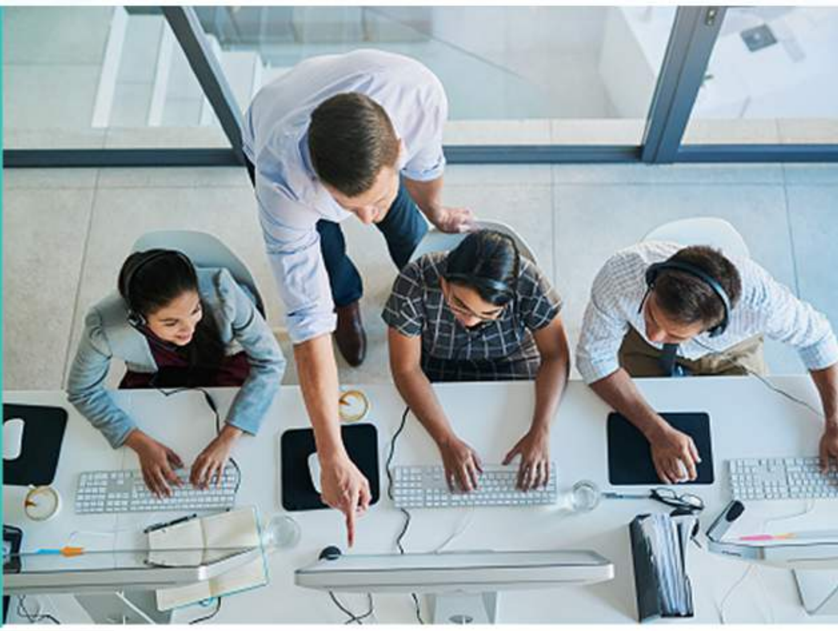
Andhra Pradesh State Skill Development Corporation



Programming in C

Loops

Loops

The for statement lets you repeat a statement or compound statement a specified number of times. The body of a for statement is executed zero or more times until an optional condition becomes false. You can use optional expressions within the for statement to initialize and change values during the for statement's execution.
A loop is a construction that allows you to execute a block of code multiple times. It consists of two parts - a condition and a body.

Loop condition

Every loop has a condition. This is the part that controls if the loop should continue or stop. All loops in C continue their iterations if the condition is true. In C, the condition is any valid value or expression that could be evaluated to a value. A value of 0 or '\0' (null) is considered as "false" and everything else is "true"

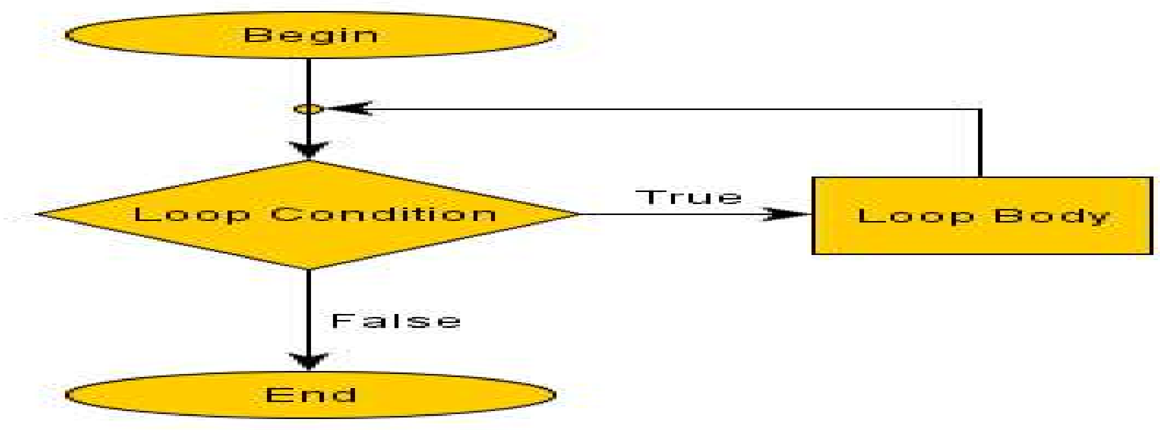
Body

The body is one or more valid C statements. If it consists of more than one statement, the body must be enclosed in curly brackets { } .

