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

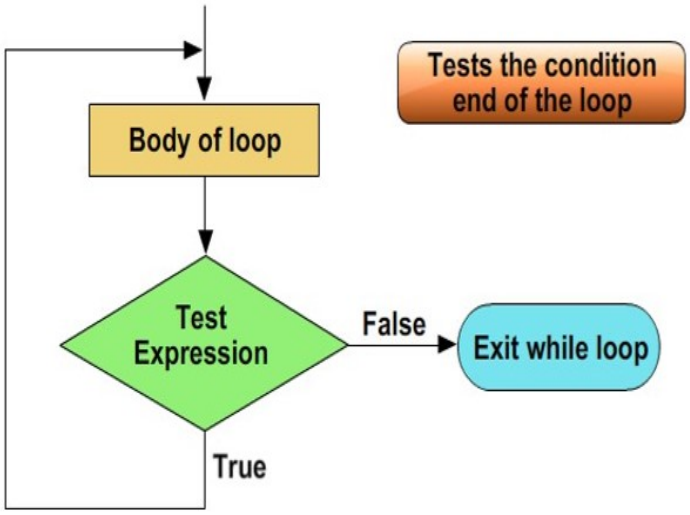
Chapter -4 : Conditional Statements and Loops

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.

The do...while Loop

- the 'do...while' loop, which executes a block of statements as long as a specified condition is true.
- The do...while loop tests the condition at the end of the loop rather than at the beginning, as is done by the 'for' loop and the while loop. It is also known **as post test loop**.
- The structure of the do...while loop is as follows:



- Condition is any C expression, and statement is a single or compound C statement.
- Execution of do...while statement :
    - Step 1: The statements in statement are executed.
    - Step 2: Condition is evaluated.
    - Step 3: If it's true, execution returns to step 1.
    - Step 4: If it's false, the loop terminates.
- The statements associated with a do...while loop are always executed at least once.
- This is because the test condition is evaluated at the end, instead of beginning, of the loop.

do...while loop: Syntax

```
..
do
{
   //C- statements

}while(condition test);
```

do...while loop: Example

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

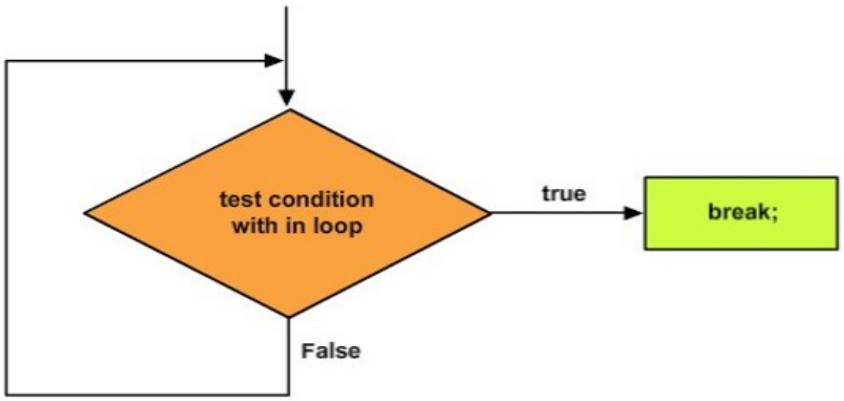
```
#include <stdio.h>

void main () {

  /* local variable definition */
  int a = 10;

  /* do loop execution */
  do {
    printf("value of a: %d\n", a);
    a = a + 1;
  } while( a < 20 );

}
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Break Statement

- The break statement can be placed only in the body of a 'for' loop, 'while' loop, or 'do...while' loop.
- When a break statement is **encountered**, execution exits the loop.
- This can also be used in switch case control structure. Whenever it is encountered in switch-case block, the control comes out of the switch-case



Example – Use of break in a while loop

```
#include <stdio.h>
void main()
{
    int num =0;
    while(num<=100)
    {
      printf("value of variable num is: %d\n", num);
      if (num==2)
      {
         break;
      }
      num++;
    }
    printf("Out of while-loop");

}
```

Output:

```
value of variable num is: 0
value of variable num is: 1
value of variable num is: 2
Out of while-loop
```

Example – Use of break in a for loop

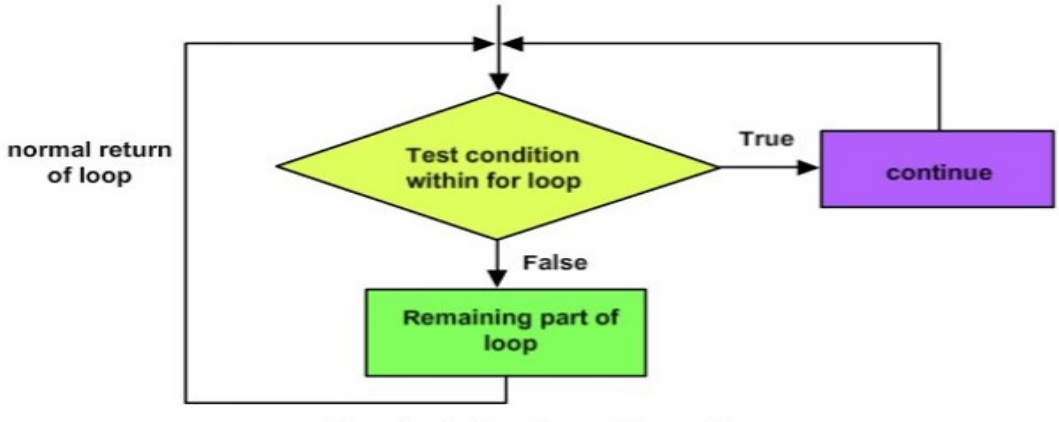
```
#include <stdio.h>
void main()
{
    int var;
    for (var =100; var>=10; var --)
    {
        printf("var: %d\n", var);
        if (var==99)
        {
           break;
        }
    }
   printf("Out of for-loop");
}
```

Output:

```
var: 100
var: 99
Out of for-loop
```

Continue Statement

- Like the break statement, the continue statement can be placed only in the body of a 'for' loop, a while loop, or a 'do...while' loop.
- When a continue statement executes, the next iteration of the enclosing loop begins immediately.
- The statements between the continue statement and the end of the loop aren′t executed.
- Continue is used inside a loop.
- It causes the control of a program to skip the rest of the current iteration of a loop and start the next iteration.



Flowchart of continue statement

Example: continue statement inside for loop

```
#include <stdio.h>
void main()
{
  for (int j=0; j<=8; j++)
  {
    if (j==4)
     {
     /* The continue statement is encountered when
      * the value of j is equal to 4.
      */
     continue;
     }

     /* This print statement would not execute for the
  * loop iteration where j ==4  because in that case
  * this statement would be skipped.
  */
     printf("%d ", j);
  }

}
```

Output:

```
0 1 2 3 5 6 7 8
```

Value 4 is missing in the output, why?

When the value of variable j is 4, the program encountered a continue statement, which makes the control to jump at the beginning of the for loop for next iteration, skipping the statements for current iteration

Example: Use of continue in While loop

```
#include <stdio.h>
void  main()
{
   int counter=10;
   while (counter >=0)
   {
   if (counter==7)
   {
       counter--;
       continue;
   }
   printf("%d  ", counter);
   counter--;
   }

}
```

Output:

```
10 9 8 6 5 4 3 2 1 0
```

The print statement is skipped when counter value was 7.