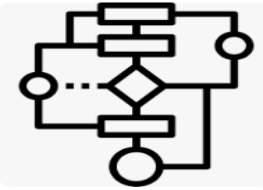
Algorithms and Data Structure 01

Dr. Sabri Ghazi
sabri.ghazi@univ-annaba.dz
Computer Science Department
Badji Mokhtar Annaba University

Chapter 04 Loops



In many cases, we require a **block** of **instructions** to be executed multiple times.
In algorithms, this is achieved using **loops**.
A **loop** is an instruction that **determines** how **many times** a block of instructions will be executed.

Loop

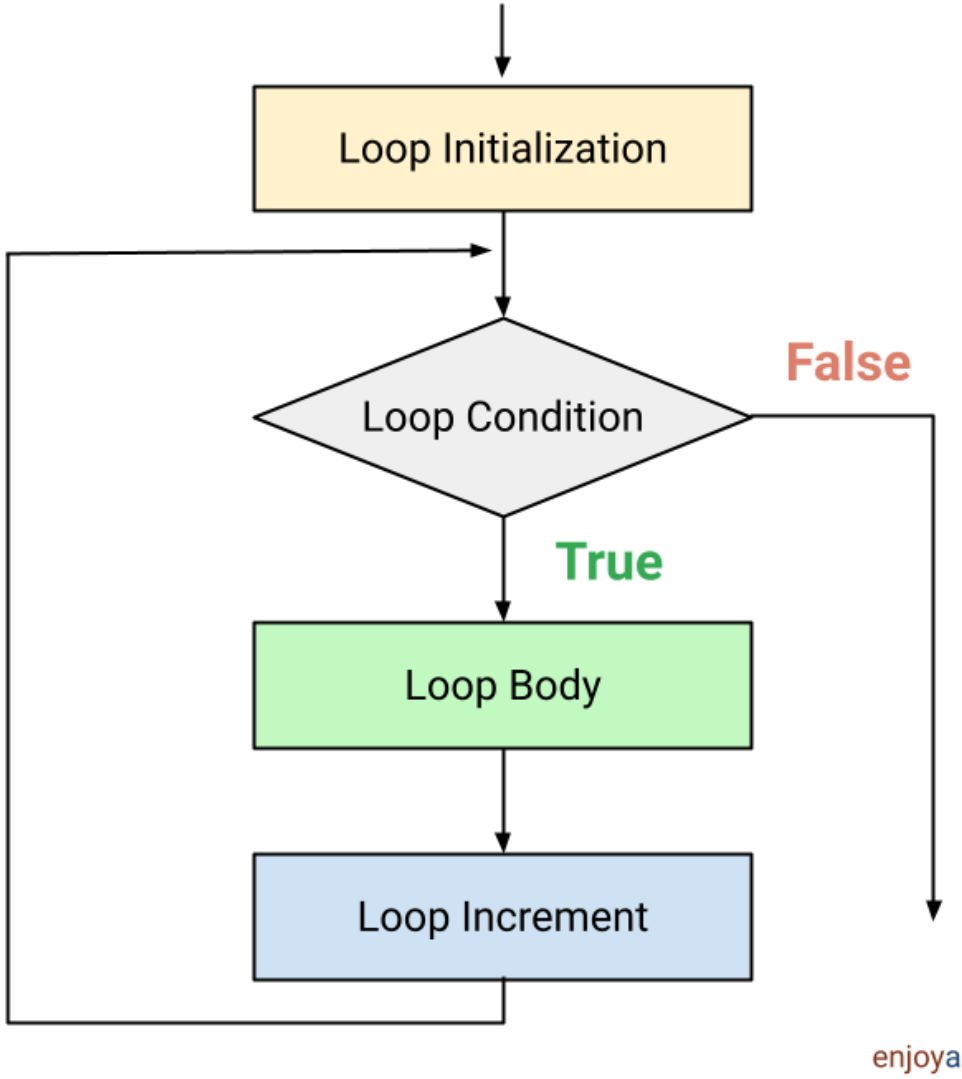
- A loop statement has three main parts :
  - **Initialization**: In this step, we assign values to the **variables** used in the **condition**.
  - **Condition**: At each **iteration**, this **condition** is verified. If it **remains true**, the loop will continue to run. If it becomes **false**, the **loop** will **end**.
  - **Block of Instructions**: The crucial aspect of this block is that it contains instructions that **modify** the **variables** used in the **condition**. Without these modifications, the loop may run **indefinitely**.

