Iteration Statements/ Loop Control Statements

Iteration is a fundamental concept in programming that involves repeating a specific set of instructions ***multiple times*** until a certain condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.

A looping process would include the following four steps.

1 : Initialization of a condition variable.
2 : Test the condition.
3 : Executing the body of the loop depending on the condition.
4 : Updating the condition variable.

C language provides three iterative/repetitive loops.

1 : while loop
2 : for loop
3 : do-while loop

While Loop:

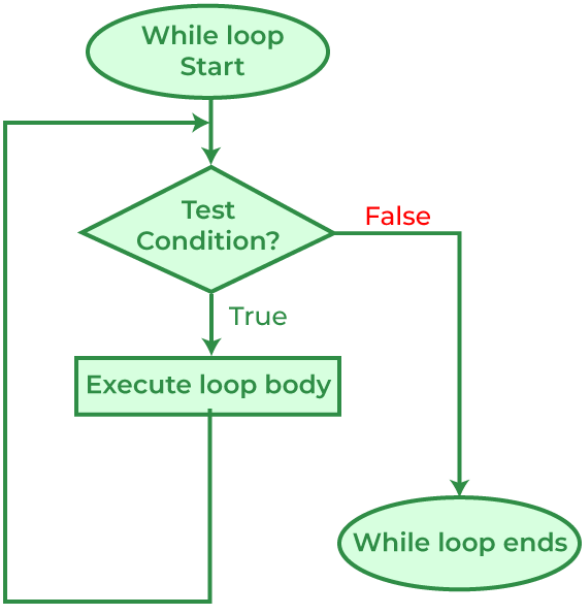
The ***while loop*** is used to execute a set of statements repeatedly as long as a certain condition is ***true.*** While loop can be addressed as an entry control loop.

Syntax :

variable initialization ;
While (condition)
{
statements ;
variable increment or decrement ;
}

Prasant Dash
7008191907

Flowchart of while loop in C



- Variable initialization.(e.g. int x=0;)
- condition(e.g. while(x<=10))
- Variable increment or decrement (x++ or x-- or x=x+2)

The while loop is an entry controlled loop statement, i.e. means the condition is evaluated first and it is true, then the body of the loop is executed. After executing the body of the loop, the condition is once again evaluated and if it is true, the body is executed once again, the process of repeated execution of the loop continues until the condition finally becomes false and the control is transferred out of the loop.

Prasant Dash
7008191907

Experiments #01 print 10 natural number

```
#include<stdio.h>
void main()
{
int i;
i=1;
        while(i<=10)
        {
        printf(" %d,",i);
        i++;
        }

}
```

Output: 1, 2, 3 ,4, 5, 6, 7, 8, 9 , 10

Experiments #02 Reverse a Number

```
#include <stdio.h>
 void main()
{
int n,reverse=0,dg;
printf("Enter any number");
scanf("%d",&n);
        while(n>0)
        {
                dg=n%10;
        reverse = reverse *10 + dg;
```


Prasant Dash
7008191907

```
        n=n/10;
        }
printf("%d", reverse);
}
```

Output Enter any Number 1234

4321

Experiments # 03 W.A.P Enter a number check the number is Armstrong number or not.

Experiments # 04 W.A .P Print factorial of a number

Experiments # 05.W.A.P finds GCD any two numbers

Experiments # 06. W.A.P Enter a number check the number is Palindrome number or not.

Experiments # 07 . W.A.P Enter a number check the number is strong number or not.

Experiments # 08. Binary to decimal convert

For Loop

- The **for loop** in C Language provides a functionality/feature to repeat a set of statements a defined number of times.
- The for loop is in itself a form of an **entry-**controlled loop.
- In this loop structure, more than one variable can be initialized.
- One of the most important features of this loop is that the three actions can be taken at a time like variable initialization, condition checking and increment/decrement.
- The for loop can be more concise and flexible than that of while and do-while loops.

Prasant Dash
7008191907

Syntax :

```
for(initialization; condition; increment/decrement)
{
Statements;
}
```

Structure of for Loop

1. **Initialization**: This step initializes a loop control variable with an initial value that helps to progress the loop or helps in checking the condition. It acts as the index value when iterating an array or string.

2. **Check/Test Condition**: This step of the for loop defines the condition that determines whether the loop should continue executing or not. The condition is checked before each iteration and if it is true then the iteration of the loop continues otherwise the loop is terminated.

