Faculty of Information Technology, University of Moratuwa
BSc. (Hons) in Information Technology
BSc. (Hons) in Artificial Intelligence
IN1101 - Fundamentals of Programming

Level 1 – Semester 1

Introduction

In this lab session, you'll learn about repetition which is the flow control part in C. Here you will learn for loop properly.

Most algorithms require repetition structures in order to execute one or more instructions a certain number of times.

A loop can be:

– **a definite repetition loop** if we know in advance the number of repetitions, or
– **an indefinite repetition loop** if we do not know in advance the number of repetitions.

The number of repetitions,

– in a definite repetition loop is controlled by a counter (usually an integer variable),
– Whereas the number of repetitions in an indefinite repetition loop is controlled by a sentinel (usually an integer variable).

There exist in C three repetition structures:

– the for structure,
– the while structure and
– the do/while structure. **(while and do-while will be done in future lab sessions)**

for Repetition Structure

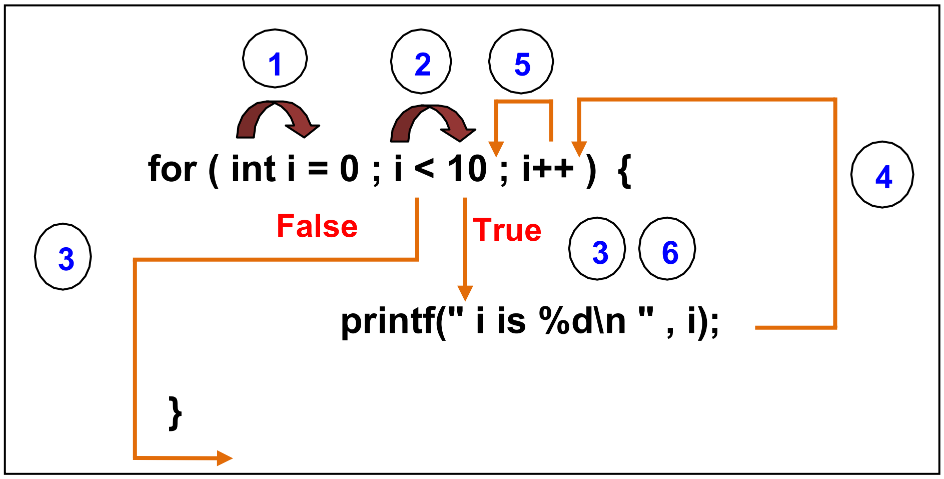
Operation of **for structure** in a program:

```
          A          B    E          D
for (initial value; condition; increment/decrement) {
        command 1;
        command 2;     D

}   C
```

A Program execution flows first into initial value which is evaluated once to initialize the loop counter.
B condition is then evaluated; recall that it is a logical expression containing the loop counter;
C if condition evaluates to false, then the commands between the {} are skipped and the program continues to execute after the loop;
D if condition evaluates to true, then the commands between the {} are executed and increment/decrement is performed to modify the value of the counter;
E back to step B.

Form of execution of for loop


1
2
5
4
for (int i = 0 ; i < 10 ; i++) {
False
True
3
3
6
printf(" i is %d\n " , i);
}

1 Declaration & initialization of i

2 Check whether i<10

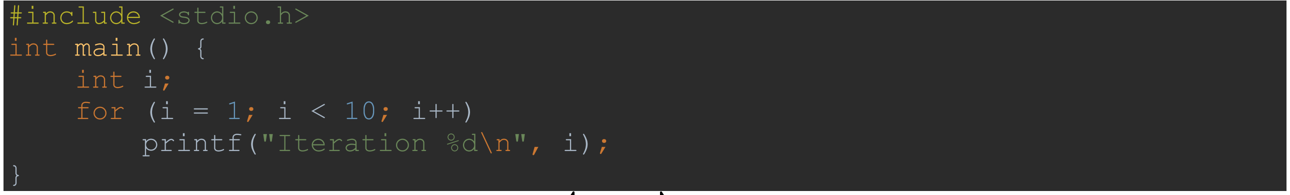
3 True; Execute the for loop body
False; come out of loop