Three type of Loop

- In Algorithm there is three types of loop statement:
  - The “**While**“ statement: in which the condition is verified at the start of each iteration.
  - The **do While** statement, the condition is verified at the end of each iteration.
  - The “**for**” statement, which include how the variable are initilized, incimented and the condition.

The **for** Loop

```
Algorithm Loop1;
var
 i:int;
Begin
  For ( i←0; i<10; i++ )
     Begin
        print("Hello ", i);
     End
End
```



for loop visualization

The **for** Loop in C

```c
#include<stdio.h>
int main(){
    int i;
    for(i=0;i<5;i++){
        printf("Hello %d",i);
    }
}
```

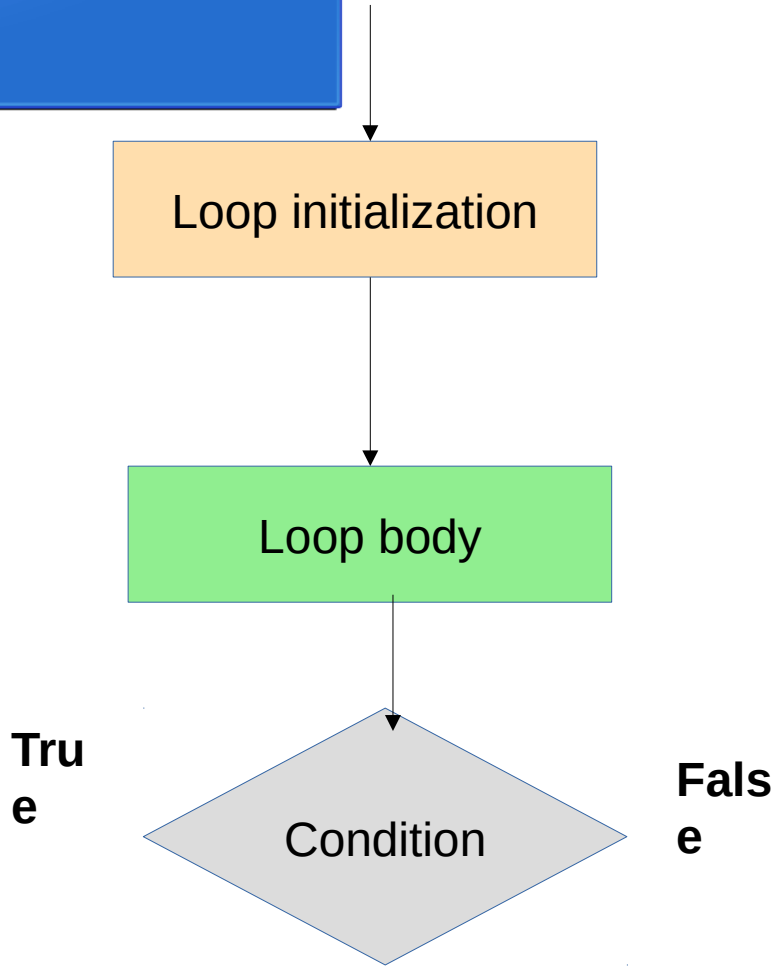
Loop variable initialisation

The condition

The progress of the loop.

The **do While** Loop

```
Algorithm Loop1;
var
 i:int;
Begin
  i← 0 ;
do
     Begin
        print("Hello ", i);
         i←i+1;
     End
 While ( i< 10 ) ;
End
```



The **do While** Loop

```
1   #include<stdio.h>
2   int main(){
3       int i;
4       i=0;
5       do{
6           printf("Hello %d",i);
7           i=i+1;
8       }
9       while(i<5);
10  }
```

