A **Loop** executes the sequence of statements many times until the stated condition becomes false. A loop consists of two parts, **a body of a loop** and a **control statement**. The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the loop is to repeat the same code a number of times.

'C' programming language provides us with three types of loop constructs:

**1. The while loop**

**2. The do-while loop**

**3. The for loop**

**While Loop in C**

A while loop is the most straightforward looping structure. Syntax of while loop in C programming language is as follows:

while (condition) {

 statements;

}

It is an entry-controlled loop. In while loop, a condition is evaluated before processing a body of the loop. If a condition is true then and only then the body of a loop is executed. After the body of a loop is executed then control again goes back at the beginning, and the condition is checked if it is true, the same process is executed until the condition becomes false. Once the condition becomes false, the control goes out of the loop.

After exiting the loop, the control goes to the statements which are immediately after the loop. The body of a loop can contain more than one statement. If it contains only one statement, then the curly braces are not compulsory. It

**Note**--In while loop, if the condition is not true, then the body of a loop will not be executed, not even once. It is different in **do while loop** .

Following program illustrates while loop in C programming example:

#include<stdio.h>

#include<conio.h>

int main()

{

 int num=1; //initializing the variable

 while(num<=10) //while loop with condition

 {

 printf("%d\n",num);

 num++; //incrementing operation

 }

 return 0;

}

Output:

1

2

3

4

5

6

7

8

9

10

The above program illustrates the use of while loop. In the above program, we have printed series of numbers from 1 to 10 using a while loop.

## Do-While loop in C

A do...while loop in C is similar to the while loop except that the condition is always executed after the body of a loop. It is also called an exit-controlled loop.

Syntax of do...while loop in C programming language is as follows:

 do {

 statements

} while (expression);

As we saw in a while loop, the body is executed if and only if the condition is true. In some cases, we have to execute a body of the loop at least once even if the condition is false. This type of operation can be achieved by using a do-while loop.

In the do-while loop, the body of a loop is always executed at least once. After the body is executed, then it checks the condition. If the condition is true, then it will again execute the body of a loop otherwise control is transferred out of the loop.

Similar to the while loop, once the control goes out of the loop the statements which are immediately after the loop is executed.

Below is a do-while loop in **C example** to print a table of number 2:

#include<stdio.h>

#include<conio.h>

int main()

{

 int num=1; //initializing the variable

 do //do-while loop

 {

 printf("%d\n",2\*num);

 num++; //incrementing operation

 }while(num<=10);

 return 0;

}

Output:

2

4

6

8

10

12

14

16

18

20

In the above example, we have printed multiplication table of 2 using a do-while loop. Let's

**Difference between while and do while**

The while test the condition before executing any of the statement within the while loop.But the do-while test the condition after having executed the statements within the loop.This means that do-while would execute its statement at least once,even if the condition fails for the first time.The while ,on the other hand will not execute its statements if the condition fails for the first time.

#include<stdio.h>

int main()

{

while(4<1) // **example shown using while loop**

printf(“hello there\n”);

return 0;

}

**Note-here ,since the condition fails the first time itself,the printf() will not get executed at all.**

#include<stdio.h>

int main()

{

do // example shown using do-while loop

 {

 printf(“hello there\n”);

 }

 while(4<1)

return 0;

}

**Output-**

Hello there

**Note—in this program,the printf() would be executed once,since first the body of the loop is executed and then the condition is tested.**

**For loop in C**

A for loop is a more efficient loop structure in 'C' programming. The general structure of for loop syntax in C is as follows:

for (initial value; condition; incrementation or decrementation )

{

 statements;

}

* The initial value of the for loop is performed only once.
* The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.
* The incrementation/decrementation increases (or decreases) the counter by a set value.

Following program illustrates the for loop in C programming example:

#include<stdio.h>

int main()

{

 int number;

 for(number=1;number<=10;number++) //for loop to print 1-10 numbers

 {

 printf("%d\n",number); //to print the number

 }

 return 0;

}

Output:

1

2

3

4

5

6

7

8

9

10

The above program prints the number series from 1-10 using for loop.

In C, the for loop can have multiple expressions separated by commas in each part.

**For example:**

for (x = 0, y = num; x < y; i++, y--) {

 statements;

}

Also, we can skip the initial value expression, condition and/or increment by adding a semicolon.

**For example**:

int i=0;

int max = 10;

for (; i < max; i++) {

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

}

Notice that loops can also be nested where there is an outer loop and an inner loop. For each iteration of the outer loop, the inner loop repeats its entire cycle.

Consider the following example, that uses nested for loop in C programming to output a multiplication table:

#include <stdio.h>

int main() {

int i, j;

int table = 2;

int max = 5;

for (i = 1; i <= table; i++) { // outer loop

 for (j = 0; j <= max; j++) { // inner loop

 printf("%d x %d = %d\n", i, j, i\*j);

 }

 printf("\n"); /\* blank line between tables \*/

}}

Output:

1 x 0 = 0

1 x 1 = 1

1 x 2 = 2

1 x 3 = 3

1 x 4 = 4

1 x 5 = 5

2 x 0 = 0

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

2 x 5 = 10

The nesting of for loops can be done up-to any level. The nested loops should be adequately indented to make code readable. In some versions of 'C,' the nesting is limited up to 15 loops, but some provide more.

The nested loops are mostly used in **array applications** .

## Break Statement in C

The break statement is used mainly in in the switch statement. It is also useful for immediately stopping a loop.

We consider the following program which introduces a break to exit a while loop:

#include <stdio.h>

int main() {

int num = 5;

while (num > 0) {

 if (num == 3)

 break;

 printf("%d\n", num);

 num--;

}}

Output:

5

4

## Continue Statement in C

When you want to skip to the next iteration but remain in the loop, you should use the continue statement.

For example:

#include <stdio.h>

int main() {

int nb = 7;

while (nb > 0) {

 nb--;

 if (nb == 5)

 continue;

 printf("%d\n", nb);

}}

Output:

6

4

3

2

1

So, the value 5 is skipped.