3. **Body**: It is the set of statements i.e. variables, functions, etc that is executed repeatedly till the condition is true. It is enclosed within curly braces { }.

4. **Updation**: This specifies how the loop control variable should be updated after each iteration of the loop. Generally, it is the incrementation (variable++) or decre mentation (variable–) of the loop control variable.

How for Loop Works?

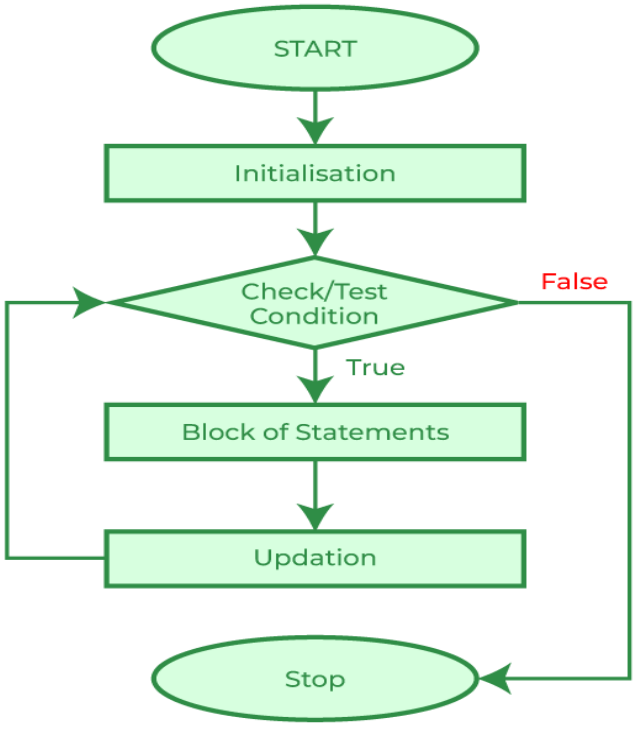
The working of for loop is mentioned below:

- **Step 1:** Initialization is the basic step of for loop this step occurs only once during the start of the loop. During Initialization, variables are declared, or already existing variables are assigned some value.

Prasant Dash
7008191907

- **Step 2:** During the Second Step condition statements are checked and only if the condition is the satisfied loop we can further process otherwise loop is broken.
- **Step 3:** All the statements inside the loop are executed.
- **Step 4:** Updating the values of variables has been done as defined in the loop. **Continue to Step 2 till the loop breaks.**

Flowchart of for Loop



Prasant Dash
7008191907

Example print 1 to 10 natural number

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

        for(i=1;i<=10;i++)
        {
        printf("%d\t",i);
        }
}
```

Output: 1 2 3 4 5 6 7 8 9 10

Various forms of FOR LOOP

I am using variable num in all the below examples –

1) Here instead of num++, I'm using num=num+1 which is nothing but same as num++.

for (num=10; num<20; num=num+1)

2) Initialization part can be skipped from loop as shown below, the counter variable is declared before the loop itself.

int num=10;

for (;num<20;num++)

Must Note: Although we can skip init part but semicolon (;) before condition is must, without which you will get compilation error.

Prasant Dash
7008191907

3) Like initialization, you can also skip the increment part as we did below. In this case semicolon (;) is must, after condition logic. The increment part is being done in for loop body itself.

```
for (num=10; num<20; )
{
//Code
num++;
}
```

4) Below case is also possible, increment in body and init during declaration of counter variable.

```
int num=10;
for (;num<20;)
{
//Statements
num++;
}
```

5) Counter can be decremented also, In the below example the variable gets decremented each time the loop runs until the condition num>10 becomes false.

```
for(num=20; num>10; num--)
```

6) If you don't initialize any variable, check condition and increment or decrement variable in for loop, it is known as infinitive for loop. In other words, if you place 2 semicolons in for loop, it is known as infinitive for loop.

