Programming Fundamentals Lab

Lab # 08

Loops in C (for, while and do-while)

Instructor: Fariba Laiq

Email: fariba.laiq@nu.edu.pk

Course Code: CL1002

Semester Spring 2023

Department of Computer Science,
National University of Computer and Emerging Sciences
FAST Peshawar Campus

Contents

Control Structures (Repetition)

In programming, sometimes there is a need to perform some operation more than once or (say) n number of times. Loops come into use when we need to repeatedly execute a block of statements.
For example: Suppose we want to print “Hello World” 10 times. This can be done in two ways as shown below:

Manual Method

Manually we have to write printf for the C statement 10 times. Let’s say you have to write it 20 times (it would surely take more time to write 20 statements) now imagine you have to write it 100 times, it would be really hectic to re-write the same statement again and again. So, here loops have their role.

```
// C program to Demonstrate the need of loops
#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;
}
```

Using Loops

In Loop, the statement needs to be written only once and the loop will be executed 10 times as shown below. In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.

```
// C program to Demonstrate the need of loops
#include <stdio.h>
int main()
{
  for (int i=0; i<5; i++){
     printf("Hello World\n");
  }
  return 0;
}
```

Advantage of loops in C

- It provides code reusability.
- Using loops, we do not need to write the same code again and again.
- Using loops, we can traverse over the elements of data structures (array or linked lists).

Types of C Loops

C programming has three types of loops:

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

for Loop

The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance. The syntax of the for loop is:

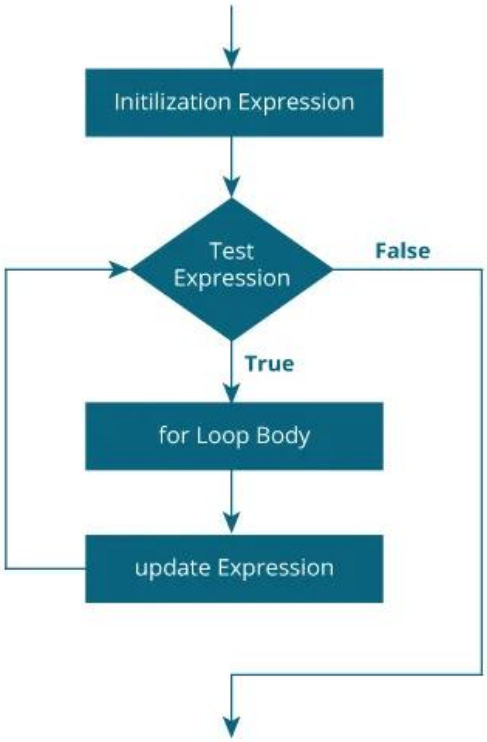
```
for (initializationStatement; testExpression; updateStatement)
{
   // statements inside the body of loop
}
```

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 the 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.

for loop Flowchart



Example 3

```
// Print numbers from 1 to 10
#include <stdio.h>
int main() {
 int i;
 for (i = 1; i < 11; ++i)
 {
   printf("%d ", i);
 }
 return 0;
}
```

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 4

```
// 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: 5
Sum = 15

While Loop

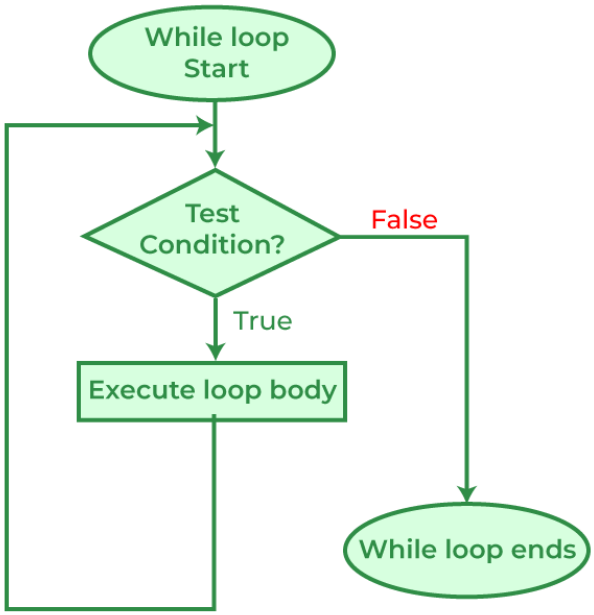
While studying **for loop** we have seen that the number of iterations is ***known beforehand***, i.e. the number of times the loop body is needed to be executed is known to us. while loops are used in situations where **we do not know** the exact number of iterations of the loop **beforehand**. The loop execution is terminated on the basis of the test conditions.
We have already stated that a loop mainly consists of three statements – initialization expression, test expression, and update expression. The syntax of the three loops – For, while, and do while mainly differs in the placement of these three statements.

Syntax:

