Programming In C

In programming, loops are used to repeat a block of code until a specified condition is met.

C programming has three types of loops.

1. for loop
2. while loop
3. do...while loop

In the previous tutorial, we learned about `for` loop. In this tutorial, we will learn about `while` and `do..while` loop.

while loop

The syntax of the `while` loop is:

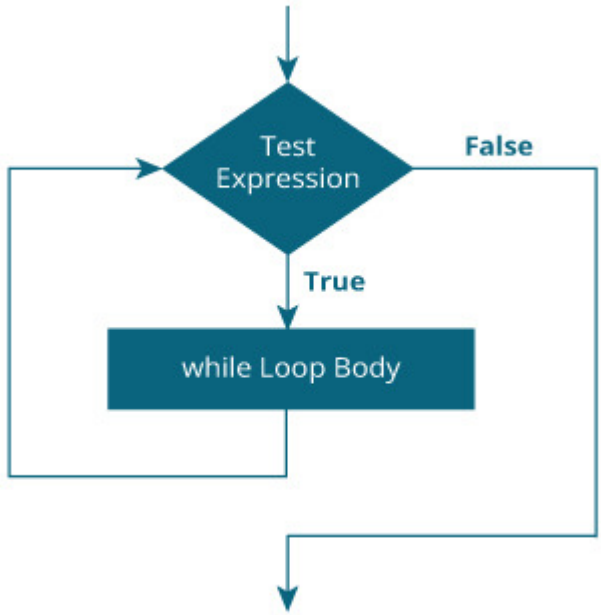
```
while (testExpression)
{
    // statements inside the body of the loop
}
```

How while loop works?

- The `while` loop evaluates the test expression inside the parenthesis `()`.
- If the test expression is true, statements inside the body of `while` loop are executed. Then, the test expression is evaluated again.
- The process goes on until the test expression is evaluated to false.
- If the test expression is false, the loop terminates (ends).

To learn more about test expression (when the test expression is evaluated to true and false), check out relational and logical operators.

Flowchart of while loop



Example 1: while loop

```
// Print numbers from 1 to 5

#include <stdio.h>
int main()
{
    int i = 1;

    while (i <= 5)
    {
        printf("%d\n", i);
        ++i;
    }

    return 0;
}
```

Output

```
1
2
3
4
```

```
5
```

Here, we have initialized `i` to 1.

1. When `i` is 1, the test expression `i <= 5` is true. Hence, the body of the `while` loop is executed. This prints 1 on the screen and the value of `i` is increased to 2.
2. Now, `i` is 2, the test expression `i <= 5` is again true. The body of the `while` loop is executed again. This prints 2 on the screen and the value of `i` is increased to 3.
3. This process goes on until `i` becomes 6. When `i` is 6, the test expression `i <= 5` will be false and the loop terminates.

do...while loop

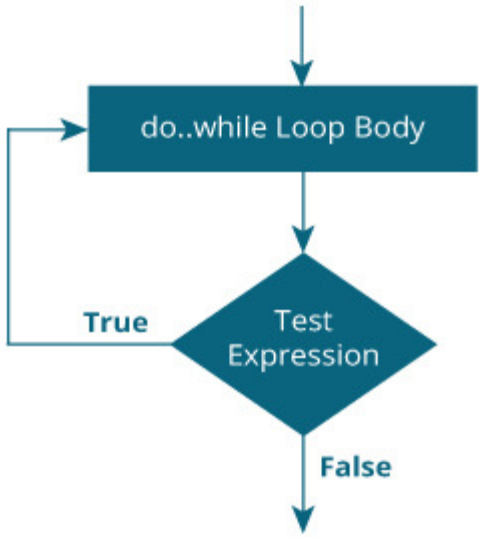
The `do..while` loop is similar to the `while` loop with one important difference. The body of `do...while` loop is executed at least once. Only then, the test expression is evaluated.
The syntax of the `do...while` loop is:

```
do
{
   // statements inside the body of the loop
}
while (testExpression);
```

How do...while loop works?

- The body of do...while loop is executed once. Only then, the test expression is evaluated.
- If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
- This process goes on until the test expression becomes false.
- If the test expression is false, the loop ends.