```
for(; ;)
{
Printf("I read in MCA");
}
```


Prasant Dash
7008191907

Experiments: #01 Write C program to find factorial of any number

```
#include <stdio.h>
int main()
{
    int no, fact=1,i;
    printf("Enter a number\n");
    scanf("%d",&no);
    for(i=1;i<=no;i++)
        fact=fact*i;
    printf("Given no factorial is:%d",fact);

    return 0;
}
```

Output :Enter a number:5

Given no factorial is 120

Experiments: #02 W.A.P to print Fibonacci series enter by user upper rang

```
#include <stdio.h>
int main()
{

int a=0,b=1,c,n,i;
printf("Enter upper limit ");
scanf("%d",&n);
printf("%d %d",a ,b);
```


Prasant Dash
7008191907

```
for(i=1;i<=n;i++)
{
c=a+b;
a=b;
b=c;
printf(" %d ",c);
}

return 0;
}
```

Output: Enter upper limit 8

0 1 1 2 3 5 8 13

Experiments: #03 Check a number is prime number or not

Experiments: #04 C Programs to display all numbers from 1 to 100 those are power of 2

Experiments: #05 Write a program, where s=x1/1!+ x2/2!+ x3/3!..... up to xn/n! (Here 2! Means factorial of 2, that means2= 1*2, similarly 3!=1*2*3

```
#include<stdio.h>
#include<conio.h>

#include<math.h>
int main()
{
float i,x,no,fact=1;
float sum=0.0;

printf("enter x values:");
scanf("%f",&x);
```


Prasant Dash
7008191907

```
printf("enter range numbr:");
scanf("%f",&no);
for(i=1;i<=no;i++)
{
fact=fact*i;
sum=sum+(x*i)/fact;
 }
 printf("%f",sum);
 return 0;

}
```

Nested for loop

We can also have nested **for** loops, i.e. one **for** loop inside another **for** loop. Nesting is often used for handling multidimensional arrays.

Syntax:

```
for (initialization; condition; update)
{
        for(initialization; condition; update)
        {
            // inner loop statements.
        }
   // outer loop statements.
}
```


Prasant Dash
7008191907

Experiments #01 Multiplication Table

```
#include <stdio.h>
int main()
{
        int i,n;
        printf("enter a number multiplication table:");
        scanf("%d",&n);
        printf("\n**********************************************************
**********\n");
        for(i=1;i<=10;i++)

                printf("%d\t",n*i);
        for(i=1;i<=10;i++)
                printf("%d\t",i);
for(i=1;i<=10;i++)
                printf("%d\t",n);
printf("\n**********************************************************************
**********\n");

        return 0;
}
```

Experiments # 02 W.A.P print the prime number 1 to 100

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


Prasant Dash
7008191907

```
//clrscr();
for(i=1;i<=100;i++)
{
count=0;
        for(j=1;j<=i;j++)
        {
        if(i%j==0)
        count++;
        }
if(count==2)

printf("\n prime no is %d",i);
//else
//printf("\n not prime%d",i);
}
getch();
}
```

Experiments # 03 Display twin prime number 1 to 100

```
        #include<stdio.h>

        int main()
        {
        int i,j,n,a=0,pn=0,temp;

        for(i=3; i<=100; i++)
        {
        for(j=2; j<i; j++)
        {
```


Prasant Dash
7008191907

```
if((i%j)==0)
a=1;
}
if(a==0)
{
if(pn==0)
pn=i;

        else if(i-pn==2)
        {
        printf("\n Twin prime number:%d,%d",pn,i);
        pn=i;
        }
else
pn=i;
}
a=0;
}
return 0;
}
```

Experiments # 04 Half Pyramid of *

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


Prasant Dash
7008191907

```
#include <stdio.h>
int main()
{
        int i,j,no;

        printf("enter range\n");
        scanf("%d",&no);
        for(i=1;i<=no;i++)
                {
                for(j=1;j<=i;j++)
                        printf("*");

                printf("\n");
                }

        return 0;
}
```

Experiments # 05 Floyd's Triangle

```
            1
          1  2  1
        1  2  3  2  1
      1 2  3  4  3  2 1
    1  2  3  4  5   4  3  2  1
```

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


Prasant Dash
7008191907

