# Loops in C and C++

<br>

In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things.

Loops are supported by all modern programming languages, though their implementations and syntax may differ. Two of the most common types of loops are the while loop and the for loop.

<br>

### Introduction to Loops in C++

Loop statements in C++ run a certain piece of code or statement many times, and are mostly used to shorten code by performing the same function multiple times, as well as to eliminate code redundancy. C++ provides a variety of loops, including for loops, while loops, and do-while loops, each with its own syntax, benefits, and applications. The loop is a control structure in programming that is used when we wish to execute a block of code many times. It generally keeps running until and unless a certain end condition is met.

If we didn't have loops, we'd have to print a repeating block of statements using the iterative approach, which would look like this:

<br>

```// C program

#include <stdio.h>

int main()

{

printf( "Hello World\n");

printf( "Hello World\n");

printf( "Hello World\n");

printf( "Hello World\n");

printf( "Hello World\n");

return 0;

}

```

<br>

```

// C++ Code

#include <iostream>

using namespace std;

int main()

{

cout << " Hello World \n";

cout << " Hello World \n";

cout << " Hello World \n";

cout << " Hello World \n";

cout << " Hello World \n";

}

```

<br>

### Types of Loops

\* For loop

\* While Loop

\* do While Loop

<br>

### For Loop

A for loop is a repetition control structure that allows us to create a loop that runs a set number of times. The loop allows us to do an arbitrary number of steps in a single line.

<br>

#### Syntax

```

for (initialization expr; test expr; update expr)

{

// body of the loop

// statements we want to execute

}

```

#### Flow Diagram:

<p align="center">

<img width="50%" height="50%" src="https://spazztech.net/images/for-loop-flow-chart.jpg?crc=3844047426">

</p>

<br>

#### Examples

```// C program to illustrate for loop

#include <stdio.h>

int main()

{

int i=0;

for (i = 1; i <= 10; i++)

{

printf( "Hello World\n");

}

return 0;

}

```

```

// C++ program to illustrate for loop

#include <iostream>

using namespace std;

int main()

{

for (int i = 1; i <= 10; i++)

{

cout << "Hello World\n";

}

return 0;

}

```

#### Output

```

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

Hello World

```

<br>

### While Loop

While loops are also entrance controlled loops in which we check the condition we provide before starting the loop. The distinction is that we employ When we know the number of times the loop's body needs to run, we use for loops; but, when we don't know the exact number of times the loop's body needs to run, we use while loops. The loop's execution is stopped when the test condition is met.

#### Syntax

```

initialization expression;

while (test\_expression)

{

// statements

update\_expression;

}

```

#### Flow Diagram:

<p align="center">

<img width="50%" height="50%" src="https://spazztech.net/images/while-loop-flow-chart.jpg?crc=4292634586">

</p>

#### Examples

```

// C program to illustrate while loop

#include <stdio.h>

int main()

{

// initialization expression

int i = 1;

// test expression

while (i < 6)

{

printf( "Hello World\n");

// update expression

i++;

}

return 0;

}

```

<br>

```

// C++ program to illustrate while loop

#include <iostream>

using namespace std;

int main()

{

// initialization expression

int i = 1;

// test expression

while (i < 6)

{

cout << "Hello World\n";

// update expression

i++;

}

return 0;

}

```

#### Output

```

Hello World

Hello World

Hello World

Hello World

Hello World

```

<br>

### do-While Loop

The do while loop is an exit controlled loop, which means that the test condition is checked after the loop has been executed, at the end of the loop's body. As a result, regardless of whether the test condition is true or false, the body executes at least once. This is the primary distinction between a while loop and a do while loop. The condition is checked before being verified in a while loop, but in a do while loop, the condition is confirmed at the end of the loop's body.

#### Syntax

```

initialization expression;

do

{

// statements

update\_expression;

} while (test\_expression);

```

<br>

### Flow Diagram:

<p align="center">

<img width="50%" height="50%" src="https://spazztech.net/images/do-while-loop-flow-chart.jpg?crc=4064782943">

</p>

```

// C program to illustrate do-while loop

#include <stdio.h>

int main()

{

int i = 2; // Initialization expression

do

{

// loop body

printf( "Hello World\n");

// update expression

i++;

} while (i < 1); // test expression

return 0;

}

```

<br>

```

// C++ program to illustrate do-while loop

#include <iostream>

using namespace std;

int main()

{

int i = 2; // Initialization expression

do

{

// loop body

cout << "Hello World\n";

// update expression

i++;

} while (i < 1); // test expression

return 0;

}

```

#### Output

```

Hello World

```

#### Infinite Loop

An infinite loop, also known as an endless loop, is a loop that runs indefinitely because it lacks a sufficient exit condition. When a test condition isn't stated correctly, it evaluates to true all of the time. This is generally due to a programming mistake.

#### Syntax

```

#include <iostream>

using namespace std;

int main ()

{

int i;

for ( ; ; )

{

cout << "This loop run forever.\n";

}

}

```

#### Output

```

This loop will run forever.

This loop will run forever.

...................

````