How loops in C work:

The loop starts by entering the condition. If it is "true" the body will be executed. Then the execution repeats the check of the condition and then the body. It looks like this:
begin -> condition(true)->body-> condition(true)->body ... condition(false)->end.

One check of the condition, followed by one execution of the body is one iteration. Here is a flowchart to visualize the idea:



Types of loops:

1.for loop
2.Nested for
3.while loop
4.Nested While
5.do...while loop

For loop:

A loop is used for executing a block of statements repeatedly until a given condition returns false.

Syntax of the For loop:

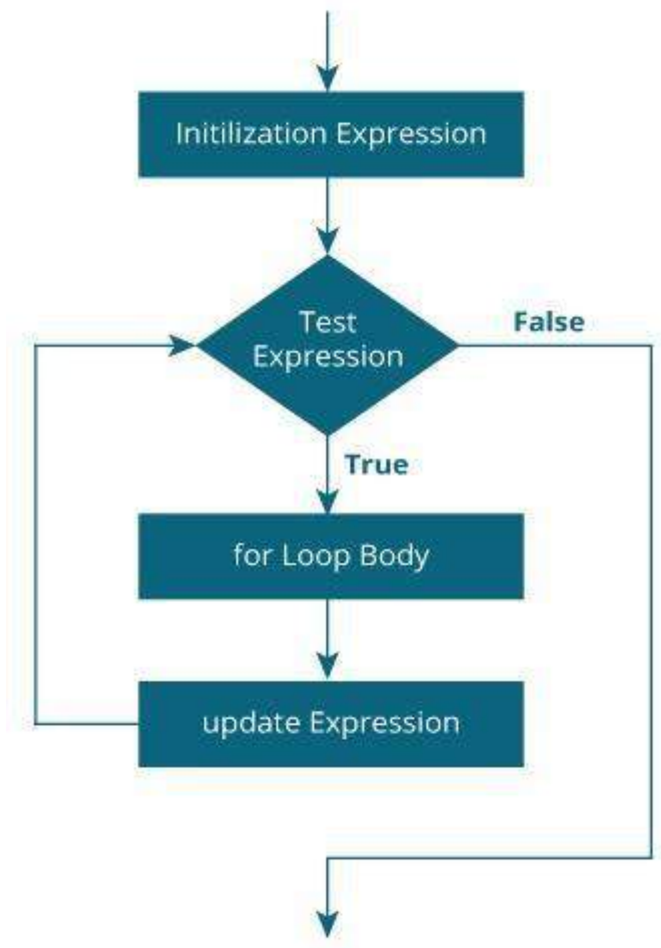
```
for (initialization; condition test; increment or decrement)
{
    //Statements to be executed repeatedly
}
```

How for loop works:

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of 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 of for loop:

```
        //  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
```

Explanation:

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.

Nested for loop :

Nesting of loops is the feature in C that allows the looping of statements inside another loop. ... The nesting level can be defined at n times. You can define any type of loop inside another loop; for example, you can define 'while' loop inside a 'for' loop.

Syntax for Nested For loop:

```
for ( initialization; condition; increment )
 {

   for ( initialization; condition; increment )
 {
     // statement of inside loop
   }

   // statement of outer loop
}
```

Nested for loop Flowchart:

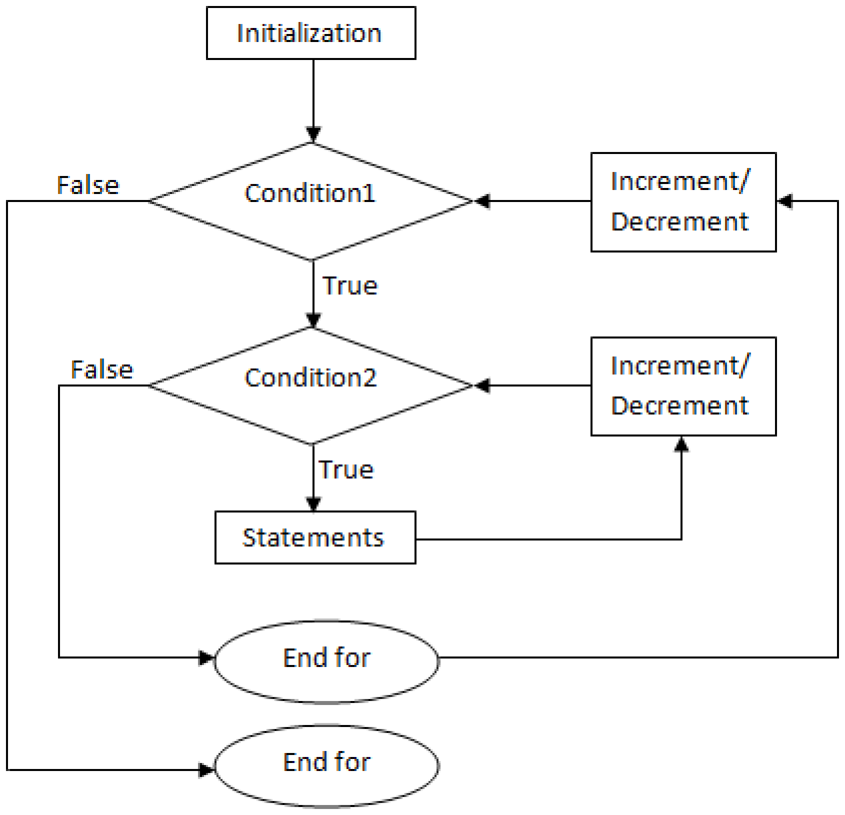
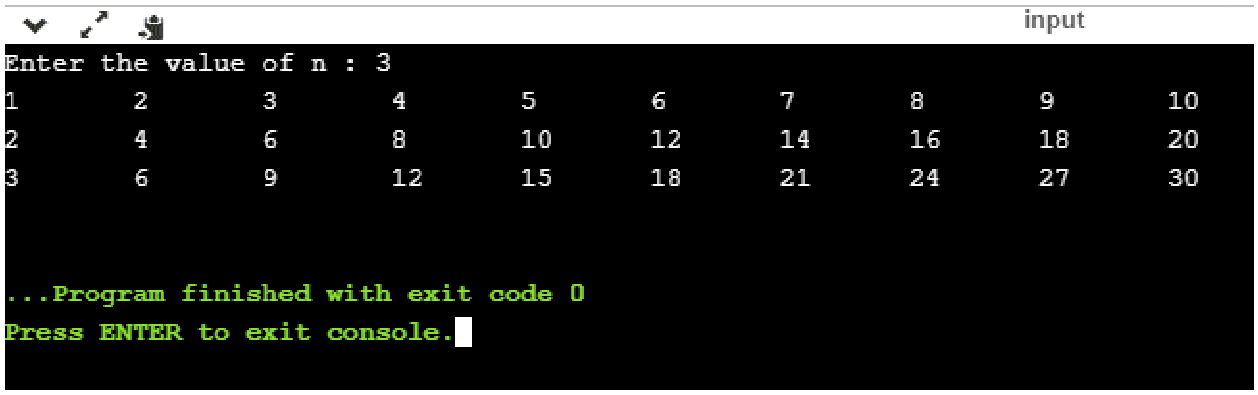


Fig: Flowchart for nested for loop

Example of Nested for loop:

```
#include <stdio.h>
int main()
{
   int n; // variable declaration
   printf("Enter the value of n :");
   // Displaying the n tables.
   for(int i=1;i<=n;i++)  // outer loop
   {
       for(int j=1;j<=10;j++)  // inner loop
       {
           printf("%d\t",(i*j)); // printing the value.
       }
       printf("\n");
   }
```