```
for(i=1;i<=5;i++)
  {
  for(sp=1;sp<=5-i;sp++)
  printf(" ");
for(j=1;j<=i;j++)
printf("%d",j);
for(j=j-2;j>=1;j--)
printf("%d",j);

printf("\n");
   }

}
```

do-while loop

The **do-while** loop is an **exit controlled loop statement** The body of the loop are executed first and then the condition is evaluated. If it is true, then the body of the loop is executed once again. The process of execution of body of the loop is continued until the condition finally becomes false and the control is transferred to the statement immediately after the loop. The statements are always executed at least once.

Syntax :

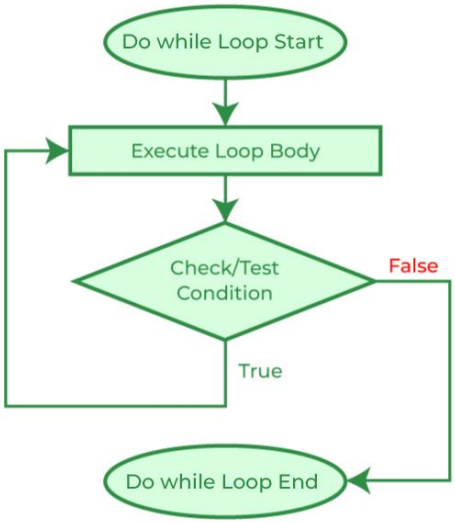
```
 variable initialization ;
do
{
statements ;
```


Prasant Dash
7008191907

Variable increment or decrement;

} **while (condition);**

Flowchart



Experiments #01 Example of do while program

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
  char c;
  int choice,dummy;
  do
      {
  printf("\n1. Print Hello\n2. Hello USBM\n3. Exit\n");
    printf("\n Enter choice:");
```


Prasant Dash
7008191907

```
    scanf("%d",&choice);
    switch(choice)
    {
       case 1 :
       printf("\n Hello");
       break;
       case 2:
       printf("\n Hello USBM");
       break;
       case 3:
       exit(0);
       break;
       default:
       printf("\n please enter valid choice");
    }
    printf("\ndo you want to enter more?");
    scanf("%d",&dummy);
    scanf("%c",&c);
    }while(c=='y');
}
```

Experiments #02 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: ");
```


Prasant Dash
7008191907

```
    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
```

While vs. Do-While

Parameter	**While**	**Do-While**
Definition	Loop body is executed after the given condition is evaluated.	Loop body is executed, and then the given condition is checked.
Variable Initialization	Variables are initialized before the execution of the loop.	Variables may initialize before or within the loop.
Loop Type	Entry Control Loop	Exit Control Loop.
Semicolon	Semicolon is not used as a part of the syntax.	Semicolon is used as a part of the syntax.

Syntax	while(condition){
	// loop body
	}
		do{
		// loop body
		} while (condition);

Prasant Dash
7008191907

Jump Statements

Jumping statements are used to transfer the program's control from one location to another; these are set of keywords which are responsible to transfer program's control within the same block or from one function to another**.**

There are four jumping statements in C language:

- ❖ goto statement
- ❖ return statement
- ❖ break statement
- ❖ continue statement

goto statement

Goto statement is a form of jump statement; it is used to jump from one block to another block during execution. It is often also termed as an *unconditional jump statement*.

Syntax :

```
goto label;
. . . . .
. . . . .
label:
statements;
```

Experiments#01

```
#include <stdio.h>
int main()
{
  int num,i=1;
```


Prasant Dash
7008191907

```
 printf("Enter the number whose table you want to print?");
 scanf("%d",&num);
 table:
 printf("%d x %d = %d\n",num,i,num*i);
 i++;
 if(i<=10)
 goto table;
}
```

Break Statement

Break is a keyword. When a break statement is encountered inside a loop ,switch or if , the loop is immediately terminated and the program control resumes at the next statement following the loop.

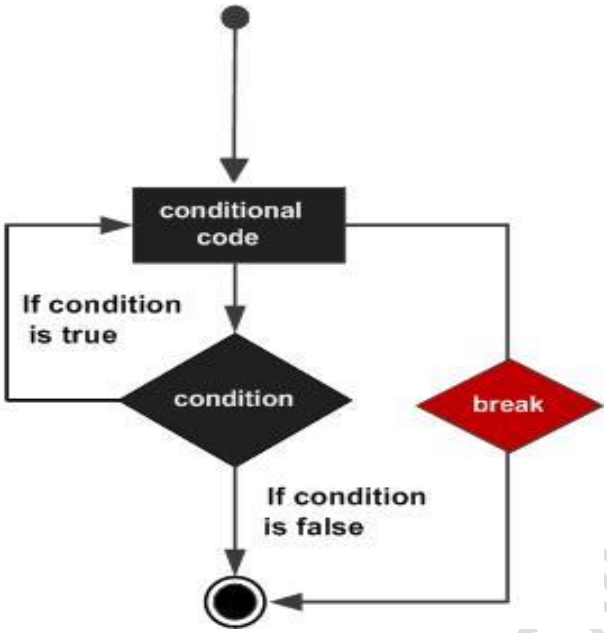
If you are using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax of break statement

```
break;
```

Flow chart

Prasant Dash
7008191907



How break statement works?