Flowchart of do...while Loop



Example 2: do...while loop

```
1.     // Program to add numbers until the user enters zero
2.
3.     #include <stdio.h>
4.     int main()
5.     {
6.         double number, sum = 0;
7.
8.         // the body of the loop is executed at least once
9.         do
10.        {
11.            printf("Enter a number: ");
12.            scanf("%lf", &number);
13.            sum += number;
14.        }
15.        while(number != 0.0);
16.
17.        printf("Sum = %.2lf",sum);
18.
19.        return 0;
20.    }
```

Output

```
Enter a number: 1.5
Enter a number: 2.4
Enter a number: -3.4
Enter a number: 4.2
```

```
Enter a number: 0
Sum = 4.70
```

C programming has three types of loops:

1. for loop
2. while loop
3. do...while loop

We will learn about `for` loop in this tutorial. In the next tutorial, we will learn about `while` and `do...while` loop.

for Loop

The syntax of the `for` loop is:

```
for (initializationStatement; testExpression; updateStatement)
{
    // statements inside the body of loop
}
```

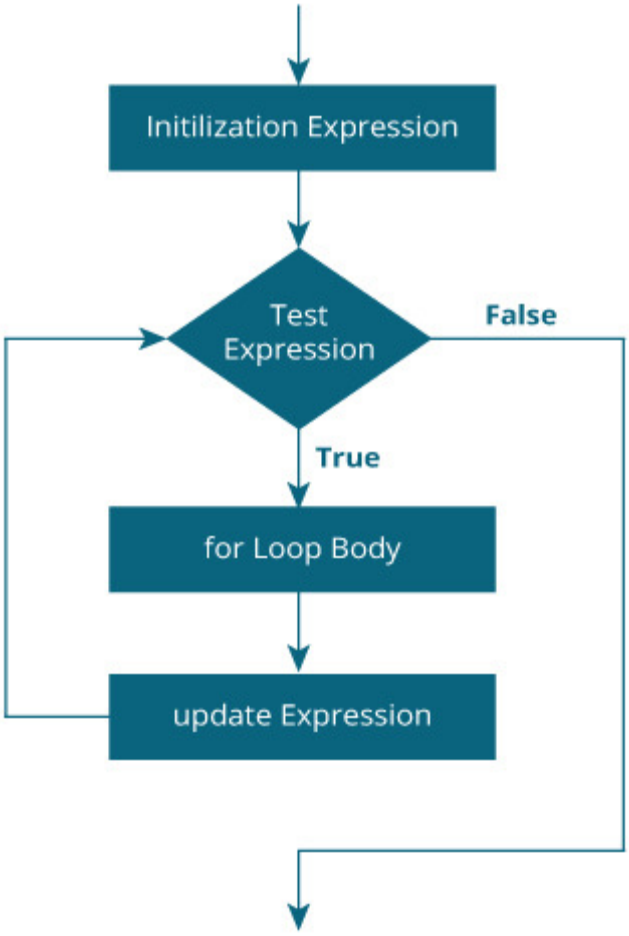
How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the `for` loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of `for` loop are executed, and the update expression is updated.
- Again the test expression is evaluated.

This process goes on until the test expression is false. When the test expression is false, the loop terminates.

To learn more about test expression (when the test expression is evaluated to true and false), check out relational and logical operators.

for loop Flowchart



Example 1: for loop

```
1.      # Print numbers from 1 to 10
2.     #include <stdio.h>
3.
4.     int main() {
5.       int i;
6.
7.       for (i = 1; i < 11; ++i)
8.       {
9.         printf("%d ", i);
10.      }
11.      return 0;
12.    }
```

Output

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

1. `i` is initialized to 1.
2. The test expression `i < 11` is evaluated. Since 1 less than 11 is true, the body of `for` loop is executed. This will print the **1** (value of `i`) on the screen.
3. The update statement `++i` is executed. Now, the value of `i` will be 2. Again, the test expression is evaluated to true, and the body of for loop is executed. This will print **2** (value of `i`) on the screen.
4. Again, the update statement `++i` is executed and the test expression `i < 11` is evaluated. This process goes on until `i` becomes 11.
5. When `i` becomes 11, `i < 11` will be false, and the `for` loop terminates.

Example 2: for loop

```c
// Program to calculate the sum of first n natural numbers
// Positive integers 1,2,3...n are known as natural numbers

#include <stdio.h>
int main()
{
    int num, count, sum = 0;

    printf("Enter a positive integer: ");
    scanf("%d", &num);

    // for loop terminates when num is less than count
    for(count = 1; count <= num; ++count)
    {
        sum += count;
    }

    printf("Sum = %d", sum);

    return 0;
}
```

Output

```
Enter a positive integer: 10
```

```
Sum = 55
```