Loops

Loops are used to repeat a block of code. Being able to have program repeatedly execute a block of code is one of the most basic but useful tasks in programming.

Loops are of two types.

a. Entry controlled loops:

In Entry controlled loops test condition is checked before entering the loop body. If the condition is satisfied then only the body of the loop gets executed. In C entry controlled loops are for, while.

b. Exit controlled loops

In Exit controlled Loops, condition is checked after executing the loop body. Here atleast once the body of loop gets executed. In C do...while is an exit controlled loop.

Based on the condition specified loops are further classified as

counter controlled loops
condition controlled loops

Counter Controlled loops

counter controlled loops are the loops where the number of times the loops will be executed is known in advance. That means, in this case, the value of the variable which controls the execution of the loop is previously known. The control variable is known as counter. A counter controlled loop is also called definite repetition loop.

Ex: for(int i=0;i<10;i++)

In the above example it is known in adance that loop iterates for 10 times.

Condition Controlled loops

In Condition controlled loops the number of times execution of the loop happens is unknown. In this case, the value of the control variable differs within a limitation and the execution can be terminated at any moment as the value of the variable is not controlled by the loop.

Ex:

```
int main()

{

    int flag=1;

     int a;
    scanf("%d",&a);

    while(flag)

    {

            if(a==10)

            {

                    flag=0;

            }

    }

     printf("%d",a);

}
```

In this example initially flag value is set to 1 and we don't know when the flag value will be false as it is dependent on some other condition. So the number of times the loop will be executed can't be predicted in advance.

<u>a. Entry Controlled Loops</u>

1. for loop

syntax:

```
for(initialization;condition;updation)
{
        stmts within for loop;
}
```

Variations of for loop:

```
a)      for(int i=1;i<11;i++)

        {
                printf("%d\n",i);
        }

b)      int i=1;        //variable initialization

        for(;i<11;i++)
```

```
            {
                printf("%d\n",i);
             }

c)          int i=1;        //intial expr

            for(;i<11;)
            {
                printf("%d\n",i);
                i++         //updation
            }

d)          for(;;)//always true(Infinte loop)

            {
                  printf("%d\n",i);
                  i++;     //updation
            }
```

Ex:Program to find factorial of given number.

```
#include<stdio.h>
int main()
{
        int n,fact=1,i=0;
        printf("Enter num:");
        scanf("%d",&n);
        for(i=n;i>0;i--)
        {
                fact=fact*i;
        }
        printf("Factorial of %d is %d\n\n",n,fact);
}
```

Ex: nested for:

A loop inside another loop is known as nested loop. C programming allows using one loop inside another loop.

Syntax

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

  for (initialization; condition; updation ) {
    statement(s);
  }

  statement(s);
}
```

Program to display triangle of * using nested for loop

```
#include<stdio.h>
int main()
{
        for(int i = 1; i <=5 ; i++)
        {
                for(int j = 1; j <= i; j++)
                {
                        printf("* ");
                }
                        printf("\n");
        }
  }
```

Output

```
*
* *
* * *
* * * *
* * * * *
```

Ex: C program to count the numbers divisible by 11.

```
#include<stdio.h>
int main()
{
        int a[20];
        int i,n,count=0;
        printf("enter how many numbers to read\n");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        for(i=0;i<n;i++)
        {
                if(a[i]%11==0)
                {
                        count=count+1;
                }
        }
        printf("count is %d",count);
        }
}
```

2. while loop

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Syntax

```
intial expr;
while(condition)
{
        stmts within loop;
        updation;
}
```

Ex's:

a) while(0)

```
{
    printf(“pesu\n”);
}
```

o/p: prints nothing

b) while(123)

```
{
    printf(“pesu\n”);
}
```

o/p: infinite loop

c) while(-12)

```
{
    printf(“pesu\n”);
}
```

o/p: infinite loop

d) int i=1;

```
while(i<11)
{
    printf("%d",i);
    i++;
}
```

o/p: 12345678910

e) while() //compile time error

f) int i=1;

```
    while(1)//always true (Infinte loop)
   {
       printf("%d\n",i);
         i++;
    }
```

o/p: infinite loop

g) int i=1;

```
while(i<11); //since expression is true it goes to infinite loop
{
     printf("%d",i);
     i++;
}
```

o/p:infinite loop

h) int i=12;

```
 while(i<11); //expression is false
{
     printf("%d",i);
     i++;
}
```

o/p: 12

<u>b. Exit Controlled Loops</u>

do...while loop

It is called as exit controlled loop since it tests the condition at the end of the loop body. The block of statements inside while loop are executed at least once.

Syntax:

```
intial expr;
do
{
        stmts within loop;
                updation;
}while(condition);
```

Ex's:

```
do
{
        printf("pesu\n");
}while(0);
```

o/p:pesu

```
do
{
        printf(“good\n”);
}while(12);
```

o/p: goes to infinite loop

```
int i=1;//intial expr
do
{
        printf("%d\n",i);
        i++;    //updation
}while(i<11);
```

o/p: 12345678910

Differences between while and do...while

	while	**do...while**

General syntax	while (condition)
	{
	
	statements; //body of loop
	
	}
		do
		{
		
		statements;// body of loop
		
		} while(Condition);

Condition	In 'while' loop the condition appears at the start of the loop.	In 'do-while' loop the condition appears at the end of the loop.
Iterations	The iterations do not occur if, the condition at the first iteration, is false.	The iteration occurs at least once even if the condition is false at the first iteration.