C - Loops

Last Updated : 13 May, 2025

In C programming, there is often a need for repeating the same part of the code multiple times. For example, to print a text three times, we have to use printf() three times as shown in the code:

```c
#include <stdio.h>

int main() {
    printf( "Hello GfG\n");
    printf( "Hello GfG\n");
    printf( "Hello GfG");
    return 0;
}
```

Output

```
Hello GfG
Hello GfG
Hello GfG
```

But if we say to write this 20 times, it will take some time to write statement. Now imagine writing it 100 or 1000 times. Then it becomes a really hectic task to write same statements again and again. To solve such kind of problems, we have loops in programming languages.

```c
#include <stdio.h>

int main() {

    // Loop to print "Hello GfG" 3 times
```

```
10    }
```

Output

```
Hello GfG
Hello GfG
Hello GfG
```

In the above code, we have used a loop to print text 5 times. We could have done it for 100 or even 1000 times in the same number of code lines.

What are loops in C?

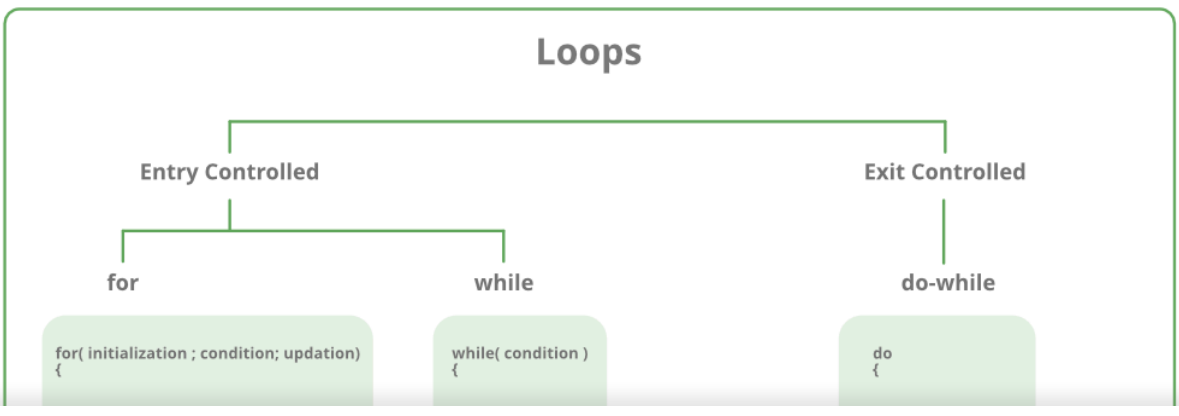
Loops in C programming are used to repeat a block of code until the specified condition is met. It allows programmers to execute a statement or group of statements multiple times without writing the code again and again.

Types of Loops in C

There are 3 looping statements in C:

Table of Content

- for Loop
- while Loop
- do-while Loop



for Loop

for loop in C programming is a repetition control structure that allows programmers to write a loop that will be executed a specific number of times. It enables programmers to perform n number of repetitions in a single line. for loop is **entry-controlled** loop, which means that the condition is checked before the loop's body executes.

Syntax

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

The various parts of the for loop are:

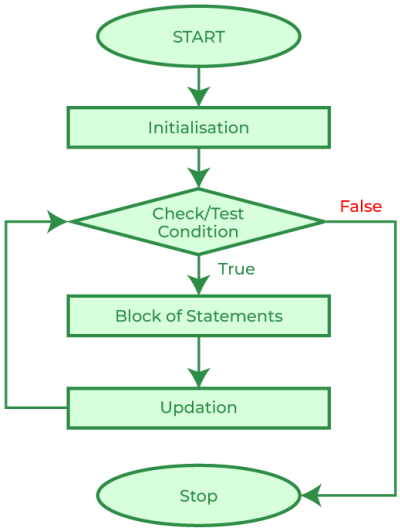
- **Initialization:** Initialize the variable to some initial value.
- **Test Condition:** This specifies the test condition. If the condition evaluates to true, then body of the loop is executed. If evaluated false, loop is terminated.
- **Update Expression:** After the execution loop's body, this expression increments/decrements the loop variable by some value.
- **Body of Loop**: Statements to repeat. Generally enclosed inside **{} braces.**

Example:

```
#include <stdio.h>

int main() {

    // Loop to print numbers from 1 to 5
    for (int i = 0; i < 5; i++) {
        printf( "%d ", i + 1);
    }

    return 0;
}
```

```
1 2 3 4 5
```

Flowchart of for Loop



while Loop

A **while loop** is a block of code to perform repeated task till given condition is true. When condition become false it terminates. while loop is also an **entry-controlled loop** in which the condition is checked before entering the body.

