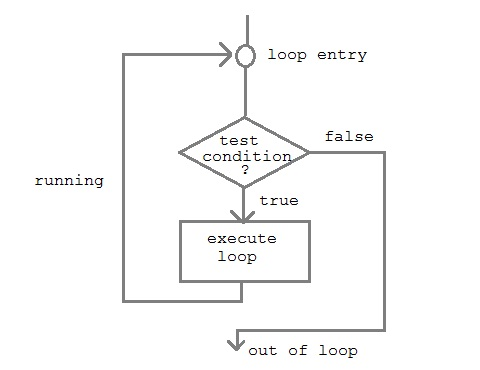
**LOOPING IN C**

Loops are used in programming to repeat a specific block until some end condition is met. There are three loops in C programming:

1. for loop
2. [while loop](https://www.programiz.com/c-programming/c-do-while-loops)
3. [do...while loop](https://www.programiz.com/c-programming/c-do-while-loops)

**How it Works**

The below diagram depicts a loop execution,



1. **for loop:**

for loop 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,

for(initialization; condition; increment/decrement)

{

statement-block;

}

A **for loop** will run statements a set number of times. For example, let's say you have 15 employees. You want to update each employee and add a bonus of 5% to their pay rate, the for loop would be used to look at each one. Below is the code that would execute this:

1. int emp\_count = 15;
2. int i;
3. double pay\_rate;
4. for(i = 0; i < emp\_count; i++) {
5. pay\_rate = pay\_rate \* 1.05;
6. }

If you had 50,000 employees, this would be a very manually-intensive operation! But thanks to the for loop, we can accomplish the task very quickly.

1. **while loop:**

while loop can be addressed as an **entry control** 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 :**

variable initialization;

while(condition)

{ statements;

variable increment or decrement;

}

We can convert the for loop above to a while loop:

1. int emp\_count = 15;
2. double pay\_rate;
3. int i = 0;
4. while(i < emp\_count) {
5. pay\_rate = pay\_rate \* 1.05;
6. i++;
7. }

Note that we incremented the counter, *i*. If we had not done this, the loop would have run forever, creating an **infinite loop**. An infinite loop is one with no way out.

1. **do while loop:**

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. It means that the body of the loop will be executed at least once, even though the starting condition inside while is initialized to be **false**.

General syntax is,

do

{.....

….

}

while(condition);

Let's re-work the previous loop into a do...while loop:

1. int emp\_count = 15;
2. double pay\_rate;
3. int i=0;
4. do
5. {
6. Pay\_rate = pay\_rate\*1.05;
7. i++;
8. }
9. While(i < emp\_count);

**Break Statement:**

The break statement is used inside loops and [switch case](https://beginnersbook.com/2014/01/switch-case-statements-in-c/).

* When a break statement is encountered inside a loop, the control directly comes out of loop and the loop gets terminated.
* Whenever it is encountered in switch-case block, the control comes out of the switch-case

#include <stdio.h>

int main()

{

int num =0;

while(num<=100)

{

printf("value of variable num is: %d\n", num);

if (num==2)

{

break;

}

num++;

}

printf("Out of while-loop");

return 0;

}

**Output:**

var: 100

var: 99

Out of for-loop

**Continue Statement:**

The **continue statement** is used inside [loops](https://beginnersbook.com/2014/01/c-loops-examples/). When a continue statement is encountered inside a loop, control jumps to the beginning of the loop for next iteration, skipping the execution of statements inside the body of loop for the current iteration.

#include <stdio.h>

int main()

{

int counter=10;

while (counter >=0)

{

if (counter==7)

{

counter--;

continue;

}

printf("%d  ", counter);

counter--;

}

return 0;

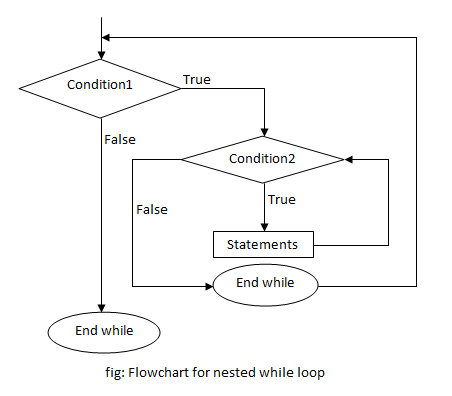
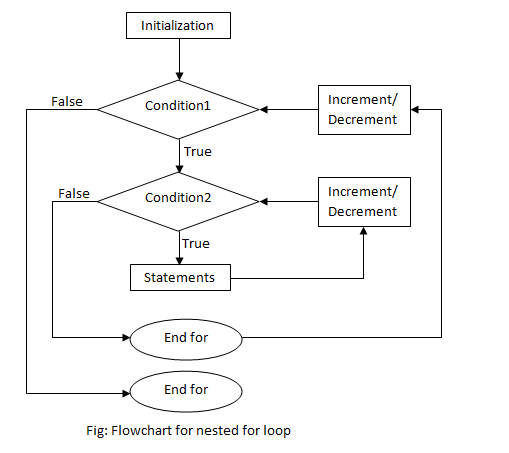
}

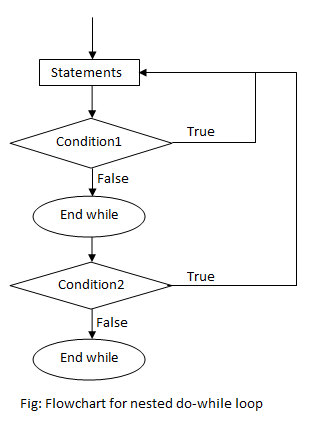
**Output:**

10 9 8 6 5 4 3 2 1 0

**Nested Loops:**

* A loop within another loop is called nested loop. A loop (e.g., while, do-while, for) can be placed within the body of another loop.
* When one loop is nested within another, several iterations of the inner loop are performed for every single iteration of the outer loop.





Now let’s try a small program with nested loops.

|  |
| --- |
| 1. #include<stdio.h> 2. void main() 3. { int row,col; 4. for(row=1;row<4;row++){ 5. for(col=1;col<4;col++){ 6. printf("%d\t%d\n",row,col); 7. }}} |