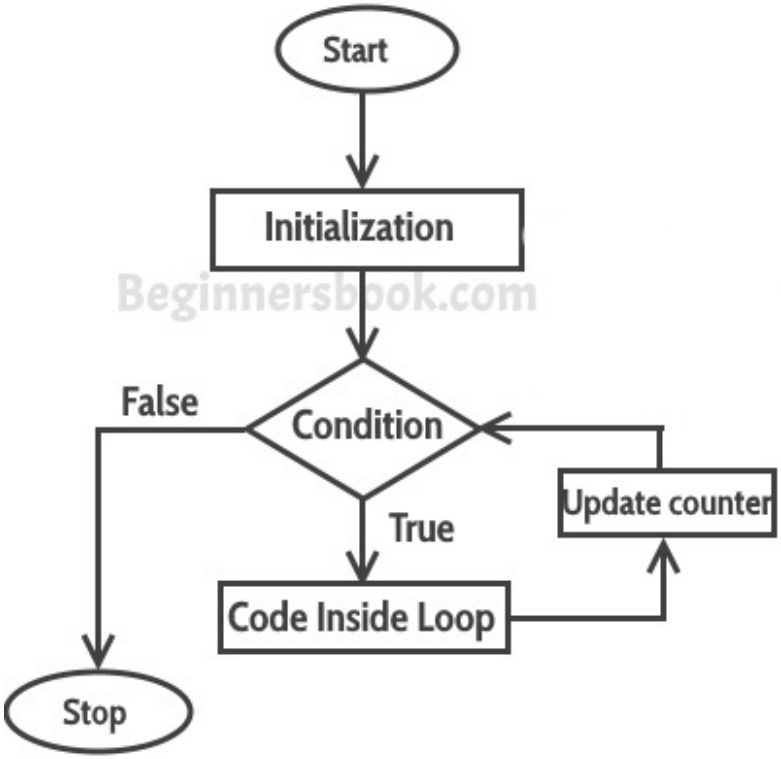
CSE115.12 - Loops



Step 1: First initialization happens and the counter variable gets initialized.
Step 2: In the second step the condition is checked, where the counter variable is tested for the given condition, if the condition returns true then the C statements inside the body of for loop gets executed, if the condition returns false then the for loop gets terminated and the control comes out of the loop.
Step 3: After successful execution of statements inside the body of loop, the counter variable is incremented or decremented, depending on the operation (++ or–).

Example of For loop

```
#include <stdio.h>
int main()
{
   int i;
   for (i=1; i<=3; i++)
   {
       printf("%d\n", i);
   }
   return 0;
}
```

Output:

```
1
2
3
```

Various forms of for loop in C

I am using variable num as the counter in all the following examples –
1) Here instead of num++, I'm using num=num+1 which is same as num++.

```
for (num=10; num<20; num=num+1)
```

2) Initialization part can be skipped from loop as shown below, the counter variable is declared before the loop.

```
int num=10;
for (;num<20;num++)
```

Note: Even though we can skip initialization part but semicolon (;) before condition is must, without which you will get compilation error.
3) Like initialization, you can also skip the increment part as we did below. In this case semicolon (;) is must after condition logic. In this case the increment or decrement part is done inside the loop.

```
for (num=10; num<20; )
{
      //Statements
      num++;
}
```

4) This is also possible. The counter variable is initialized before the loop and incremented inside the loop.

```
int num=10;
for (;num<20;)
{
      //Statements
      num++;
}
```

5) As mentioned above, the counter variable can be decremented as well. In the below example the variable gets decremented each time the loop runs until the condition num>10 returns false.

```
for(num=20; num>10; num--)
```

Nested For Loop in C

Nesting of loop is also possible. Let's take an example to understand this:

```
#include <stdio.h>
int main()
{
   for (int i=0; i<2; i++)
   {
        for (int j=0; j<4; j++)
        {
           printf("%d, %d\n",i ,j);
        }
   }
   return 0;
}
```

Output:

```
0, 0
0, 1
0, 2
0, 3
1, 0
1, 1
1, 2
1, 3
```

In the above example we have a for loop inside another for loop, this is called nesting of loops. One of the example where we use nested for loop is **Two dimensional array**.

Multiple initialization inside for Loop in C

We can have multiple initialization in the for loop as shown below.

```
for (i=1,j=1;i<10 && j<10; i++, j++)
```

What's the difference between above for loop and a simple for loop?

1. It is initializing two variables. Note: both are separated by comma(,).
2. It has two test conditions joined together using AND (&&) logical operator. Note: You cannot use multiple test conditions separated by comma, you must use logical operator such as && or || to join conditions.
3. It has two variables in increment part. **Note:** Should be separated by comma.

Example of for loop with multiple test conditions

```
#include <stdio.h>
int main()
{
   int i,j;
   for (i=1,j=1 ; i<3 || j<5; i++,j++)
   {
        printf("%d, %d\n",i ,j);
   }
   return 0;
}
```

