Iterative Structures (Loops)

Mohamed MESSABIHI

mohamed.messabihi@gmail.com

University of Tlemcen
Department of Computer Science

https://sites.google.com/site/informatiquemessabihi/

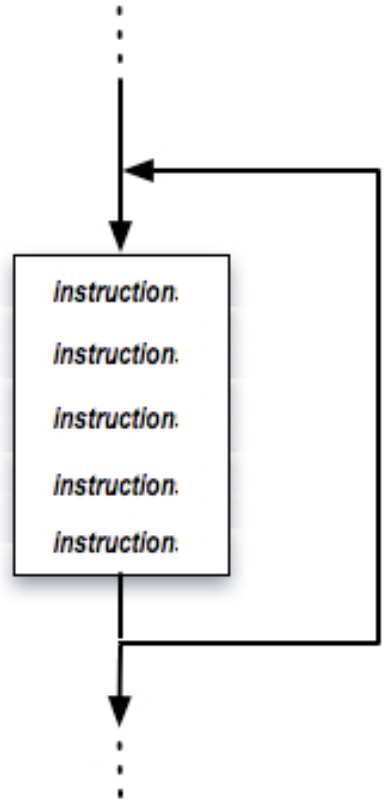
Some Examples Before We Begin!

Exercise

1. Write a program that displays the number 1 ten times.
2. Modify the program to display the number 1 one hundred times.
3. Write a program that displays numbers from 1 to 100.
4. Modify the previous program to display numbers from 1 to 1000.
5. Modify the program to display only even numbers.

What is a Loop?

A loop is a control structure that allows you to repeat the same instructions multiple times.



There are three common loop types in C:

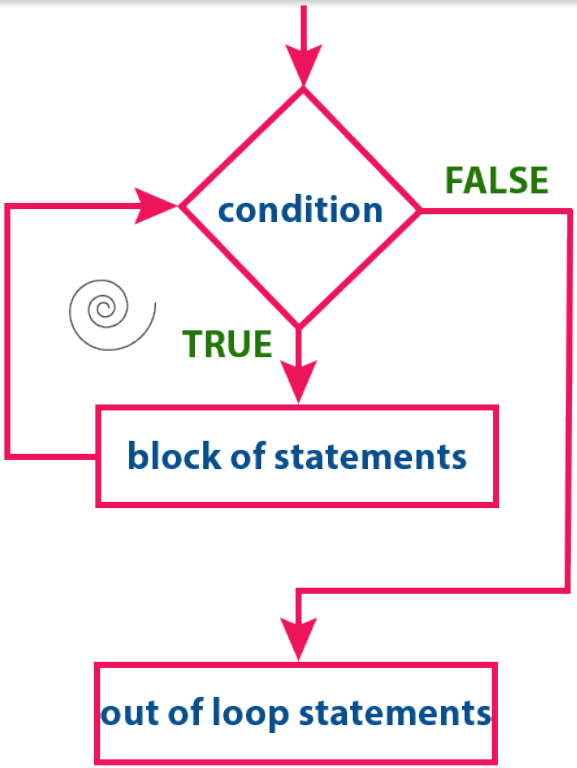
1. **while**
2. **do... while**
3. **for**

An iteration is the term used for each pass through a loop.

The "While" Loop

Syntaxe :

```
while (Condition)
{
  // Block of instructions
}
```



- The condition (loop control condition) is evaluated before each

Example of a "While" Loop

Let's create a program to validate the input of a positive integer.

Example:

```
int positiveInteger = 0;

while (positiveInteger <= 0)
{
    printf("Enter a positive integer: ");
    scanf("%d", &positiveInteger);
}
```

Repeating a Specific Number of Times

To achieve this, we create a counter variable that starts at 0 and increments with each iteration.

Example:

```c
int counter = 0;

while (counter < 10)
{
    printf("The variable counter is %d\n", counter);
    counter++; // equivalent to counter = counter + 1;
}
```

Incrementing a variable means adding 1 to it (e.g., ‘variable++;‘).

Beware of Infinite Loops

Infinite loops occur when the number of iterations in a ‘while‘ loop is not known in advance. It depends on the condition evaluation.
Make sure your loops can terminate at some point. If the condition always remains true, your program will run indefinitely!
One of the instructions inside the loop must change the condition from true to false after a certain number of iterations.

