23ITT101 – PROGRAMMING IN C & DATA STRUCTURES

UNIT – II - DECISION STATEMENTS & FUNCTIONS

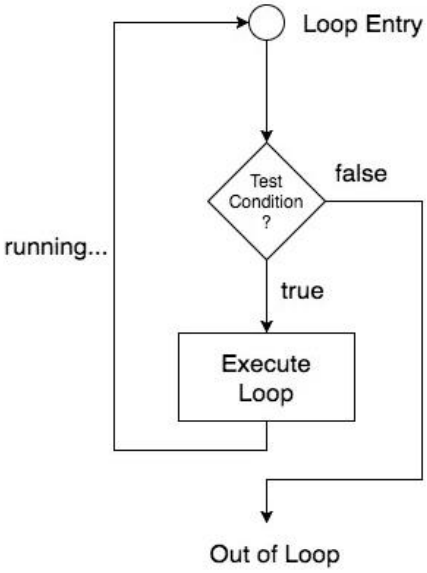
TOPIC – LOOPING STATEMENTS

THE LOOPING STATEMENTS

A segment of the program that is executed repeatedly is called as a loop. Some portion of the program has to be specified several number of times or until a particular condition is satisfied. Such repetitive operation is done through a loop structure. The loop concept is essential and it is fundamental for good programming.

The programs that we have developed so far uses either sequential or decision control statements. In sequential control statements, the calculations are carried out in a fixed order while in the decision control statements the statements are executed depending upon the outcome of the condition being tested.

Working of Loop



Loops are broadly classified into two types:

1. Entry controlled loops

In this kind of loop, the condition is checked before executing the loop's body. So, if the condition is never true, it won't execute even once. For example, `for` and `while` loop.

2. Exit controlled loops

In this kind of loop, the condition is checked after the loop's body is executed, i.e., in the end. Hence, even if the condition is not fulfilled, this loop will execute one time. The do-while loop is an example of exit controlled loop.

The Four Methods by which you can repeat a part of a program are:

By using a while loop

By using a do-while loop

By using a for loop

By using a nested for loop

The while loop

The simplest of all the looping structures in C is the while loop. The general form or the syntax of the while loop is:

Initialize loop counter; while(condition)

{

statement (s) ;

increment or decrement loop counter ;

}

Here, we can see that firstly, we initialize our iterator. Then we check the condition of `while` loop. If it is **false**, we exit the loop and if it is **true**, we enter the loop. After entering the loop, we execute the statements inside the `while` loop, update the iterator and then again check the condition. We do the same thing unless the condition is **false**.

Example:

```
#include <stdio.h>

int main()

{

    int n;

    printf("Enter the number of times you want to print your name:");

    scanf("%d", &n);

    char name[30];
```

```
    printf("Enter your name:");
    scanf("%s", name);
    while(n) {
        //here we are checking if n is non-zero
        printf("%s\n", name);
        n--;    //decrementing n
    }
    return 0;
}
```

Output

```
Enter the number of times you want to print your name:3
Enter your name: Looping
Looping
Looping
Looping
```

The do while loop

The do-while loop sometimes referred to as the do loop differs from its counterpart the while loop in checking the condition. The condition of the loop is not tested until the body of the loop has been executed once. If the condition is false, after the first loop iteration the loop terminates. However, if the condition is true the loop continues. The general form or the syntax of the do-while loop is

```
do
{
statement (s) ;
}
while (condition) ;
```

In do-while the statements would be executed at least once even if the condition fails for the first time itself.

Example

```
#include<stdio.h>

void main()

{

   int a, i;

   a = 5;

   i = 1;

   do

   {

      printf("%d\t", a*i);

      i++;

   }

   while(i <= 10);

}
```

Output

1 2 3 4 5 6 7 8 9 10

The for loop

The for loop is most commonly and popularly used loop in C. The for loop allows us to specify three things about the loop in single line. Initializing the value for the loop.

Condition in the loop counter to determine whether the loop should continue or not.

Increasing or decreasing the value of loop counter each time the program segment has been executed.

The syntax of the for loop is

for(initialization, condition; increment/decrement operation)

{

statement (s) ;

}

Example:

Program to print first 10 natural numbers using `for` loop

```
#include<stdio.h>

void main( )

{

   int x;

   for(x = 1; x <= 10; x++)

   {

      printf("%d\t", x);

   }

}
```

Output

1 2 3 4 5 6 7 8 9 10

Nested For Loop

We can also have nested `for` loops, i.e one `for` loop inside another `for` loop in C language. This type of loop is generally used while working with multi-dimensional arrays. Basic syntax for nested `for` loop is,

```
for(initialization; condition; increment/decrement)

{

   for(initialization; condition; increment/decrement)

   {

      statement ;

   }
```

}

Example

Program to print half Pyramid of numbers using Nested loops

```
#include<stdio.h>
void main( )
{
   int i, j;
   /* first for loop */
   for(i = 1; i < 5; i++)
   {
      printf("\n");
      /* second for loop inside the first */
      for(j = i; j > 0; j--)
      {
         printf("%d", j);
      }
   }
}
```

Output

```
1
21
321
4321
54321
```