C - while loop in C programming with example

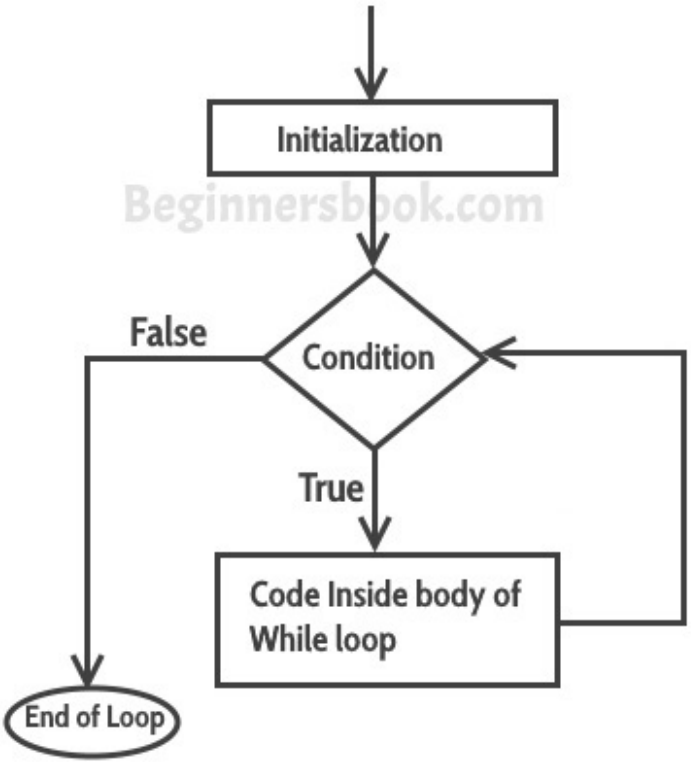
A loop is used for executing a block of statements repeatedly until a given condition returns false. In the previous tutorial we learned **for loop**. In this guide we will learn while loop in C.

C - while loop

Syntax of while loop:

```
while (condition test)
{
      //Statements to be executed repeatedly
      // Increment (++) or Decrement (--) Operation
}
```

Flow Diagram of while loop



Example of while loop

```
#include <stdio.h>
int main()
{
   int count=1;
   while (count <= 4)
   {
        printf("%d ", count);
        count++;
   }
   return 0;
}
```

Output:

```
1 2 3 4
```

step1: The variable count is initialized with value 1 and then it has been tested for the condition.
step2: If the condition returns true then the statements inside the body of while loop are executed else control comes out of the loop.
step3: The value of count is incremented using ++ operator then it has been tested again for the loop condition.

C ess the output of this while loop

```
#include <stdio.h>
int main()
{
     int var=1;
     while (var <=2)
     {
        printf("%d ", var);
     }
}
```

The program is an example of **infinite while loop**. Since the value of the variable var is same (there is no ++ or – operator used on this variable, inside the body of loop) the condition var<=2 will be true forever and the loop would never terminate.

Examples of infinite while loop

Example 1:

```
#include <stdio.h>
int main()
{
     int var = 6;
     while (var >=5)
     {
        printf("%d", var);
        var++;
     }
   return 0;
}
```

Infinite loop: var will always have value >=5 so the loop would never end.

Example 2:

```
#include <stdio.h>
int main()
{
    int var =5;
    while (var <=10)
    {
       printf("%d", var);
       var--;
    }
    return 0;
```

```
}
```

Infinite loop: var value will keep decreasing because of –- operator, hence it will always be <= 10.

Use of Logical operators in while loop

Just like relational operators (<, >, >=, <=, ! =, ==), we can also use logical operators in while loop. The following scenarios are valid :

```
while(num1<=10 && num2<=10)
```

-using AND(&&) operator, which means both the conditions should be true.

```
while(num1<=10||num2<=10)
```

– OR(||) operator, this loop will run until both conditions return false.

```
while(num1!=num2 &&num1 <=num2)
```

– Here we are using two logical operators NOT (!) and AND(&&).

```
while(num1!=10 ||num2>=num1)
```

Example of while loop using logical operator

In this example we are testing multiple conditions using logical operator inside while loop.

```
#include <stdio.h>
int main()
{
   int i=1, j=1;
   while (i <= 4 || j <= 3)
   {
        printf("%d %d\n",i, j);
        i++;
        j++;
   }
   return 0;
}
```

Output:

```
1 1
2 2
3 3
```

C - do while loop in C programming with example

In the previous part, we learned **while loop in C**. A do while loop is similar to while loop with one exception that it executes the statements inside the body of do-while before checking the condition. On the other hand in the while loop, first the condition is checked and then the statements in while loop are executed. So you can say that if a condition is false at the first place then the do while would run once, however the while loop would not run at all.

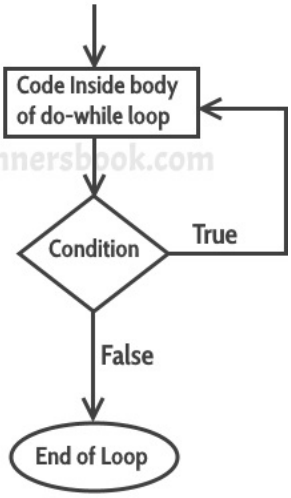
C - do..while loop

Syntax of do-while loop

```
do
{
    //Statements

}while(condition test);
```

Flow diagram of do while loop



Example of do while loop

```
#include <stdio.h>
int main()
{
        int j=0;
        do
        {
                printf("Value of variable j is: %d\n", j);
                j++;
        }while (j<=3);
        return 0;
}
```

Output:

```
Value of variable j is: 0
Value of variable j is: 1
Value of variable j is: 2
Value of variable j is: 3
```

While vs do..while loop in C

Using while loop:

```
#include <stdio.h>
```

```
int main()
{
    int i=0;
    while(i==1)
    {
        printf("while vs do-while");
    }
    printf("Out of loop");
}
```

Output:

```
Out of loop
```

Same example using do-while loop

```
#include <stdio.h>
int main()
{
   int i=0;
   do
   {
        printf("while vs do-while\n");
   }while(i==1);
   printf("Out of loop");
}
```

Output:

```
while vs do-while
Out of loop
```

Explanation: As I mentioned in the beginning of this guide that do-while runs at least once even if the condition is false because the condition is evaluated, after the execution of the body of loop.