Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Matter Science and Computer Science
Department of Mathematics

Chapter : 4

Loops (in algorithmic language and in C)

MI-L1-UEF121 : Algorithms and Data Structures I

Ali Khalfi

Khalfiali.udbkm@gmail.com

Course Topics

1. Introduction
2. « for » loop
3. « while » loop
4. « repeat ... until » loop
5. Nested loops

1. Introduction

Problem

- ✓ Write an algorithm that displays the multiplication table of an integer between 1 and 10

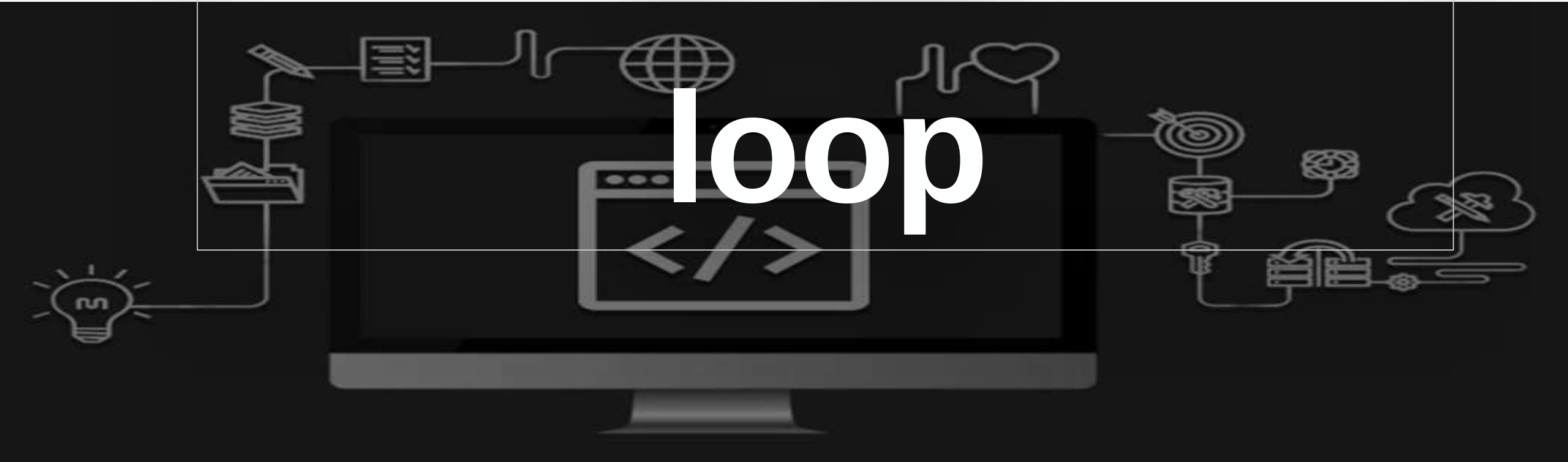
Definition

- ✓ A **repetitive structure** is a structure that repeats the same processing as many times as desired depending on an **execution condition**.
- ✓ A **Repetitive Structure** is also called **Iterative Structure** or a **Loop**.
- ✓ A loop consists of four essential elements :
  - ➢ A **statement block**, which will be executed a certain number of times;
  - ➢ A **condition/test**, which concerns at least one so-called ***loop variable***.
  - ➢ An **initialization** of the ***loop variable***.
  - ➢ An **update/modification** to the ***loop variable*** to stop the loop.

Definition

- ✓ There are 3 forms of **repetitive structures** (loops)
  1. The **FOR** loop
  2. The **WHILE** loop
  3. The **REPEAT** loop
- ✓ These structures have the same power; but by convention the choice of a loop depends essentially on the nature of the problem to be solved
- ✓ **Infinite loop**: is a loop whose condition never changes, causing an infinite number of repetitions.
- ✓ **Iteration**: it is a complete loop cycle including the loop block, the condition test, and also the modification.