Syntax

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

Only the **condition** is the part of **while loop** syntax, we have to initialize and update loop variable manually.

Example:

```
1  #include <stdio.h>
2  int main() {
3
4      // Initialization expression
```

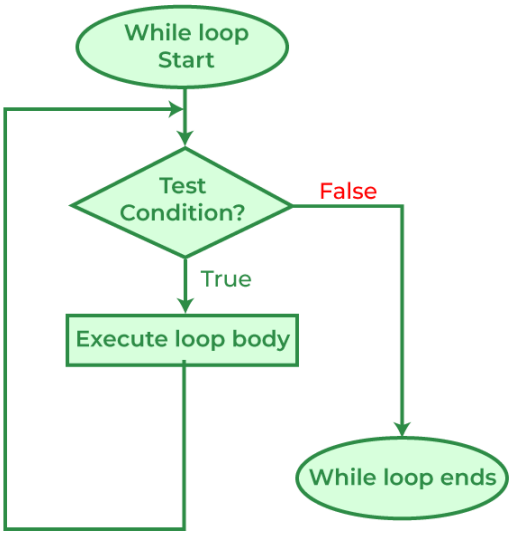
```
    while(i <= 5) {
      printf("%d ", i + 1);

      // update expression
      i++;
    }

    return 0;
  }
```

Output

```
1 2 3 4 5 6
```

Flowchart of while Loop

The below flowchart demonstrates execution flow of the **while loop.**



do-while Loop

The do-while loop is similar to a while loop, but the difference is that do-while loop tests condition after entering the body at the end. do-while loop is **exit-controlled loop**, which means that the condition is checked after executing the loop body. Due to this, the loop body will **execute at least once** irrespective of the test condition.

```
    // Body of the loop
} while (condition);
```

Like while loop, only the condition is the part of **do while loop** syntax, we have to do the initialization and updating of loop variable manually.

Example:

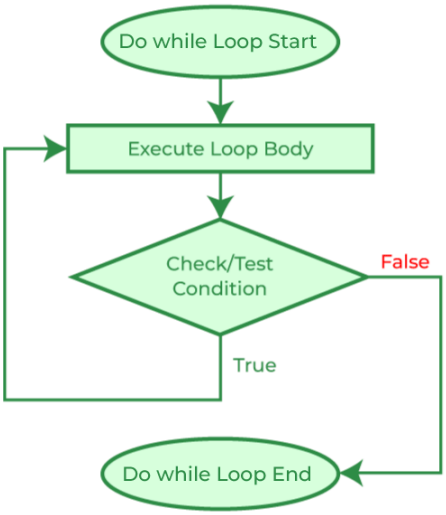
```c
#include <stdio.h>

int main() {

  // Initialization expression
  int i = 0;

  do
  {
    // loop body
    printf( "%d ", i);

    // Update expression
    i++;

    // Condition to check
  }  while (i <= 10);

  return 0;
}
```

Output

```
0 1 2 3 4 5 6 7 8 9 10
```

Flowchart of do-while Loop

The below flowchart demonstrates execution flow of the do while loop.



Infinite Loop

An **infinite loop** is executed when the test expression never becomes false, and the body of the loop is executed repeatedly. A program is stuck in an Infinite loop when the condition is always true. Mostly this is an error that can be resolved by using Loop Control statements.