```
while (testExpression) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

```
do {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```


Prasant Dash
7008191907

Experiments: #01

```
#include<stdio.h>
void main()
{
        int i = 0;

while (i < 10)
{
          if (i == 4)
          {
                  break;
          }
 printf("%d\t", i);
 i++;
}
}
```

Output: 0 1 2 3

Break in use for loop

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
   int i;
   for(i = 0; i<10; i++)
   {
      printf("%d ",i);
      if(i == 5)
```


Prasant Dash
7008191907

```
        break;
    }
    printf("came outside of loop i = %d",i);

}
```

0 1 2 3 4 5 came outside of loop i = 5

Continue Statement

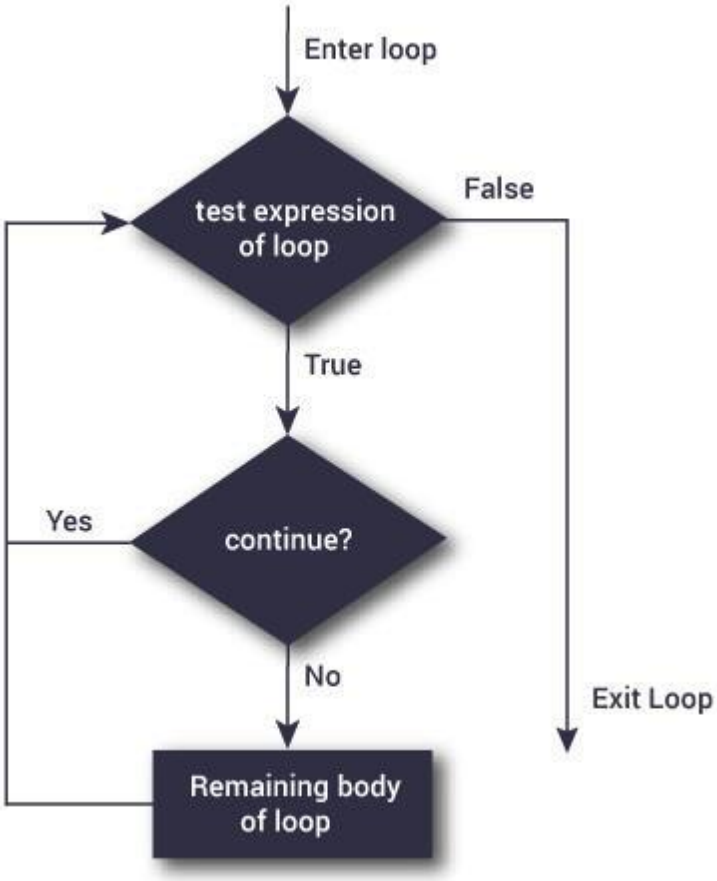
Continue is keyword exactly opposite to break. The continue statement is used for continuing next iteration of loop statements. When it occurs in the loop it does not terminate, but it skips some statements inside the loop / the statements after this statement. . The continue statement is used/ associated with decision making statement such as if , if-else but not in switch case .

Syntax of continue Statement

```
continue;
```

Flowchart of continue Statement

Prasant Dash
7008191907



How continue statement works?

Prasant Dash
7008191907

```
while (test Expression)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```

```
for (init, condition, update)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```

Experiments #01

```
#include <stdio.h>
int main()
 {
  int i = 0;

  while (i < 10)
{
      if (i == 4)
     {
        i++;
        continue;
      }
   printf("%d\t", i);
   i++;
```


Prasant Dash
7008191907


```
    }

    return 0;
}
```

Output: 0 1 2 3 5 6 7 8 9

Difference between break and continue statements

Break	Continue
1 : Break statement takes the control to the outside of the loop	1: Continue statement takes the control to the beginning of the loop..
2 : It is also used in switch statement.	2 : This can be used only in loop statements.
3 : Always associated with if condition in loops.	3 : This is also associated with if condition.

Prasant Dash
7008191907