Introduction to C Programming

Looping Constructs

Computers are very good at performing repetitive tasks very quickly. In this section we will learn how to make computer repeat actions either a specified number of times or until some stopping condition is met.

Iterative statements are used to repeat the execution of a list of statements, depending on the value of an integer expression. C language supports three types of iterative statements also known as **looping statements**.

Loop:-it is a block of statement that performs set of instructions. In loops repeating particular portion of the program either a specified number of time or until a particular no of condition is being satisfied. There are three types of loops in C language:-

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

while loop (Condition-Controlled Loops)

Syntax:-

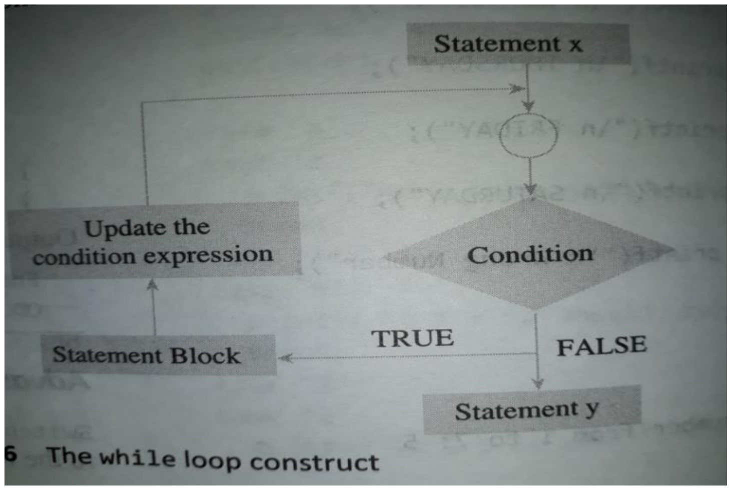
```
while( condition )
    body;
```

where the body can be either a single statement or a block of statements within { curly braces }.

Or

```
initialization;

while (condition)
{
Statement 1;
Statement 2;
increment/decrement;
}
```



6 The while loop construct

Or

```
Statement x;
initialization;
while (test condition)
{
Statement 1;
Statement 2;
increment/decrement;
}
Statement y;
```

The test condition may be any expression .when we want to do something a fixed number of times but not known about the number of iteration, in a program then while loop is used. Here first condition is checked if, it is true body of the loop is executed else, If condition is false control will be come out of loop.

Example:-

```
/* write a program to print 5 times "Welcome to C" */
#include<stdio.h>
void main()
{                        /* main begins */
int p;
p=1;                     /* initialization */
while (p<=5)             //test the condition
{                        /*loop begins*/
  printf("Welcome to C\n");
  p=p+1;               /* increment , condition updated */
 }                       /*end of loop */
}                      /* end of main */
```

```
Output: Welcome to C
Welcome to C
Welcome to C
Welcome to C
Welcome to C
```

```
/* write a program to print 5 times "Welcome to C" */
#include<stdio.h>
void main()
{                        /* main begins */

        int i = 0;  //declaration and initialisation
        while( i < 5 )   //tested condition
            printf( "i = %d\n", i++ );   //body of while loop
        printf( "After loop, i = %d\n", i );  //statement outside loop
}// end of main
```

```
Output: i=0
            i=1
i=2
i=3
i=4
After loop i = 5
```

So as long as condition remains true statements within the body of while loop will get executed repeatedly.

<u>do while loop</u>

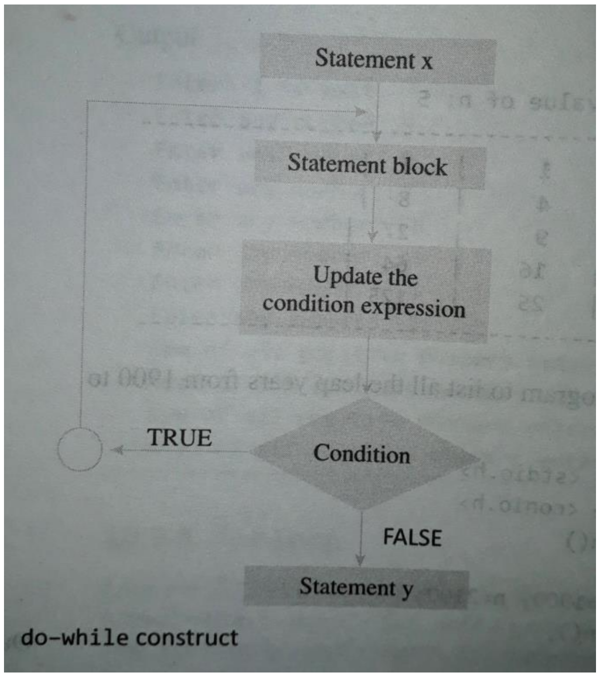
This (do while loop) statement is also used for looping. The body of this loop may contain single statement or block of statement. It is exactly like while loops, except that the test is performed at the end of the loop rather than the beginning.

- This guarantees that the loop will be performed at least once, which is useful for checking user input among other things (see example below.)

