C++ for Loop

In computer programming, loops are used to repeat a block of code.

For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.

That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.

There are 3 types of loops in C++.

- `for` loop
- `while` loop
- `do...while` loop

This tutorial focuses on C++ `for` loop. We will learn about the other type of loops in the upcoming tutorials.

C++ for loop

The syntax of for-loop is:

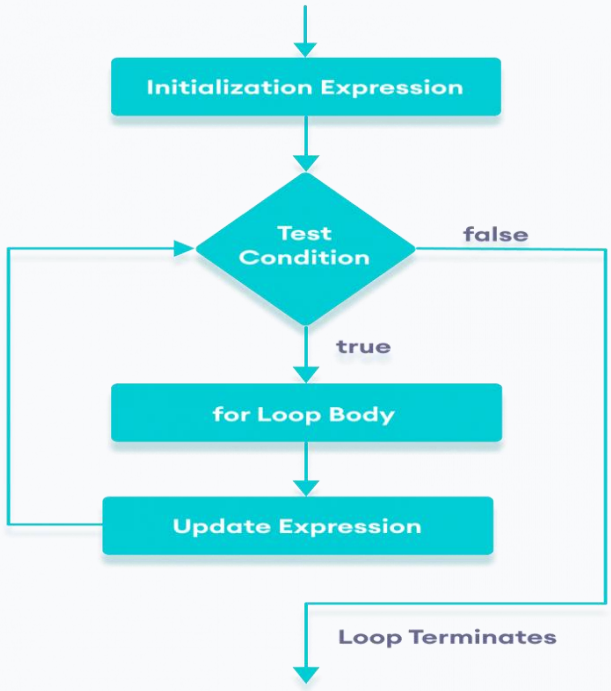
```
for (initialization; condition; update) {
    // body of-loop
}
```

Here,

- `initialization` - initializes variables and is executed only once
- `condition` - if `true`, the body of `for` loop is executed if `false`, the for loop is terminated
- `update` - updates the value of initialized variables and again checks the condition

To learn more about `conditions`, check out our tutorial on C++ Relational and Logical Operators.

Flowchart of for Loop in C++



Example 1: Printing Numbers From 1 to 5

```
#include <iostream>

using namespace std;

int main() {
        for (int i = 1; i <= 5; ++i) {
        cout << i << " ";
    }
    return 0;
}
```

Output

```
1 2 3 4 5
```

Here is how this program works

Iteration	Variable	i <= 5	Action
1st	`i = 1`	`true`	`1` is printed. `i` is increased to `2`.
2nd	`i = 2`	`true`	`2` is printed. `i` is increased to `3`.
3rd	`i = 3`	`true`	`3` is printed. `i` is increased to `4`.
4th	`i = 4`	`true`	`4` is printed. `i` is increased to `5`.
5th	`i = 5`	`true`	`5` is printed. `i` is increased to `6`.
6th	`i = 6`	`false`	The loop is terminated

Example 2: Display a text 5 times

```
// C++ Program to display a text 5 times
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 5; ++i) {
        cout <<  "Hello World! " << endl;
    }
    return 0;
}
```

Output

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Here is how this program works

Iteration	Variable	i <= 5	Action
1st	`i = 1`	`true`	`Hello World!` is printed and `i` is increased to `2`.
2nd	`i = 2`	`true`	`Hello World!` is printed and `i` is increased to `3`.
3rd	`i = 3`	`true`	`Hello World!` is printed and `i` is increased to `4`.
4th	`i = 4`	`true`	`Hello World!` is printed and `i` is increased to `5`.
5th	`i = 5`	`true`	`Hello World!` is printed and `i` is increased to `6`.
6th	`i = 6`	`false`	The loop is terminated

Example 3: Find the sum of first n Natural Numbers

```
// C++ program to find the sum of first n natural numbers
// positive integers such as 1,2,3,...n are known as natural numbers

#include <iostream>
using namespace std;
int main() {
    int num, sum;
    sum = 0;

    cout << "Enter a positive integer: ";
    cin >> num;

    for (int i = 1; i <= num; ++i) {
        sum += i;
    }

    cout << "Sum = " << sum << endl;

    return 0;
}
```

Output

```
Enter a positive integer: 10
Sum = 55
```

In the above example, we have two variables `num` and `sum`. The `sum` variable is assigned with `0` and the `num` variable is assigned with the value provided by the user.

