Programming and Problem Solving through C Language
O Level / A Level

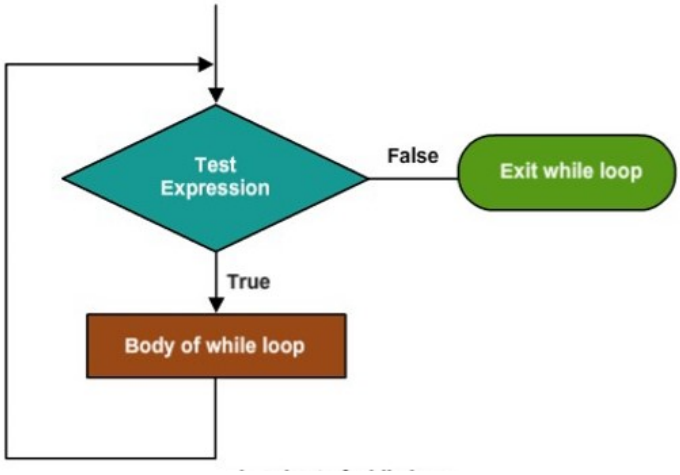
Chapter -4 : Conditional Statements and Loops

Loops

- A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages.
- C programming language provides the following types of loop to handle looping requirements.
    - while loop.
    - for loop.
    - do…while loop.

While loop - Statement

- The 'while' statement, also called the while loop, executes a block of statements as long as a specified condition is true.
- Condition is any C expression, and statement is a single or compound C statement.
- Execution of while statement :
    1. The expression condition is evaluated.
    2. If condition evaluates to false (that is, zero), the while statement terminates, and execution passes to the first statement following the while statement.
    3. If condition evaluates to true (that is, nonzero), the C statement(s) in statement are executed.
    4. Execution returns to **step 1**.



while loop: Syntax

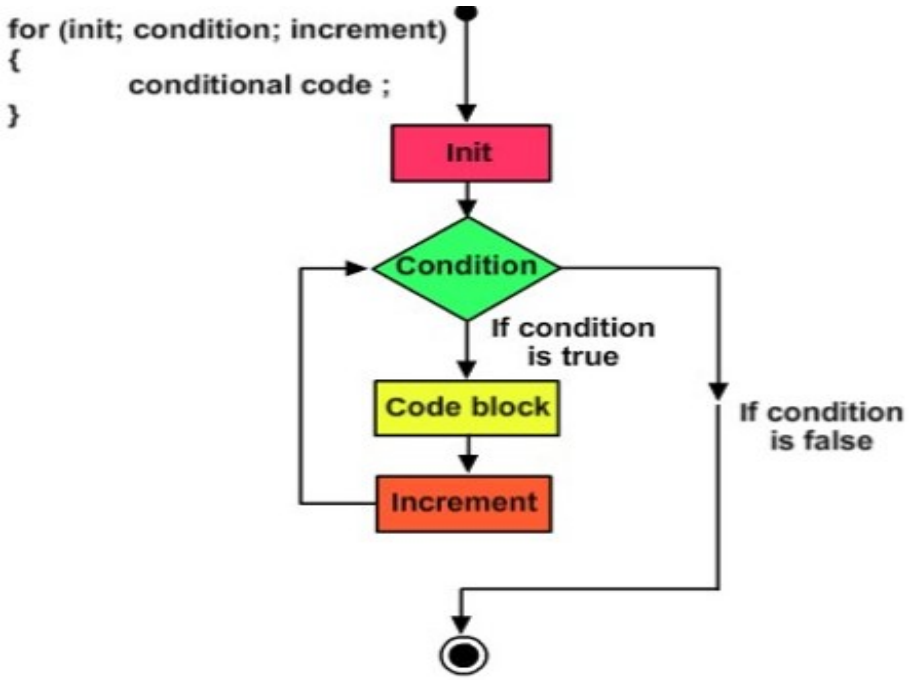
```
while (condition test)
{
    // C- statements, which requires repetition.
    // Increment (++) or Decrement (--) Operation.
}
```

while loop: Example

```
..
main()
{
   int count=1;
   while (count <=4)
   {
      printf("%d ", count);
      count++;
   }
}
```

For loop

- The 'for' statement is a C programming construct that executes a block of one or more statements a certain number of times.
- A for statement has the following structure:



- Initial, condition, and increment are all C expressions, and Code Block statement is a single or compound C Statement.

- All the 3 parts of for Loop is optional. If no condition is specified , it is treated as True.
- Execution of 'for' statement :-
    1. The expression **initial** is evaluated. **Initial** is an assignment statement that sets a variable to a particular value.
    2. The expression **condition** is evaluated; **condition** is a relational expression.
    3. If **condition** evaluates to false (that is, as zero), the 'for' statement terminates, and execution passes to the first statement following the **for** statement.
    4. If **condition** evaluates to true (that is, as nonzero), the C statement(s) in statement are executed.
    5. The expression **increment** is evaluated, and execution returns to the expression condition [i.e. Step-2].

for loop: Syntax

```
for(initialization; condition test; increment or decrement)
{
    //Code – C statements needs to be repeated
}
```

for loop: Example

// **Program -1 to print the - Hello, World- 3 times**

```
#include<stdio.h>
void main()
{
   int i;

   for(i=1 ; i<=3 ; i++)
     printf("hello, World");
}
```

// **Program -2 to print the number- 1,2,3,----10**

```
#include<stdio.h>
void main()
{
   int i;

   for(i=1 ; i<=3 ; i++)
     printf("%d " , i);
}
```

Assignment

1. Write the program to display the first 10 terms of the following series :
    a. 1 , 3, 5,……
    b. 2 , 4 , 6 ……
    c. 1 , 4 , 9 , 16…….
    d. 1.5 , 3.0 , 4.5 , 6.0 ……..
    e. -5 , -10 , -15, -20 …….
2. Write a program to input any 20 numbers (including positive and negative). Perform the following tasks
    a. Count and display the positive numbers
    b. Count and display the negative numbers
    c. Display the sum of positive numbers
    d. Display the sum of negative numbers
3. Write a program to calculate and display the sum of all odd numbers and even numbers between a range of numbers from m to n where m < n. Input m and n.
4. Write a program to print the 10 multiples of any entered number.
5. Write a program to display the sum of 10 natural numbers.
6. Write a program to calculate and display the factorial of a entered number.