The syntax for writing this statement is:-

Syntax:-

```
initialization;
do {
            body;
        increment/decrement;
        } while( condition );
```



do-while construct

Or

```
Statement x;
initialization;
do
{
Statement;
Increment/decrement;
}while(condition);
Statement y;
```

Example:-

```
#include<stdio.h>
void main()
{                    // main begins
int X;
X=4;  //initialization
do
{          //loop begins
  Printf("%d",X);
   X=X+1;            //condition updated
}whie(X<=10);  //test the condition, end of loop
printf(" ");    //statement outside loop
}   //end of main
Output: 4 5 6 7 8 9 10
```

Here firstly statement inside body is executed then condition is checked. If the condition is true again body of loop is executed and this process continues until the condition becomes false. Unlike while loop semicolon is placed at the end of while.

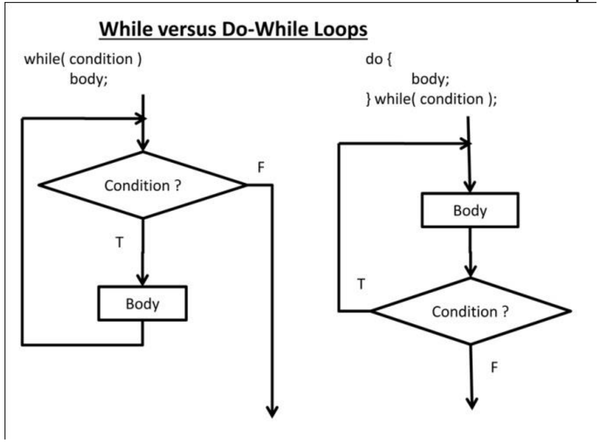
```
//Write a program to calculate the average of first n numbers
#include<stdio.h>
int main()
{
  int n, i=1,sum=0;
  float avg=0.0;

  printf("\nEnter the value of n:");
  scanf("%d",&n);
 do
 {
   sum = sum+i;
   i++;
}while(i<=n);
avg= (float) sum/n;
printf("\nThe sum of first %d numbers=%d",n,sum);
printf("\nThe average of first %d numbers= %f",n,avg);
return 0;
}
```

Output

```
  Enter the value of n:18
The sum of first 18 numbers = 171
The average of first 18 numbers=9.00
```

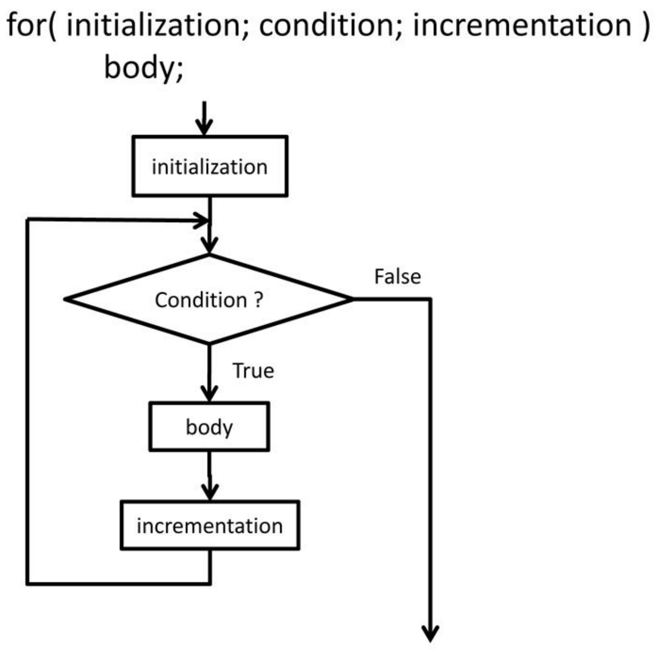
There is a difference between while and do while loop, while loop test the condition before executing any of the statement of loop. Whereas do while loop test condition after having executed the statement at least one within the loop. If initial condition is false while loop would not executed it's statement on other hand do while loop executed it's statement at least once even If condition fails for first time. It means do while loop always executes at least once.



- Both while loops and do-while loops are *condition-controlled*, meaning that they continue to loop until some condition is met.
- Both while and do-while loops alternate between performing actions and testing for the stopping condition.
- While loops check for the stopping condition first, and may not execute the body of the loop at all if the condition is initially false.

<u>for loop</u>

- for-loops are *counter-controlled*, meaning that they are normally used whenever the number of iterations is known in advance.



