C - Loops

Loops are a programming construct that denote a block of one or more statements that are repeatedly executed a specified number of times, or till a certain condition is reached.

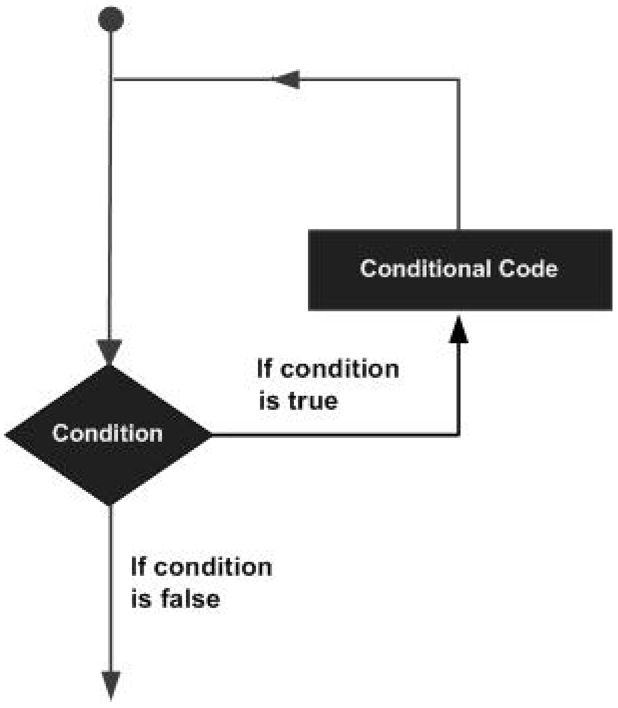
Repetitive tasks are common in programming, and loops are essential to save time and minimize errors. In C programming, the keywords **while**, **do–while** and **for** are provided to implement loops.

Looping constructs are an important part of any processing logic, as they help in performing the same process again and again. A C programmer should be well acquainted with implementing and controlling the looping construct.

Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times.

Flowchart of C Loop Statement

Given below is the general flowchart of a loop statement which is applicable to any programming language −



The statements in a C program are always executed in a top-to-bottom manner. If we ask the compiler to go back to any of the earlier steps, it constitutes a loop.

Example: Loops in C

To understand the need of loops in a program, consider the following snippet −

Open Compiler

```c
#include <stdio.h>
int main (){

   // local variable definition
   int a = 1;

   printf("a: %d\n", a);
   a++;

   printf("a: %d\n", a);
   a++;

   printf("a: %d\n", a);
   a++;

   printf("a: %d\n", a);
   a++;

   printf("a: %d\n", a);

   return 0;
}
```

Output

On running this code, you will get the following output −

```
a: 1
a: 2
a: 3
a: 4
a: 5
```

The program prints the value of "a", and increments its value. These two steps are repeated a number of times. If you need to print the value of "a" from 1 to 100, it is not desirable to manually repeat these steps in the code. Instead, we can ask the compiler to repeatedly execute these two steps of printing and incrementing till it reaches 100.

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Example: Using While Loop in C

You can use **for**, **while** or **do-while** constructs to repeat a loop. The following program shows how you can print 100 values of "a" using the "while" loop in C −

Open Compiler

```
#include <stdio.h>

int main () {

   // local variable definition
   int a = 1;

   while (a <= 100){
      printf("a: %d\n", a);
      a++;
   }

   return 0;
}
```

Output

Run this code and check the output −

```
a: 1
a: 2
a: 3
a: 4
.....
.....
a: 98
a: 99
a: 100
```

If a step redirects the program flow to any of the earlier steps, based on any condition, the loop is a conditional loop. The repetitions will stop as soon as the controlling condition turns false. If the redirection is done without any condition, it is an **infinite loop**, as the code block repeats forever.

Parts of C Loops

To constitute a loop, the following elements are necessary −

- Looping statement (**while**, **do−while** or **for**)
- Looping block
- Looping condition

Loops are generally of two types −

Counted Loops in C

If the loop is designed to repeat for a certain number of times, it is a counted loop. In C, the **for** loop is an example of counted loop.

Conditional Loops in C

If the loop is designed to repeat till a condition is true, it is a conditional loop. The **while** and **do−while** constructs help you to form conditional loops.

Looping Statements in C

C programming provides the following types of loops to handle looping requirements −

Sr.No.	Loop Type & Description
1	**while loop** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	**for loop** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**do-while loop** It is more like a while statement, except that it tests the condition at the end of the loop body.
4	**nested loops** You can use one or more loops inside any other **while**, **for** or **do-while** loop.

Each of the above loop types have to be employed depending upon which one is right for the given situation. We shall learn about these loop types in detail in the subsequent chapters.

Loop Control Statements in C

Loop control statements change the execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

C supports the following control statements −

Sr.No.	Control Statement & Description
1	**break statement** Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.
2	**continue statement** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3	**goto statement** Transfers the control to the labeled statement.

The **break** and **continue** statements have contrasting purposes. The **goto** statement acts as a jump statement if it causes the program to go to a later statement. If the **goto** statement redirects the program to an earlier statement, then it forms a loop.

The Infinite Loop in C

A loop becomes an **infinite loop** if a condition never becomes false. An infinite loop is a loop that repeats indefinitely because it has no terminating condition, or the termination condition is never met or the loop is instructed to start over from the beginning.

Although it is possible for a programmer to intentionally use an infinite loop, they are often mistakes made by new programmers.

Example: Infinite Loop in C

The **for** loop is traditionally used for creating an infinite loop. Since none of the three expressions that form the "for" loop are required, you can make an endless loop by leaving the conditional expression empty.

Open Compiler

```
#include <stdio.h>

int main (){

   for( ; ; ){
      printf("This loop will run forever. \n");
   }

   return 0;
}
```

Output

By running this code, you will get an endless loop that will keep printing the same line forever.

```
This loop will run forever.
This loop will run forever.
........
........
This loop will run forever.
```

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the **for(;;)** construct to signify an infinite loop.

Note − You can terminate an infinite loop by pressing the "**Ctrl + C**" keys.