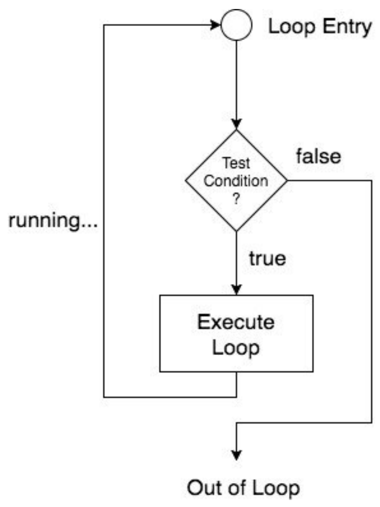
Loops in C

- During programming, sometimes we might need to execute a certain code statement again and again.
- We can write the code statement as many times as we need it to execute but that would be very inefficient, because what if you want a code statement to execute a 100 times? This is why we use loops.
- In any programming language including C language, loops are used to execute a single statement or a set of statements, repeatedly, until a particular condition is satisfied.

How Loops in C works?
The below diagram depicts a loop execution,



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.

Types of Loop in C

There are 3 types of Loop in C language, namely:

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

for loop

- The **for loop** in C is used to execute a set of statements repeatedly until a particular condition is satisfied. We can say it is an open ended loop. General format is,

Syntax :

```
for(Initialiazation;  condition; Increament/Decrement);
{
                    Statement –Block;
}
```

The for loop is executed as follows:

1. It first evaluates the initialization code.
2. Then it checks the condition expression.
3. If it is true, it executes the for-loop body.
4. Then it evaluate the increment/decrement condition and again follows from step 2.
5. When the condition expression becomes false, it exits the loop.

ex 1 : Program to print repeated statement using for loop;

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

    printf("\nEnter a Number : ");
    scanf("%d",&n);

    for(i=1; i<=n; i++)
    {
       printf("\nHello World!");
    }

    return 0;
}
```

Output :

```
Enter a Number : 5

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

ex 2 : Program to print natural number using for loop.

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

    printf("\nEnter a Number elments : ");
    scanf("%d",&n);

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

    return 0;
}
```

Output :

```
Enter a Number elments : 10
1       2       3       4       5       6       7       8       9       10
```

while loop in C

The while loop is an entry controlled loop. It is completed in 3 steps.

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

Syntax of while Loop:

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

ex 1 : Program to print repeated statement using for loop;

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

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   i=1;
   while(i<=n)
   {
      printf("\nHello World!");
      i++;
   }

   return 0;
}
```

Output :

```
Enter a Number : 5

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

ex 2 : Program to find the Factorial of given number using while loop.

```
#include<stdio.h>
int main()
{
   int n,i,fact;

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   fact=1;
   i=1;
   while(i<=n)
   {
      fact=fact*i;
      printf("\nFactrial of %d is : %d ",i,fact);
      i++;
   }


   return 0;
}
```

Output :

```
Enter a Number : 7

Factrial of 1 is : 1
Factrial of 2 is : 2
Factrial of 3 is : 6
Factrial of 4 is : 24
Factrial of 5 is : 120
Factrial of 6 is : 720
Factrial of 7 is : 5040
```

do-while loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax :

```
Initialization;
do
{
    // the body of the loop
    Increment/Decrement;
}
while (testExpression);
```

ex 1 : Program to print repeated statement using for loop;

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

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   i=1;
   do
   {
      printf("\nHello World!");
      i++;
   }while(i<=n);

   return 0;
}
```

Output :

```
Enter a Number : 5

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

ex 2 : Program to find the Factorial of given number using while loop.

```
#include<stdio.h>
int main()
{
   int n,i,fact;

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   fact=1;
   i=1;
   do
   {
      fact=fact*i;
      printf("\nFactrial of %d is : %d ",i,fact);
      i++;
   } while(i<=n);


   return 0;
}
```

Output :

```
Enter a Number : 10

Factrial of 1 is : 1

Factrial of 2 is : 2

Factrial of 3 is : 6

Factrial of 4 is : 24

Factrial of 5 is : 120

Factrial of 6 is : 720

Factrial of 7 is : 5040

Factrial of 8 is : 40320

Factrial of 9 is : 362880

Factrial of 10 is : 3628800
```

Nested loop :

A nested loop means a loop statement inside another loop statement. That is why nested loops are also called “loop inside loops“. We can define any number of loops inside another loop.

OR

Defining one loop body into another loop body is nothing but nested loop.

- Nested loops are generally used in sorting techniques , arrays elements etc.
- pattern printing.

ex 1:

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

      printf("\nEnter a Number : ");
      scanf("%d",&n);

      for(i=1; i<=n; i++)
      {
         for(j=1; j<=n; j++)
         {
             printf("%d ",i);
         }
         printf("\n");
       }

      return 0;
}
```