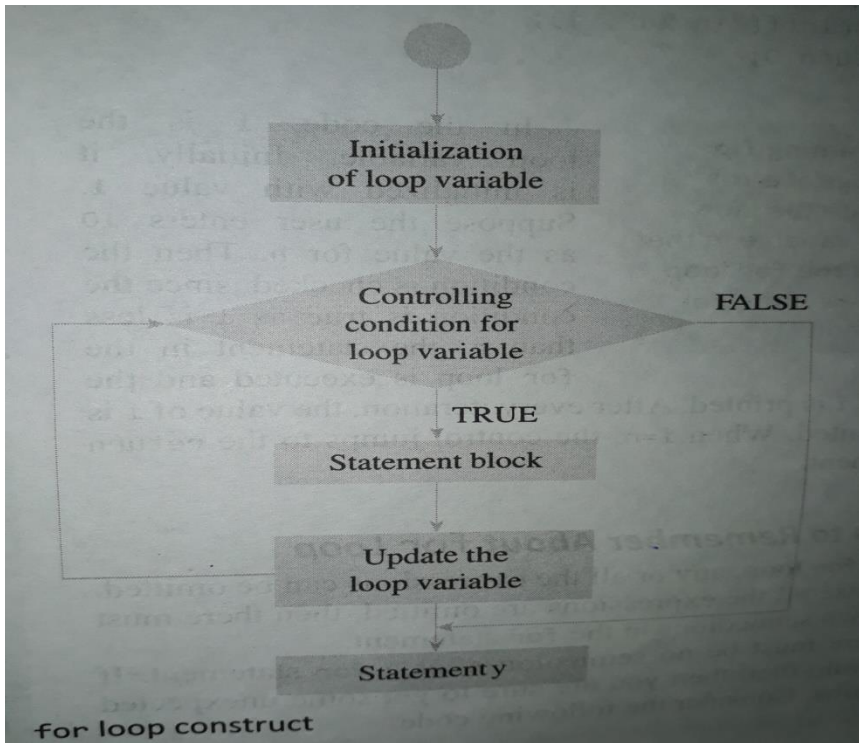
  - The initialization step occurs one time only, before the loop begins.

- The condition is tested at the beginning of each iteration of the loop.
    - If the condition is true (non-zero), then the body of the loop is executed next.
    - If the condition is false (zero), then the body is not executed, and execution continues with the code following the loop.
- The incrementation happens AFTER the execution of the body, and only when the body is executed.

In a program, for loop is generally used when number of iteration are known in advance. The body of the loop can be single statement or multiple statements. Its syntax for writing is:
Syntax:-

```
for(initialization; condition; increment/decrement/update)
{
  Statement block;
}
```

Or
```
for(exp1;exp2;exp3)
{                   //begin of for loop
Statement;
}   //end of for loop
```



Or
```
Statement x;
for(initialized counter; test counter; update counter)
{
Statement; ←——Body of for loop
}
Statement y;
```

Here exp1 is an initialization expression, exp2 is test expression or condition and exp3 is an update expression. Expression 1 is executed only once when loop started and used to initialize the loop variables. Condition expression generally uses relational and logical operators. And updation part executed only when after body of the loop is executed.

Example:-

```
//Write a program to print sum of first 10 numbers
void main()
{
  int i,sum;
```

```
   for(i=0, sum=0; i<10;i++)
    sum+=i;

  printf("%d",sum);
}
```

Output 45

NOTE

```
#include <stdio.h>
Void main()
{
   for(;;);---------------------------syntactically correct and leads to infinite loop
 printf("****");
}
```

Nesting of loop

When a loop written inside the body of another loop then, it is known as nesting of loop. Any type of loop can be nested in any type such as while, do while, for.

Example: Nesting of for loop

```
//write a program to print the following pattern.
1
12
123
1234
12345
#include<stdio.h>
void main()
{      //main begins
  int i,j;
  for(i=1;i<=5;i++)
  {                    //outer for loop begins
     printf("\n");
     for(j=1;j<=i; j++)
         printf("%d", j);   // body of inner for loop
  }//end of outer for loop
} //end of main
```

Example: Nesting of do while and for loop
//Write a program to print table of 2 and 3

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

   i=2; prod=1;

do
{  //begin of do-while loop
  printf("\n Table of %d:",i);
   for(j=1;j<=10;j++)
    {    //begin of for loop
      prod=i*j;
      printf("\n%d X %d=%d",i,j,prod);
    } //end of for loop
   i++;
  printf("\n*****************************\n");
  }while(i<=3);  //end of do-while loop
} //end of main
```

Empty Loops

- A common error is to place an extra semi-colon at the end of the while or for statement, producing an empty loop body between the closing parenthesis and the semi-colon, such as:

```
int i;
for( i = 0; i < 5; i++ ); // Error on this line causes empty loop
    printf( "i = %d\n", i );  // This line is AFTER the loop, not inside it.
```

or

```
int i = 0;
while( i < 5 );  // Error - empty loop on this line
    printf( "i = %d\n", i++ ); // Again, this line is AFTER the loop.
```

- In the case of the while loop shown above, it is not only empty but also infinite.

Reference:-

1. LECTURE NOTE On PROGRAMMING IN “C” http://www.vssut.ac.in/lecture_notes/lecture1424354156.pdf
2. https://beginnersbook.com/2014/01/c-for-loop/
3. https://www.studytonight.com/c/loops-in-c.php
4. https://www.tutorialspoint.com/cprogramming/c_loops.htm
5. Book on Computer Fundamentals and Programming in C by Reema Thareja