Example:

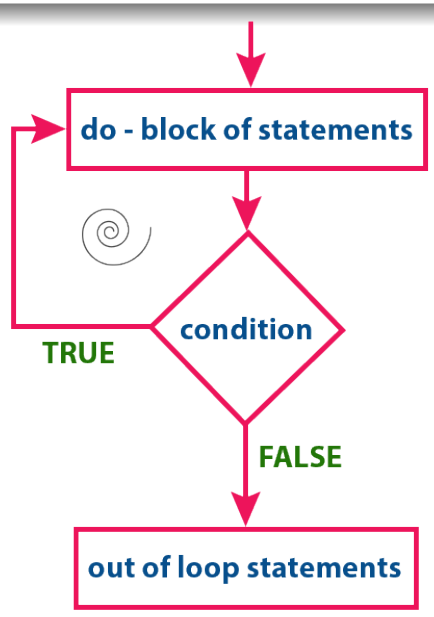
```
int counter = 0;

while (counter >= 0)
{
    printf("The variable counter is %d\n", counter);
    counter++; // equivalent to counter = counter + 1;
}
```

The "do...while" Loop

Syntax:

```
do
{
    // Block of instructions
} while (Condition);
```



- The 'do...while' loop is very similar to 'while'.
- The main difference is the position of the condition. In 'do...while', the condition is at the end, so the loop always executes at least once.

Example of a "do...while" Loop

Here's an example using a 'do...while' loop to print the variable 'n' from 0 to 9.

Example:

```
int n = 0;

do
{
    printf("The variable n is %d\n", n);
    n++;
} while (n < 10);
```

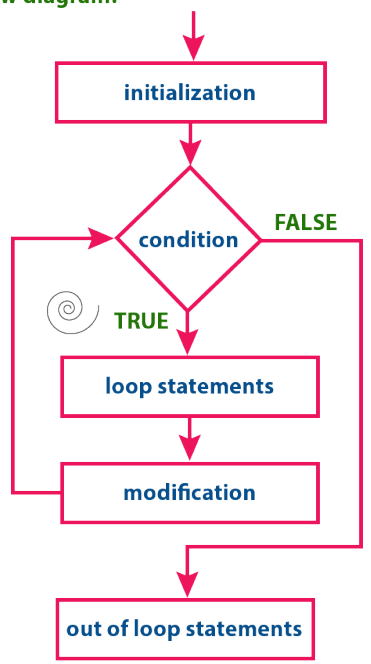
The "for" Loop

- The ‘for‘ loop is a compact loop structure that combines loop initialization, condition testing, and incrementing the loop variable.
- It’s very suitable when you know in advance how many times you want to iterate.

Syntax:

```
for( initialization ; condition ; modification )
{
    ...
    block of statements;
    ...
}
```

Execution flow diagram:



Example of a "for" Loop

Here's an example using a ‘for‘ loop to print the variable ‘i‘ from 0 to 9.

Example:

```
for (int i = 0; i < 10; i++)
{
    printf("The variable i is %d\n", i);
}
```

Summary of Loop Types

- The ‘while‘ loop repeats a block of instructions as long as a condition is true.
- The ‘do...while‘ loop is similar to ‘while‘, but it guarantees at least one execution of the loop body.
- The ‘for‘ loop is a compact loop structure that combines initialization, condition, and incrementing.

Loop Control Statements

Loop control statements allow you to modify the flow of loop execution:

- ‘break‘: Terminates the loop and transfers control to the statement following the loop.
- ‘continue‘: Skips the current iteration and jumps to the next iteration of the loop.

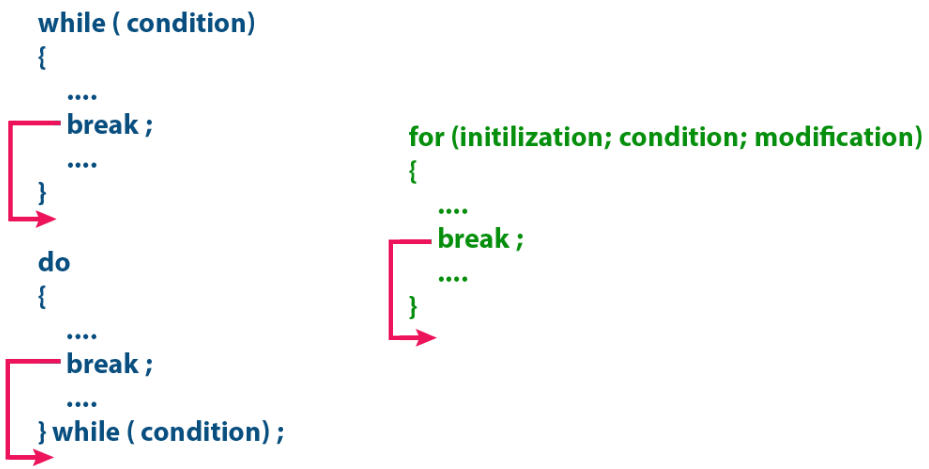
These statements can be used in ‘for‘, ‘while‘, and ‘do...while‘ loops.