Note that we have used a `for` loop.

```
for(int i = 1; i <= num; ++i)
```

Here,

- `int i = 1`: initializes the `i` variable
- `i <= num`: runs the loop as long as `i` is less than or equal to `num`
- `++i`: increases the `i` variable by 1 in each iteration

When `i` becomes `11`, the `condition` is `false` and `sum` will be equal to `0 + 1 + 2 + ... + 10`.

C++ Infinite for loop

If the `condition` in a `for` loop is always `true`, it runs forever (until memory is full). For example,

```
// infinite for loop
for(int i = 1; i > 0; i++) {
    // block of code
}
```

In the above program, the `condition` is always `true` which will then run the code for infinite times.

C++ Program to Calculate Sum of Natural Numbers

Positive integers 1, 2, 3, 4... are known as natural numbers.

This program takes a positive integer from user(suppose user entered `n`) then, this program displays the value of 1+2+3+....+n.

Example: Sum of Natural Numbers using loop

```
#include <iostream>
using namespace std;
int main() {
    int n, sum = 0;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= n; ++i) {
        sum += i;
    }
    cout << "Sum = " << sum;
    return 0;
}
```

Output

```
Enter a positive integer: 50
Sum = 1275
```

This program assumes that user always enters positive number.

If user enters negative number, **Sum = 0** is displayed and program is terminated.

C++ Program to Find Factorial

The factorial of a number is the product of all the integers from **1** up to that number. The factorial can only be defined for positive integers.
The factorial of a negative number doesn't exist. And the factorial of **0** is **1**.
For example,

The factorial of a positive number `n`, say `5`, is denoted by `5!` and is given by:

```
5! = 1 * 2 * 3 * 4 * 5 = 120
```

So, the Mathematical logic for factorial is:

```
n! = 1 * 2 * 3 * ... * n
n! = 1 if n = 0 or n = 1
```

In this program, the user is asked to enter a positive integer. Then the factorial of that number is computed and displayed on the screen.

Example: Find the Factorial of a Given Number

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

int main() {
    int n;
    long factorial = 1.0;

    cout << "Enter a positive integer: ";
    cin >> n;

    if (n < 0)
        cout << "Error! Factorial of a negative number doesn't exist.";
    else {
        for(int i = 1; i <= n; ++i) {
            factorial *= i;
        }
        cout << "Factorial of " << n << " = " << factorial;
    }

    return 0;
}
```

Output

```
Enter a positive integer: 4
Factorial of 4 = 24
```

In this program, we take a positive integer from the user and compute the factorial using `for` loop. We print an error message if the user enters a negative number.

We declare the type of factorial variable as `long` since the factorial of a number may be very large.

When the user enters a positive integer (say **4**), `for` loop is executed and computes the factorial. The value of `i` is initially `1`.

The program runs until the statement `i <= n` becomes `false`. This prints `Factorial of 4 = 24` on the screen. Here's how the program executes when `n = 4`.

i <= 4	**fact *= i**
1 <= 4	fact = 1 * 1 = 1
2 <= 4	fact = 1 * 2 = 2
3 <= 4	fact = 2 * 3 = 6
4 <= 4	fact = 6 * 4 = 24
5 <= 4	Loop terminates.

Note: This program can calculate the factorial only up to the number **20**. Beyond that, the program can no longer calculate the factorial as the results exceed the capacity of the `factorial` variable.

C++ Program to Generate Multiplication Table

Example 1: Display Multiplication Table up to 10

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

int main() {

    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    // run a loop from 1 to 10
    // print the multiplication table
    for (int i = 1; i <= 10; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
```

Run Code

Output

```
Enter a positive integer: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
```

```
5 * 10 = 50
```

This program above computes the multiplication table up to **10** only.

Example 2: Display Multiplication Table up to a Given Range

The program below is a modification of the above program in which the user is asked to enter the range up to which the multiplication table should be displayed.

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

int main() {

    int n, range;

    cout << "Enter an integer: ";
    cin >> n;

    cout << "Enter range: ";
    cin >> range;

    for (int i = 1; i <= range; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
```

Run Code

Output

```
Enter an integer: 8
Enter range: 12
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
8 * 11 = 88
8 * 12 = 96
```