CL1002 – Programming Fundamentals Lab

Lab # 09

Control Structure (Repetition)

Instructor: Muhammad Saood Sarwar

Email: saood.sarwar@nu.edu.pk

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar

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 must write printf for the C statement 10 times. Let’s say you must 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
// 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
// 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 a per-tested loop. It is better to use for loop if the number of iterations are known in advance.

The syntax of the for loop is:

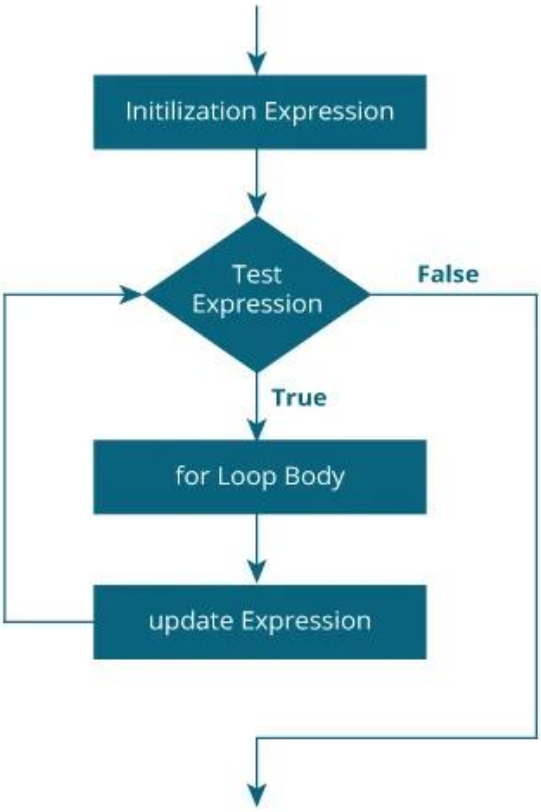
```
for (initializationStatement; testExpression; updateStatement)
{
   // statements inside the body of loop
}
```

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to be false, the for loop is terminated.
- However, if the test expression is evaluated to be 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 1

```c
// 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 2

```
// 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 loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given Boolean condition. The while loop is mostly used in the case where the number of iterations is not known in advance.

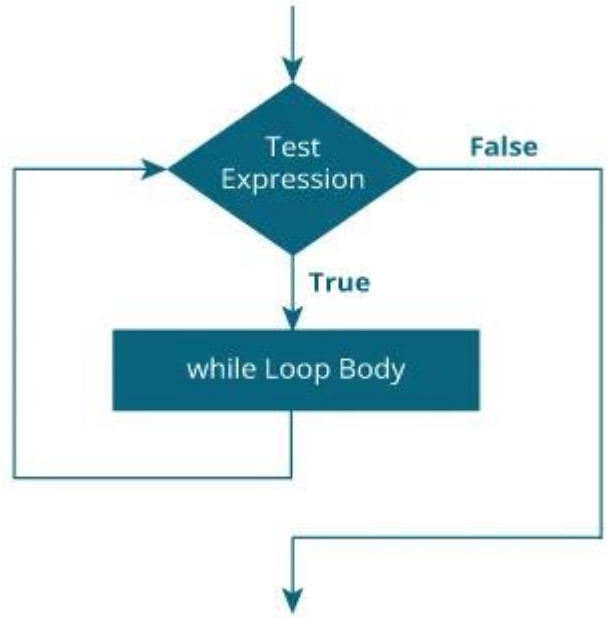
The syntax of the while loop is:

```
while (testExpression) {

 // the body of the loop

}
```

How while loop works?

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

Flowchart of while loop



Example 3

```
// 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 = 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 = 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. Then, the test expression i <= 5 will be false and the loop terminates.

do...while loop

The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements. The do-while loop is mainly used in the case where we need to execute the loop at least once. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.

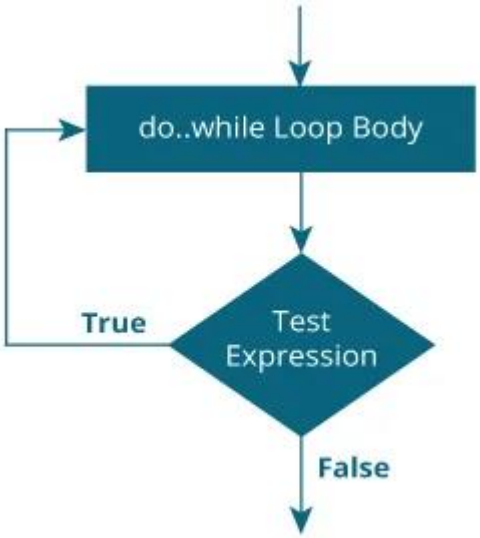
The syntax of the do...while loop is:

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

How do...while loop works?

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

Flowchart of do...while Loop



Example 4

```
// Program to add numbers until the user enters zero

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

 // the body of the loop is executed at least once
 do {
   printf("Enter a number: ");
   scanf("%d", &number);
   sum += number;
 }
 while(number != 0);

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

 return 0;
}
```

Output

```
Enter a number: 4
Enter a number: 2
Enter a number: 1
Enter a number: 0
Sum = 7
```

References:

https://www.geeksforgeeks.org/cpp-loops/

https://www.programiz.com/c-programming/c-for-loop

https://www.programiz.com/c-programming/c-do-while-loops