4 i is iterated by one

5 check whether i<10

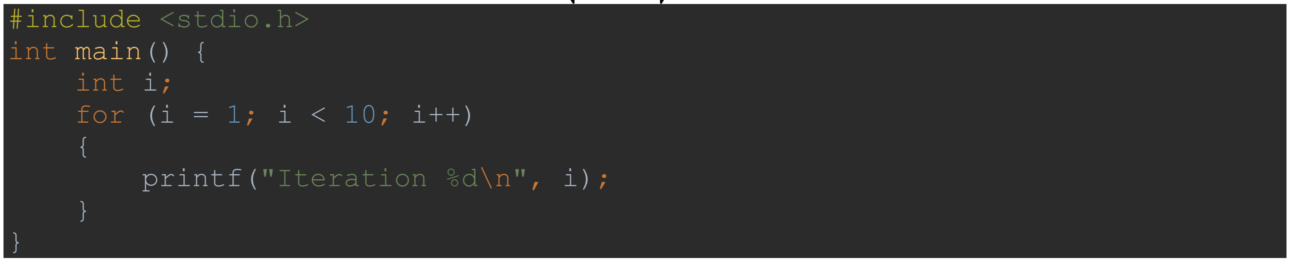
6 Similar as step 2

Example of a repetition loop using **for structure:**



```
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i < 10; i++)
        printf("Iteration %d\n", i);
}
```

OR



```
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i < 10; i++)
    {
        printf("Iteration %d\n", i);
    }
}
```

Nested Loops

C allows for the nesting of loops. C's ability to nest loops makes it possible to iterate over statements within loops. Let's have a look at an example of a C program that uses nested loops.

Nested for loop format:

```
#include <stdio.h>
int main() {
    for (initialization; condition; update) // outer loop
    {
        for(initialization; condition; update) // inner loop
        {
            // inner loop statements.
        }
        // outer loop statements.
    }
}
```

Example:

```
#include <stdio.h>
int main() {
    int x, y;
    for (x = 1; x < 3; x++)
    {
        for (y = 1; y < 4; y++)
        {
            printf("sum = %d\n", x + y);
        }
    }
}
```

There is no limit to the number of loops that can be defined; in fact, a loop can be defined inside another loop. There is an nth-level definition of nesting. One loop type can be defined inside another; for instance, a "while" loop can be defined inside a "for" loop and vice versa.

Exercises

1. Using a proper loop statement (For) Try to get the following outputs using C.

    a. Display the numbers from 1 to 10 line by line
    b. Display the numbers from 10 to 1 line by line
    c. Display the sequence 1, 3, 5, 7, 9 line by line
    d. Display the sequence 1, 4, 7, 10, 13, 16… up to 52 line by line
    e. Display the sequences 2, 3, 5, 9, 17, … up to 33 line by line
    f. Display all the odd numbers between 1 to 50.
    g. Display all the even numbers between 1 to 50.
    h. Display all the numbers that are multiples of 3 between 1 to 50.

2. Using a proper loop statement. Try to get the following outputs using C.

    a. Summation of all the numbers from 1 to 10
    b. Multiplication of all the numbers from 1 to 10
    c. Number of even numbers from 1 to 50
    d. Number of odd numbers from 1 to 50

3. Write a C program to print Fibonacci series i.e.0 1 1 2 3 5 8….. up to n^{th} term.

4. Write a C program to find the factorial of a given positive number.

5. Write a C program to input an integer and check whether it is Prime or not.

6. Write a program to find the sum of the series: $1/1^2+1/2^2+1/3^2+.....1/n^2$.

7. Write a C program to print the first 10 terms of the following series using FOR loop. 1, 5, 9, 13, ….

8. Write C programs to print the following patterns.

```
a.  1              b.  *              c.  1              d.      *
    2 2                * *                1 2                   ***
    3 3 3              * * *              1 2 3                *****
    4 4  4 4           * * * *            1 2 3 4             *******
    5 5 5 5 5          * * * * *          1 2 3 4 5          *********
```

9. Write a C program that calculates the mathematical constant 'e' using the following formula.

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots.$$

Calculate the formula e = 1 + 1/1! + 1/2! + 1/3! +…. up to n=5.

10. Write a C program that calculates and prints the sum of the even integers from 2 to 30.

11. Write a C program that calculates and prints the product of the odd integers from 1 to 15.

12. Write a C program to get the sum of odd integers between 1 and 99 using a for loop.

13. Write a C program that prints the following pattern

```
******
*****
****
***
**
*
```

14. Write a C program to print the following pattern using nested loops. You must prompt the user for entering the number of rows.

Hint: if the number of rows=7 then the output should like follows,

```
                              1
                         1    2    1
                    1    2    4    2    1
               1    2    4    8    4    2    1
          1    2    4    8    16   8    4    2    1
     1    2    4    8    16   32   16   8    4    2    1
1    2    4    8    16   32   64   32   16   8    4    2    1
```