Using for loop:

```c
#include <stdio.h>

int main () {

    // This is an infinite for loop
    // as the condition expression
    // is blank
    for ( ; ; ) {
        printf("This loop will run forever.");
    }
    return 0;
}
```

Output

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

```
int main()  {
    while (1)
        printf("This loop will run forever.\n");
    return 0;
}
```

Output

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

Using the do-while loop:

```
#include <stdio.h>

int main() {
    do {
        printf("This loop will run forever.");
    } while (1);

    return 0;
}
```

Output

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

Nested Loops

Nesting loops means placing one loop inside another. The inner loop runs fully for each iteration of the outer loop. This technique is helpful when you need to perform multiple iterations within each cycle of a larger loop, like when working with a two-dimensional array or performing tasks that require multiple levels of iteration.

Example:

```c
int main() {
    // Outer loop runs 3 times
    for (int i = 0; i < 3; i++) {

        // Inner loop runs 2 times for each
        // outer loop iteration
        for (int j = 0; j < 2; j++) {
            printf("i = %d, j = %d\n", i, j);
        }
    }
    return 0;
}
```

Output

```
i = 0, j = 0
i = 0, j = 1
i = 1, j = 0
i = 1, j = 1
i = 2, j = 0
i = 2, j = 1
```

Loop Control Statements

Loop control statements in C programming are used to change execution from its normal sequence.

Name	Description
break	The break statement is used to terminate the loop statement.
continue	When encountered, the continue statement skips the remaining body and jumps to the next iteration of the loop.
goto	goto statement transfers the control to the labeled statement.

```c
int main() {
    for (int i = 0; i < 5; i++) {
        if (i == 3) {

            // Exit the loop when i equals 3
            break;
        }
        printf("%d ", i);
    }
    printf("\n");

    for (int i = 0; i < 5; i++) {
        if (i == 3) {

            // Skip the current iteration
            // when i equals 3
            continue;
        }
        printf("%d ", i);
    }
    printf("\n");
    for (int i = 0; i < 5; i++) {
        if (i == 3) {

            // Jump to the skip label when
            // i equals 3
            goto skip;
        }
        printf("%d ", i);
    }

skip:
    printf("\nJumped to the 'skip' label %s",
    "when i equals 3.");

    return 0;
}
```

Output

```
0 1 2 3 4
Jumped to the 'skip' label when i equals 5.
```

Comment More info

Campus Training Program

Next Article

C for Loop

Similar Reads

C for Loop

In C programming, the for loop is used to repeatedly execute a block of code as many times as instructed. It uses a variable (loop variable) whos...

15+ min read

while Loop in C

The while loop in C allows a block of code to be executed repeatedly as long as a given condition remains true. It is often used when we want to...

15+ min read

do...while Loop in C

The do...while loop is a type of loop in C that executes a block of code until the given condition is satisfied. The feature of do while loops is that...

15+ min read

C Variables

In C, variable is a name given to the memory location to easily store data and access it when required. It allows us to use the memory without...

15+ min read

15+ min read

Keywords in C

In C Programming language, there are many rules so to avoid different types of errors. One of such rule is not able to declare variable names wi...

15+ min read

Loop Unrolling

Loop unrolling is a loop transformation technique that helps to optimize the execution time of a program. We basically remove or reduce iteration...

9 min read

Nested Loops in C with Examples

A nested loop means a loop statement inside another loop statement. That is why nested loops are also called "loop inside loops". We can...

15+ min read

Output of C Programs | Set 1

Predict the output of below programs. Question 1c#include<stdio.h> int main() { int n; for(n = 7; n!=0; n--) printf("n = %d", n--); getchar(); return 0...

10 min read

Output of C Programs | Set 6

Predict the output of below programs Question 1 c int main() { unsigned int i=65000; while (i++ != 0); printf("%d",i); return 0; } Output: 1...

15+ min read

Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Preparation Corner

Company-Wise Recruitment Process
Aptitude Preparation
Puzzles
Company-Wise Preparation

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]
Data Analytics Training using Excel, SQL, Python & PowerBI - [LIVE]
Data Science Training Program - [LIVE]
Data Science Course with IBM Certification

Programming Languages

C Programming with Data Structures
C++ Programming Course
Java Programming Course
Python Full Course

Clouds/Devops

DevOps Engineering
AWS Solutions Architect Certification
Salesforce Certified Administrator Course

GATE 2026

GATE CS Rank Booster
GATE DA Rank Booster
GATE CS & IT Course - 2026
GATE DA Course 2026