2. for loop



the FOR loop

- ✓ the FOR loop is a repetitive structure that iterates the same processing for a ***range of values*** between a lower bound and an upper bound. the FOR loop is used when the number of repetitions is ***known*** in advance

Syntax

```
for  Loop Variable ← initial Value  to  Final value  do
                   Bloc
endfor
...
```

Running for loop

1st Stage : **Initialization** of the Loop Variable with the Initial Value.

2nd Stage : Evaluate the **test/condition** and check whether the Loop Variable value is in range or not.

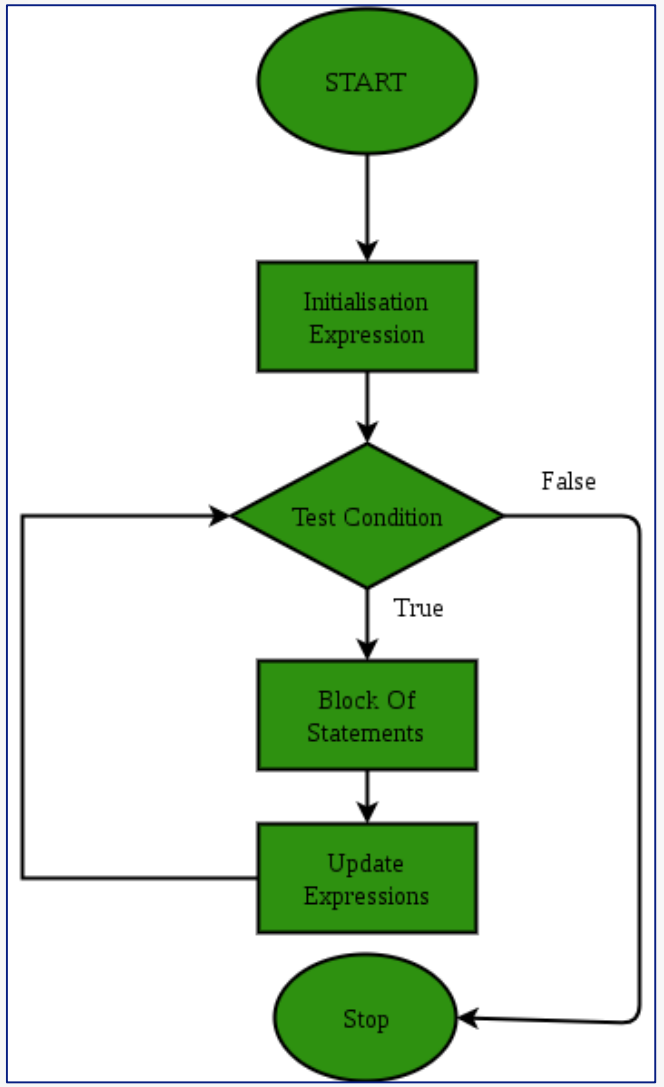
- If it is in the interval then go to the ***3rd Stage***, otherwise go to the ***5th Stage***.

3rd Stage : Execution of the loop **statement block**.

4th Stage : Automatic **update/modification** (Increment) of the value of the Loop Variable according to the value of the Step (default: Step = 1) and return to the ***2nd Stage***.

5th step: Exit from the loop and continue execution of the program starting from the first instruction which comes after ***endfor***.

for loop Flow Diagram



Example1 : Multiplication table of a number

Multiplication table

Write an algorithm which asks the user for an integer N (1 < N < 10), and which then writes the multiplication table of this number

Analyse :

```
Algorithm table_multiple;

Var N, i : integer ;

begin
    read (N);

    for i ← 1 to 10 do

        write (N, 'x', i, ' = ', N*i)

    endfor

end.
```

Example2 : Power

Power

Write an algorithm that calculates a^b such that a and b are two positive integers given by the user.