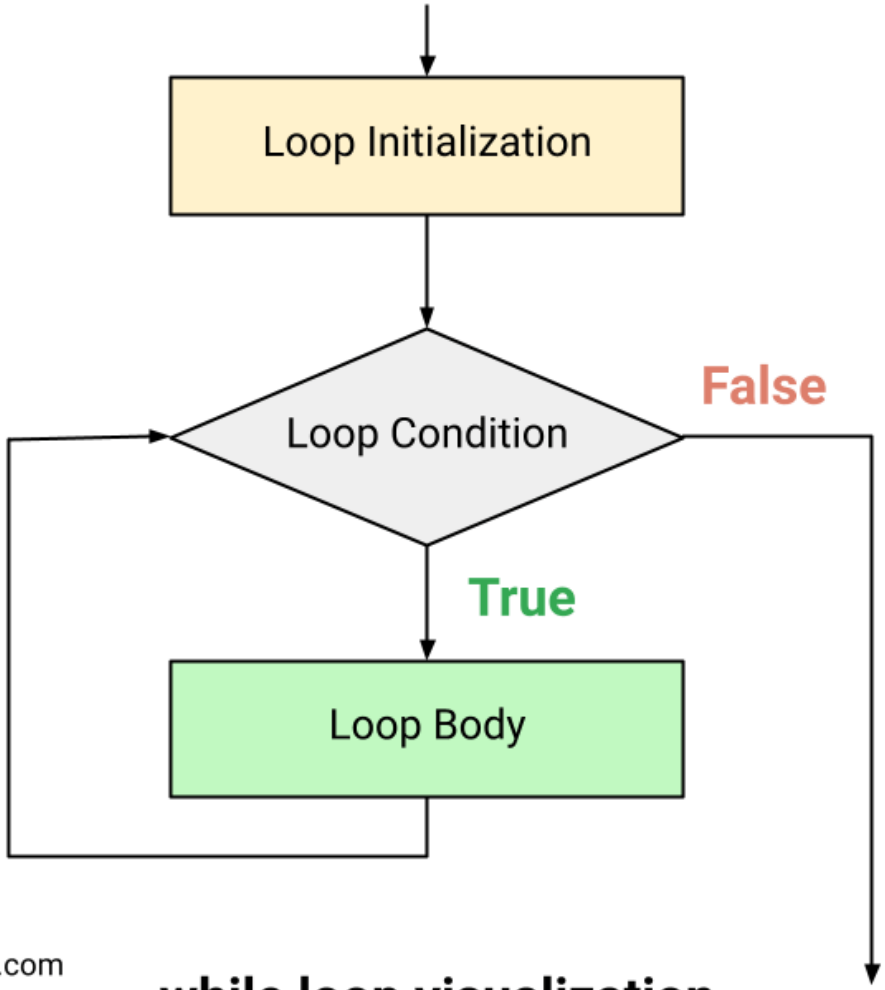
Loop variable initialisation

The progress of the loop.

The condition

The **while** Loop

```
Algorithm Loop1;
var
 i:int;
Begin
  i← 0 ;
  While ( i< 10 )
     Begin
        print("Hello ", i);
         i←i+1;
     End
End
```



hms.com

while loop visualization

The **while** Loop

```
Algorithm Loop1;
var
 i:int;
Begin
  i← 0 ;
  While ( i< 10 )
     Begin
        print("Hello ", i);
         i←i+1;
     End
End
```

Loop variable initialisation

The condition

The progress of the loop.

The **while** Loop

```
1   #include<stdio.h>
2   int main(){
3       int i;
4       i=0;
5       while(i<5){
6           printf("Hello %d",i)
7           i=i+1;
8       }
9   }
```

Loop variable initialisation

The condition

The progress of the loop.

Loops demo

Write the multiplication table of 9 using the three types of loops:

