt = myInt / 10; //myInt is now 10
    cout <<"Value of myInt after division by 10 is: " <<myInt << endl;
    myInt = myInt * 10; //myInt is back to 100
    cout <<"Value of myInt after multiplication by 10 is: " <<myInt << endl;
    myInt = myInt + 50; //myInt is up to 150
    cout <<"Value of myInt after addition of 50 is: " <<myInt << endl;
    myInt = myInt - 50; //myInt is back to where it started
    cout <<"Value of myInt after subtraction of 50 is: " <<myInt << endl;

    myInt = myInt + 100 * 2; // myInt is now 300 because multiplication takes precedence over addi
    cout << "Value of myInt after adding 100 and multiplying by 2 is: "<<myInt<<endl;
    myInt = (myInt + 100) * 2; // myInt is now 800 because we have changed the precedence using pa
    cout << "Value of myInt after doing the same operations using paranthesis is: "<<myInt<<endl;

    myInt -= 10; // myInt is now 790 because this line is the short-hand for myInt = myInt - 10;
    cout << "Value of myInt after -=10 operation is: "<<myInt<<endl;
    myInt = myInt % 100; // myInt is now 90 because % is modulus operator
    cout << "Value of myInt after taking modulus with 100 is: "<<myInt<<endl;

    cout << "Value of myInt after all operations is: " <<myInt << endl;


    return 0; //Passing message to the Operating System saying that the code has been successfully
}
```

## Operator Precedence #

**Operator precedence** specifies the order of operations in expressions that contain more than one operator. Operator associativity specifies whether, in an expression that contains multiple operators with the same precedence, an operand is grouped with the one on its left or the one on its right. The following table shows the precedence and associativity of **C++** operators (from highest to lowest precedence).

| Operators | Description | Associativity |
|---|---|---|
| | | Right to left |
| + - | Unary Plus and minus | |
| ! ~ | Logical NOT and bitwise NOT | |
| * | Indirection (dereference) | |
| & | Address-of | |
| new, new[] | Dynamic memory allocation | |
| delete, delete[] | Dynamic memory deallocation | |
| = | Direct assignment | |
| += -= | Assignment by sum and difference | |
| *= /= %= | Assignment by product, quotient, and remaind | |

| | | |
|---|---|---|
| | er | |
| <<= >>= | Assignment by bitwise l eft shift and right shift | |
| &= ^= = | Assignment by bitwise AND, XOR, and OR | |
| ++ -- | Suffix/postfix increment and decrement | Left to right |
| * / % | Multiplication, division, and remainder | |
| + - | Addition and subtraction | |
| << >> | Bitwise left shift and right shift | |
| < <= | For relational operators | |
| > >= | For relational operators | |
| == != | For relational operators | |
| ^ | Bitwise XOR (exclusive or) | |
| && | Logical AND | |
| \|\| | Logical OR | |
| \| | Bitwise OR (inclusive | |

|  |  |  |
|---|---|---|
| \| | or) |  |
| ?: | Ternary conditional |  |

## Example

If you try to compute the given expression:

`20 + 30 * 2 / 10 - 3`

According to the C++ operator precedence, it will first compute `30 * 2 = 60`. Then divide `60 / 10 = 6`. Both multiplication and division have the same precedence but evaluation is done from left to right. After that, add `20 + 6 = 26` and subtract `26 - 3 = 23`. Again addition has the same precedence as subtraction but evaluation is done from left to right . So, the final answer is `23`.

Give it a try!

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

int main() {
  // 20 + 30 * 2 / 10 - 3
  cout << "Answer is " <<  20 + 30 * 2 / 10 - 3 << endl;
  return 0;
}
```

Finding it interesting so far? Let’s take a look at some other cool operations you can perform in C++ in the next lesson!