```
initialization expression;
while (test_expression)
{
   // statements

 update_expression;
}
```

Flow Diagram of while loop:



Example:

```
// C program to Demonstrate
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;
}
```

Output

Hello World

Hello World

Hello World

Hello World

Hello World

do-while loop

In do-while loops also the loop execution is terminated on the basis of test conditions. The main difference between a do-while loop and the while loop is in the do-while loop the condition is tested at the end of the loop body, i.e do-while loop is exit controlled whereas the other two loops are entry-controlled loops.

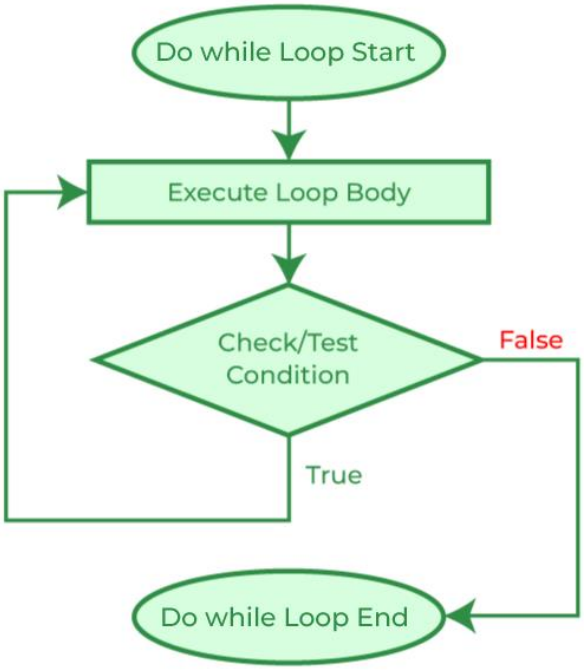
Note: In a do-while loop, the loop body will ***execute at least once*** irrespective of the test condition.

Syntax:
initialization expression;
do
{
  // statements

  update_expression;
} while (**test_expression**);

***Note**: Notice the semi – colon(";")in the end of loop.*

Flow Diagram of the do-while loop:



Example:

```
// C program to Demonstrate 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;
}
```

Output

Hello World

In the above program, the test condition (i<1) evaluates to false. But still, as the loop is an exit – controlled the loop body will execute once.

C break statement

The break is a keyword in C which is used to bring the program control out of the loop. The break statement is used inside loops or switch statement.

Example 1

```
#include<stdio.h>
int main ()
{
  int i;
  for(i = 0; i<10; i++)
  {
     printf("%d ",i);
     if(i == 5)
     break;
  }
  printf("came outside of loop i = %d",i);

}
```

Output

0 1 2 3 4 5 came outside of loop i = 5

C continue statement

The continue statement in C language is used to bring the program control to the beginning of the loop. The continue statement skips some lines of code inside the loop and continues with the next iteration. It is mainly used for a condition so that we can skip some code for a particular condition.

Example 2

```
#include<stdio.h>
int main(){
	int i=1;
	//starting a loop from 1 to 10
	for(i=1;i<=10;i++){
		if(i==5){
			continue;
		}
		printf("%d \n",i);
	}//end of for loop
	return 0;
}
```

Output

1

2

3

4

6

7

8

9

10

References:

https://www.geeksforgeeks.org/cpp-loops/
https://www.programiz.com/c-programming/c-for-loop
https://www.programiz.com/c-programming/c-switch-case-statement