Output:



```
Enter the value of n : 3
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30


...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation:

- First, the 'i' variable is initialized to 1 and then program control passes to the i<=n.
- The program control checks whether the condition 'i<=n' is true or not.
- If the condition is true, then the program control passes to the inner loop.
- The inner loop will get executed until the condition is true.
- After the execution of the inner loop, the control moves back to the update of the outer loop, i.e., i++.
- After incrementing the value of the loop counter, the condition is checked again, i.e., i<=n.
- If the condition is true, then the inner loop will be executed again.
- This process will continue until the condition of the outer loop is true.

while loop:

In the while loop, condition is evaluated first and if it returns true then the statements inside while loop execute, this happens repeatedly until the condition returns false. When the condition returns false, the control comes out of loop and jumps to the next statement in the program after the while loop.

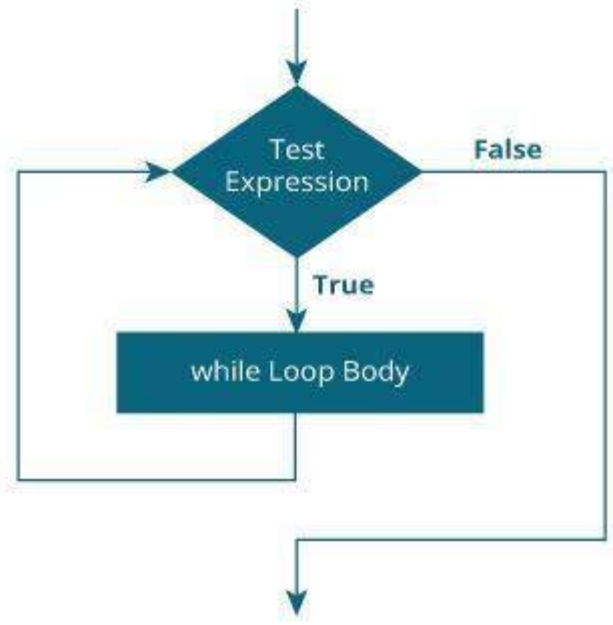
The syntax of the while loop is:

```
while (testExpression)
{
   // statements inside the body of the loop
}
```

How while loop works:

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

Flowchart of while loop:



Example :

// 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
```

Explanation:

Here, we have initialized i to 1.
1.When i is 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 is 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. When i is 6, the test expression i <= 5 will be false and the loop terminates.

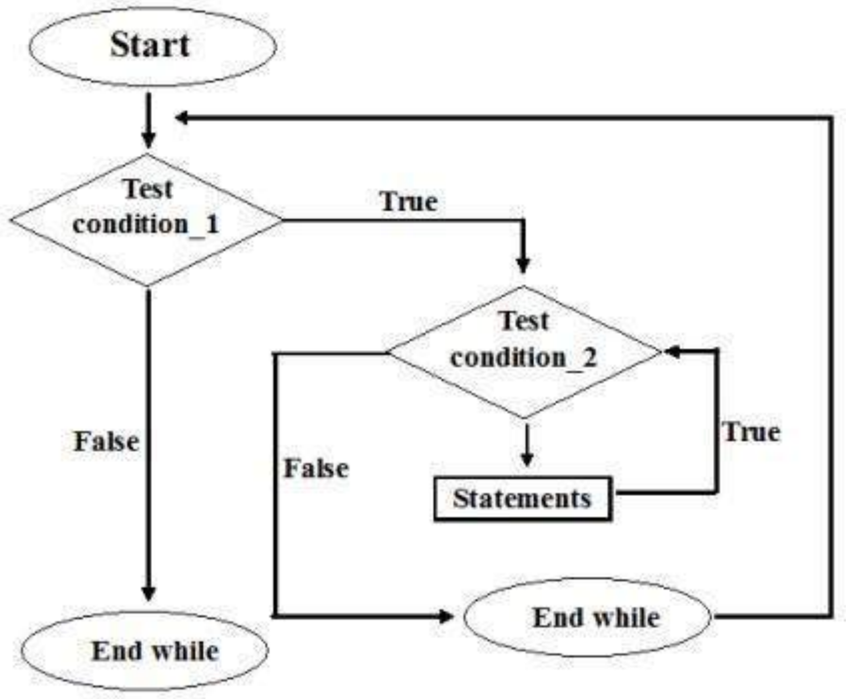
Nested while loop:

In nested while loops one or more statements are included in the body of the loop. ... In a nested while loop, the number of iterations will be equal to the number of iterations in the outer loop multiplied by the number of iterations in the inner loop which is most the same as nested for loop.