-FOR

- WHILE

- DO WHILE

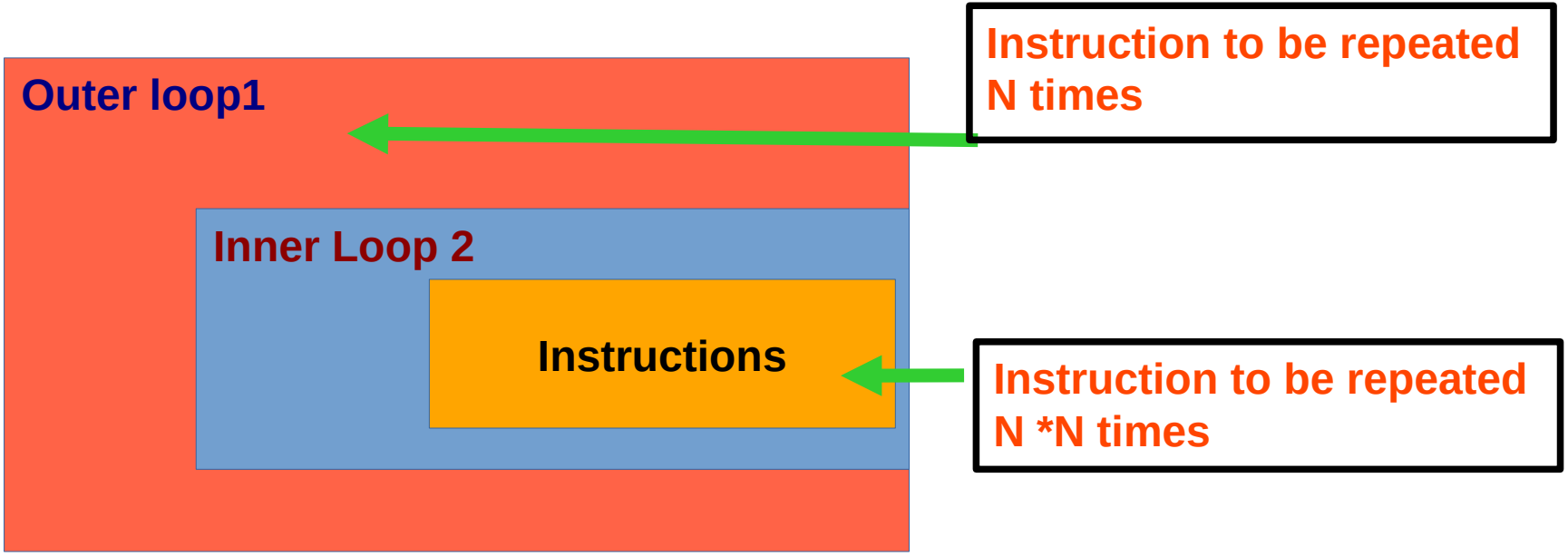
-Write an algorihtm that display the **odd** numbers from **0 to 20**

Write the multiplication table of the numbers from 0 to 9 using the three types of **loops**.

Nested Loops

Loops can help us repeat a block of instruction, but in some cases we need that the whole block of the instructions including the loop to be repeated.

In the case we are aiming to compute the multiplication table of all number from 1 to 10, in this case we need to put a loop inside another loop.



Nested Loops : with while statement

```
1   #include<stdio.h>
2   int main(){
3       int i,j;
4       i=0;
5       while ( i< 10){
6           j=0;
7           while(j<10){
8               print("Iteration # %d %d",j,i);
9               j=j+1;
10          }
11          i=i+1;
12      }
13  }
```

Outer Loop

Inner Loop

Nested Loops : with for statement

RUN AND DEBUG

VARIABLES

Locals

i: 0

j: 0

Registers

WATCH

CALL STACK Paused on breakpo...

main() text.c 4:1

text.c

home > user > Desktop > text.c > main()

```c
#include<stdio.h>
int main(){
    int i,j;
    for (i=0; i< 3;i++){
        for(j=0;j<3;j++){
            printf("Iteration # %d %d\n",j,i);
        }
    }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Nested Loops

Elements about Nested loops :

- There is many problem in which we need nested loops fro example:
  - Matrix.
  - Two variables in a functions or series.
- Order of Nesting: Determine the order in which the loops are nested. This order can significantly affect the algorithm's behavior and performance. Consider whether you need to iterate through the data row by row, column by column.
- Nested loops increase the complexity of an algorithm: Nested loops can lead to a significant increase in the number of iterations, which may impact performance.
- We can include three loops inside each others!

Nested Loops

Demo #2
Wite a program that ask the user to introduce a number , then display the bellow ilustrated pattern.(hint use two nested loops, and “\n” to return to new line)

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