"break" Statement

The 'break' statement can be used to exit a loop prematurely.

Example:

```
for (int i = 0; i < 10; i++)
{
    if (i == 5)
    {
        break; // exit the loop when i equals 5
    }
    printf("The variable i is %d\n", i);
}
```

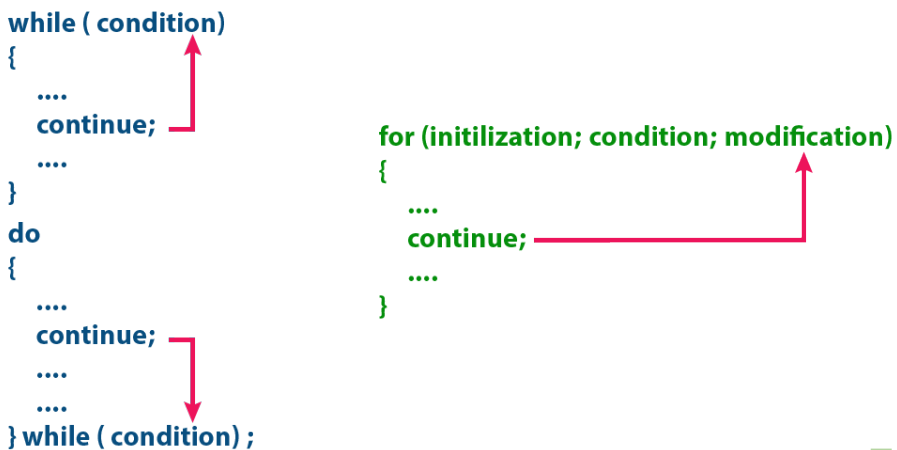


"continue" Statement

The 'continue' statement can be used to skip the current iteration of a loop and move to the next one.

Example:

```
for (int i = 0; i < 10; i++)
{
    if (i == 5)
    {
        continue; // skip iteration when i equals 5
    }
    printf("The variable i is %d\n", i);
}
```



For Loop vs. While Loop

The For loop is a specific case of the While loop (in cases where the number of iterations is known and fixed). Everything that can be written with For can be replaced by a While loop (the reverse is not necessarily true).

Example:

```
for (int i = 0; i < 10; i++)
{
    if (i == 5)
    {
        continue; // skip iteration when i equals 5
    }
    printf("The variable i is %d\n", i);
}
```

Nested Loops

The block of instructions in one loop can itself contain another loop. These are called nested loops.

Example:

```
int i;
int j=1;
for (i = 1 ; i <= 3 ; i++)
{
   j=1
   while (j <= 4)
   {
    printf("i=%d and j=%d!\n", i, j);
    j++;
   }
}
```

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Nested Loops

Execution History

Inst.	i	j	Affichage
1	1	1	i=1 et j=1
2	1	2	i=1 et j=2
3	1	3	i=1 et j=3
4	1	4	i=1 et j=4
5	2	1	i=2 et j=1
6	2	2	i=2 et j=2
7	2	3	i=2 et j=3
8	2	4	i=2 et j=4
9	3	1	i=3 et j=1
10	3	2	i=3 et j=2
11	3	3	i=3 et j=3
12	3	4	i=3 et j=4

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Which Loop Should I Use for My Program?

The choice between these loops depends on the specific problem you are solving and the control flow you want in your program. If you are unsure which loop to use, consider the following questions:

- Do you know the exact number of iterations in advance? If yes, use a for loop.
- Do you want to repeat a block of code as long as a condition is true? If yes, use a while loop.
- Do you want to ensure that a block of code is executed at least once, even if the condition is initially false? If yes, use a do-while loop.

Ultimately, the best loop for your C program depends on your program's specific logic and requirements.

Examples to Conclude ...

Example with While Loop:

```
int N;          /* number of data */
int NOMB;       /* current number     */
int I;          /* counter */
long SUM;       /* sum    of entered numbers */
double PROD; /* product of entered numbers */
printf("Number of data: ");
scanf("%d", &N);
SUM = 0; PROD = 1; I = 1;
while (I <= N)
    {
     printf("%d. number: ", I);
     scanf("%d", &NOMB);
     SUM += NOMB;
     PROD *= NOMB;
     I++;
    }
printf("The sum of %d numbers is %ld \n", N, SUM);
printf("The product of %d numbers is %.0f\n", N, PROD);
```

In Conclusion

- Loops allow you to repeat instructions a specified number of times or until a condition is met.
- There are three common loop types in C: ‘while‘, ‘do...while‘, and ‘for‘.
- Use ‘break‘ to exit a loop prematurely and ‘continue‘ to skip the current iteration.
- Make sure to avoid infinite loops by ensuring the condition eventually becomes false.