The syntax of the Nested while loop is:

```
while(condition)
{
    while(condition)
    {
         // inner loop statements.
    }
// outer loop statements.
}
```

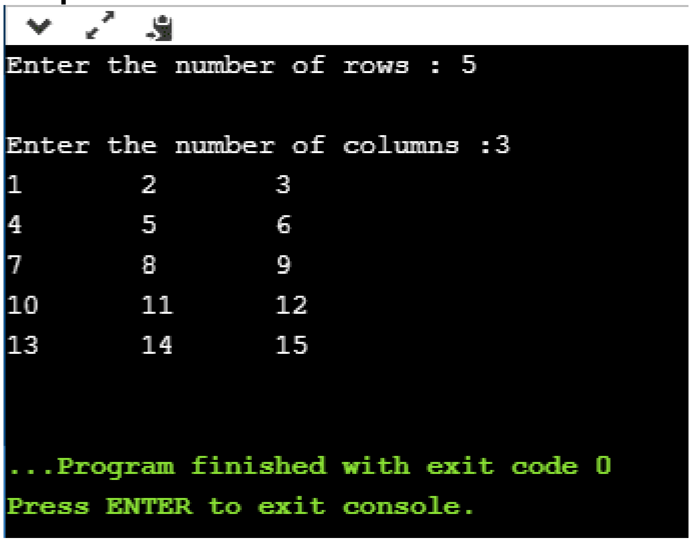
Flowchart of Nested while loop:



Example of nested while loop:

```
#include <stdio.h>
int main()
{
    int rows;  // variable declaration
    int columns; // variable declaration
    int k=1; // variable initialization
    printf("Enter the number of rows :");  // input the number of rows.
    scanf("%d",&rows);
    printf("\nEnter the number of columns :"); // input the number of columns.
    scanf("%d",&columns);
    int a[rows][columns]; //2d array declaration
    int i=1;
    while(i<=rows) // outer loop
    {
        int j=1;
        while(j<=columns)  // inner loop
        {
            printf("%d\t",k);  // printing the value of k.
            k++;   // increment counter
            j++;
        }
        i++;
        printf("\n");
    }
}
```

Output:



```
Enter the number of rows : 5

Enter the number of columns :3
1       2       3
4       5       6
7       8       9
10      11      12
13      14      15


...Program finished with exit code 0
Press ENTER to exit console.
```

Explanation:

- We have created the 2d array, i.e., int a[rows][columns].
- The program initializes the 'i' variable by 1.
- Now, control moves to the while loop, and this loop checks whether the condition is true, then the program control moves to the inner loop.
- After the execution of the inner loop, the control moves to the update of the outer loop, i.e., i++.
- After incrementing the value of 'i', the condition (i<=rows) is checked.
- If the condition is true, the control then again moves to the inner loop.
- This process continues until the condition of the outer loop is true.

do...while loop:

The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.

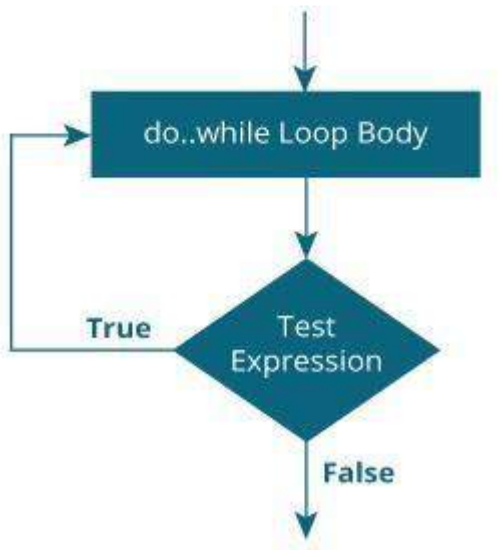
The syntax of the do...while loop is:

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

How do...while loop works:

The body of do...while loop is executed once. Only then, the test expression is evaluated.

- If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
- This process goes on until the test expression becomes false.
- If the test expression is false, the loop ends.
- Flowchart of do...while Loop:



Example :

```
// Program to add numbers until the user enters zero
#include <stdio.h>
int main()
{
   double number, sum = 0;

   // the body of the loop is executed at least once
   do
```

```
    {
        printf("Enter a number: ");
        scanf("%lf", &number);
        sum += number;
    }
    while(number != 0.0);

    printf("Sum = %.2lf",sum);

    return 0;
}
```

Output:

```
Enter a number: 1.5
Enter a number: 2.4
Enter a number: -3.4
Enter a number: 4.2
Enter a number: 0
Sum = 4.70
```