# *Loops in c++*

we will learn about the C++ for loop and its working with the help of some examples.

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,

# *Loops in c++*

- `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.

## 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
```

# *Loops in c++*

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

### 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
}
```

## C++ while and do...while Loop

### C++ while Loop

The syntax of the `while` loop is:

```
while (condition) {
    // body of the loop
}
```

Here,

- A `while` loop evaluates the `condition`
- If the `condition` evaluates to `true`, the code inside the `while` loop is executed.
- The `condition` is evaluated again.

# *Loops in c++*

- This process continues until the `condition` is `false`.
- When the `condition` evaluates to `false`, the loop terminates.

## Example 1: Display Numbers from 1 to 5

```
// C++ Program to print numbers from 1 to 5

#include <iostream>

using namespace std;

int main() {
    int i = 1;

    // while loop from 1 to 5
    while (i <= 5) {
        cout << i << " ";
        ++i;
    }

    return 0;
}
```

**Output**

```
1 2 3 4 5
```

Here is how the program works.

| Iteration | Variable | i <= 5 | Action |
|---|---|---|---|
| 1st | `i = 1` | `true` | `1` is printed and `i` is increased to `2`. |
| 2nd | `i = 2` | `true` | `2` is printed and `i` is increased to `3`. |
| 3rd | `i = 3` | `true` | `3` is printed and `i` is increased to `4` |

| | | | |
|---|---|---|---|
| 4th | `i = 4` | `true` | `4` is printed and `i` is increased to `5`. |
| 5th | `i = 5` | `true` | `5` is printed and `i` is increased to `6`. |
| 6th | `i = 6` | `false` | The loop is terminated |

---

## Example 2: Sum of Positive Numbers Only

```
// program to find the sum of positive numbers
// if the user enters a negative number, the loop ends
// the negative number entered is not added to the sum

#include <iostream>
using namespace std;

int main() {
    int number;
    int sum = 0;

    // take input from the user
    cout << "Enter a number: ";
    cin >> number;

    while (number >= 0) {
        // add all positive numbers
        sum += number;

        // take input again if the number is positive
        cout << "Enter a number: ";
        cin >> number;
    }

    // display the sum
    cout << "\nThe sum is " << sum << endl;
    return 0;
}
```

**Output**

```
Enter a number: 6
Enter a number: 12
Enter a number: 7
Enter a number: 0
Enter a number: -2

The sum is 25
```

## C++ do...while Loop

The `do...while` loop is a variant of the `while` loop with one important difference: the body of `do...while` loop is executed once before the `condition` is checked. Its syntax is:

```
do {
   // body of loop;
}
while (condition);
```

### Example 3: Display Numbers from 1 to 5

```
// C++ Program to print numbers from 1 to 5

#include <iostream>

using namespace std;

int main() {
    int i = 1;
```

# Loops in c++

```
    // do...while loop from 1 to 5
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);

    return 0;
}
```

**Output**

```
1 2 3 4 5
```

Here is how the program works.

| Iteration | Variable | i <= 5 | Action |
|---|---|---|---|
| | `i = 1` | not checked | `1` is printed and `i` is increased to 2 |
| 1st | `i = 2` | `true` | `2` is printed and `i` is increased to 3 |
| 2nd | `i = 3` | `true` | `3` is printed and `i` is increased to 4 |
| 3rd | `i = 4` | `true` | `4` is printed and `i` is increased to 5 |
| 4th | `i = 5` | `true` | `5` is printed and `i` is increased to **6** |
| 5th | `i = 6` | `false` | The loop is terminated |

# *Loops in c++*

## Example 4: Sum of Positive Numbers Only

```cpp
// program to find the sum of positive numbers
// If the user enters a negative number, the loop ends
// the negative number entered is not added to the sum

#include <iostream>
using namespace std;

int main() {
    int number = 0;
    int sum = 0;

    do {
        sum += number;

        // take input from the user
        cout << "Enter a number: ";
        cin >> number;
    }
    while (number >= 0);

    // display the sum
    cout << "\nThe sum is " << sum << endl;

    return 0;
}
```

### **Output 1**

```
Enter a number: 6
Enter a number: 12
Enter a number: 7
Enter a number: 0
Enter a number: -2

The sum is 25
```

## **for vs while loops**

# *Loops in c++*

A `for` loop is usually used when the number of iterations is known. For example,

```
// This loop is iterated 5 times
for (int i = 1; i <=5; ++i) {
   // body of the loop
}
```

Here, we know that the for-loop will be executed 5 times.

However, `while` and `do...while` loops are usually used when the number of iterations is unknown. For example,

```
while (condition) {
    // body of the loop
}
```

## C++ break Statement

### Example 1: break with for loop

```
// program to print the value of i

#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 5; i++) {
        // break condition
        if (i == 3) {
            break;
        }
        cout << i << endl;
    }
return 0;
}
```

**Output**

```
1
2
```

## Example 2: break with while loop

```
// program to find the sum of positive numbers
// if the user enters a negative numbers, break ends the loop
// the negative number entered is not added to sum

#include <iostream>
using namespace std;

int main() {
    int number;
    int sum = 0;

    while (true) {
        // take input from the user
        cout << "Enter a number: ";
        cin >> number;

        // break condition
        if (number < 0) {
            break;
        }

        // add all positive numbers
        sum += number;
    }

    // display the sum
    cout << "The sum is " << sum << endl;

    return 0;
}
```

# *Loops in c++*

**Output**

```
Enter a number: 1
Enter a number: 2
Enter a number: 3
Enter a number: -5
The sum is 6.
```

## Nested for Loop

```
// C++ program to display 7 days of 3 weeks

#include <iostream>
using namespace std;

int main() {
    int weeks = 3, days_in_week = 7;

    for (int i = 1; i <= weeks; ++i) {
        cout << "Week: " << i << endl;

        for (int j = 1; j <= days_in_week; ++j) {
            cout << "    Day:" << j << endl;
        }
    }

    return 0;
}
```

```
Week: 1
    Day:1
    Day:2
    Day:3
    ... .. ...
Week: 2
    Day:1
```

# *Loops in c++*

```
    Day:2
    Day:3
    ... ... ..
```

## C++ continue Statement

### Working of C++ continue Statement

### Example 1: continue with for loop

In a `for` loop, `continue` skips the current iteration and the control flow jumps to the `update` expression.

```
// program to print the value of i

#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 5; i++) {
        // condition to continue
        if (i == 3) {
            continue;
        }

        cout << i << endl;
    }

    return 0;
}
```

**Output**

```
1
```

```
2
4
5
```