Output :

```
Enter a Number : 5
1  1  1  1  1
2  2  2  2  2
3  3  3  3  3
4  4  4  4  4
5  5  5  5  5
```

ex 2 :

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

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   for(i=1; i<=n; i++)
   {
      for(j=1; j<=n; j++)
      {
         printf("* ");
      }
      printf("\n");
   }

   return 0;
}
```

```
Enter a Number : 5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

ex 3 :

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

   printf("\nEnter a Number : ");
   scanf("%d",&n);

   for(i=1; i<=n; i++)
   {
      for(j=1; j<=i; j++)
      {
         printf("* ");
      }
      printf("\n");
   }

   return 0;
}
```

```
Enter a Number : 5
*
*  *
*  *  *
*  *  * *
*  *  *  *  *
```

Jump or Branching statements :

- Jump statements are used to interrupt the normal flow of programs. Jump statements are used to jump from one section of the program to another.

Types of jump statements in C.

Break;
continue;
goto label;
return;

break :

Break statement is used to come out/break from looping statements and/or switch statements .

- It skips the rest of the iterations inside loops.
- Break statement can be conditional or unconditional.

ex :

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

     printf("\nEnter a Number : ");
     scanf("%d",&n);

     for(i=1; i<=n; i++)
     {
        if(i==3)
        {
           break;
         }
         printf("\n%d",i);
      }

    return 0;
}
```

Output :

```
Enter a Number : 5

1
2
```

Continue :

- It skips the current/present iteration in looping statements.
- It will not execute the current iteration but it executes remaining iterations.

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

    printf("\nEnter a Number : ");
    scanf("%d",&n);

    for(i=1; i<=n; i++)
    {
        if(i==3)
        {
          continue;

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

    return 0;
}
```

Output :

```
Enter a Number : 10

1
2
4
5
6
7
8
9
10
```

3 is not print because of the condition if(i==3).

goto statement : To jump from one part of the program to another part, parts are nothing but labels;

goto keyword is used to execute label block**.**

Syntax :

Labelname : It is a name of the part of the program, must be a valid identifier.

Label creation Syntax :

```
        Labelname:
```

ex 1 : Program to find the you are eligible for vote or not.

```
#include<stdio.h>
int main()
{
    int age;
    read:

    printf("\nEnter Your Age : ");
    scanf("%d",&age);

    if(age<18)
    {
      printf("your are not matured");
      goto read;
    }
    else
    {
      goto next;
    }
    next:
    printf("\nYes your are eligible for voting...");

    return 0;
}
```

Output 1

```
Enter Your Age : 18
Yes your are eligible for voting...
```

Output 2

```
Enter Your Age : 16
your are not matured
```

ex 2 : Program to read numbers from the keyboard continuously till the user presses 999 and to find the sum of only positive numbers.

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

     read:

      printf("\nEnter a NUmber : ");
      scanf("%d",&n);

      if(n!=999)
      {
         if(n>=0)
         {
            sum=sum+n;
         }
         goto read;
      }

      printf("\nSum of Positive Number = %d ",sum);

      return 0;
}
```

Output :

```
Enter a Number : 10

Enter a Number : 10

Enter a Number : 10

Enter a Number : 10

Enter a Number : 10

Enter a Number : 20

Enter a Number : 20

Enter a Number : 20

Enter a Number : 20

Enter a Number : 20

Enter a Number : 999

Sum of Positive Number = 150
```

return statement :

- it returns the control along with optional value back to the caller.
- It is used in functions body.

Ex :

```
#include<stdio.h>
int add(int x, int y);
int main()
{
    int res,x,y;

    printf("\nEnter two numbers  : ");
    scanf("%d%d",&x,&y);

    res=add(x,y);
    printf("\nResult = %d",res);

    return 0;
}
int add(int x, int y)
{
    int res;
    res=x+y;
    return res;
}
```

Output :

```
Enter two numbers  : 10 20
Result = 30
```