Analyse :

```
Algorithm  power;

Var  a, b : integer ;
     p, i : integer;
begin
    read (a, b);
    p ← 1;
    for  ← 1 to b do

        p ← p * a;

    endfor

    write (a,‘ power’, b, ‘=’ ,p)

end.
```

the FOR loop

Declaration

Algorithm

Syntax:

FOR compt **:=** val_initial **TO** val_final **DO**
Begin ... End

C

Syntax:

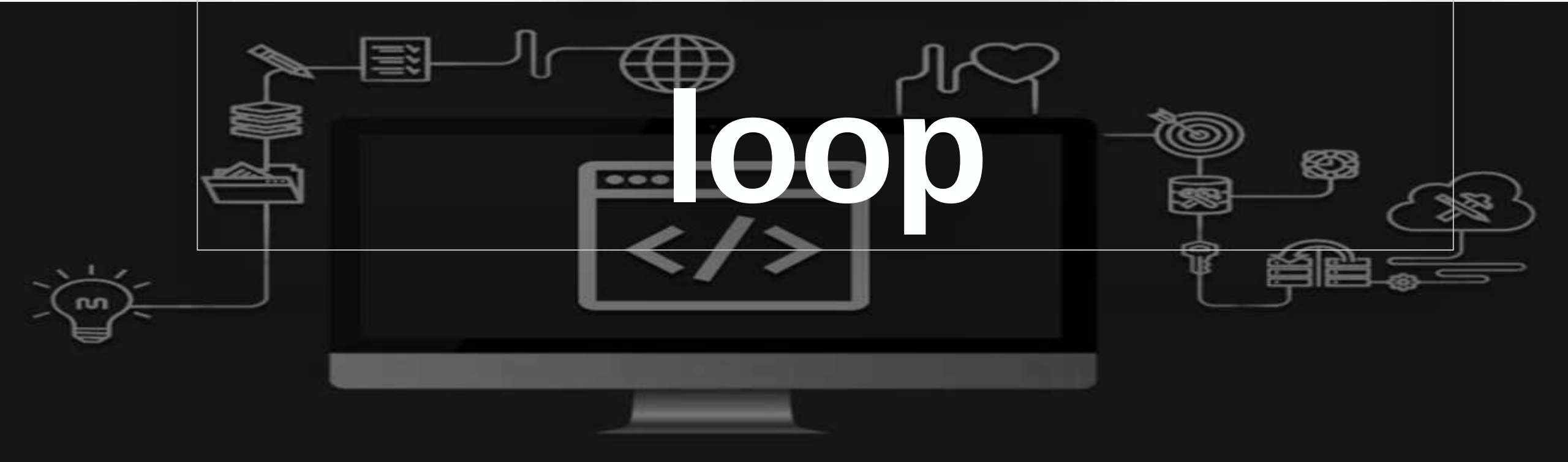
for (initialization**;** condition**;** increment**)** **{**
... **}**

Examples

```
program Example_for;

var
         a, b, p, i : Integer;

begin
    Read(a, b);    // input
    p := 1;          // initialization

    For i:= 1 to b do
      begin
        p := p * a;
      end;

    Write('The power  = ', p);

end.
```

```
#include <stdio.h>

int main (){

    int a, b, i; // declaration

    int p = 1; // declaration + initialization
    scanf("%d %d", &a, &b);

    for (i = 1; i <= b; i++)
    {
        p = p * a;
    }

    printf(« the power = %d", p);

    return 0;
}
```

3. while loop



the WHILE loop

- ✓ the WHILE loop executes the body of the loop when the execution ***condition*** is **met**; we will stop as soon as the condition is no longer verified.

Syntax

```
loop variable initialization
while   execution test/condition   do
                    Statements Block
    update loop variable
endwhile
...
```

Running While

1st step: **Initialization** of the Loop Variable before the loop.

Step 2: Evaluate and test the **execution condition**.

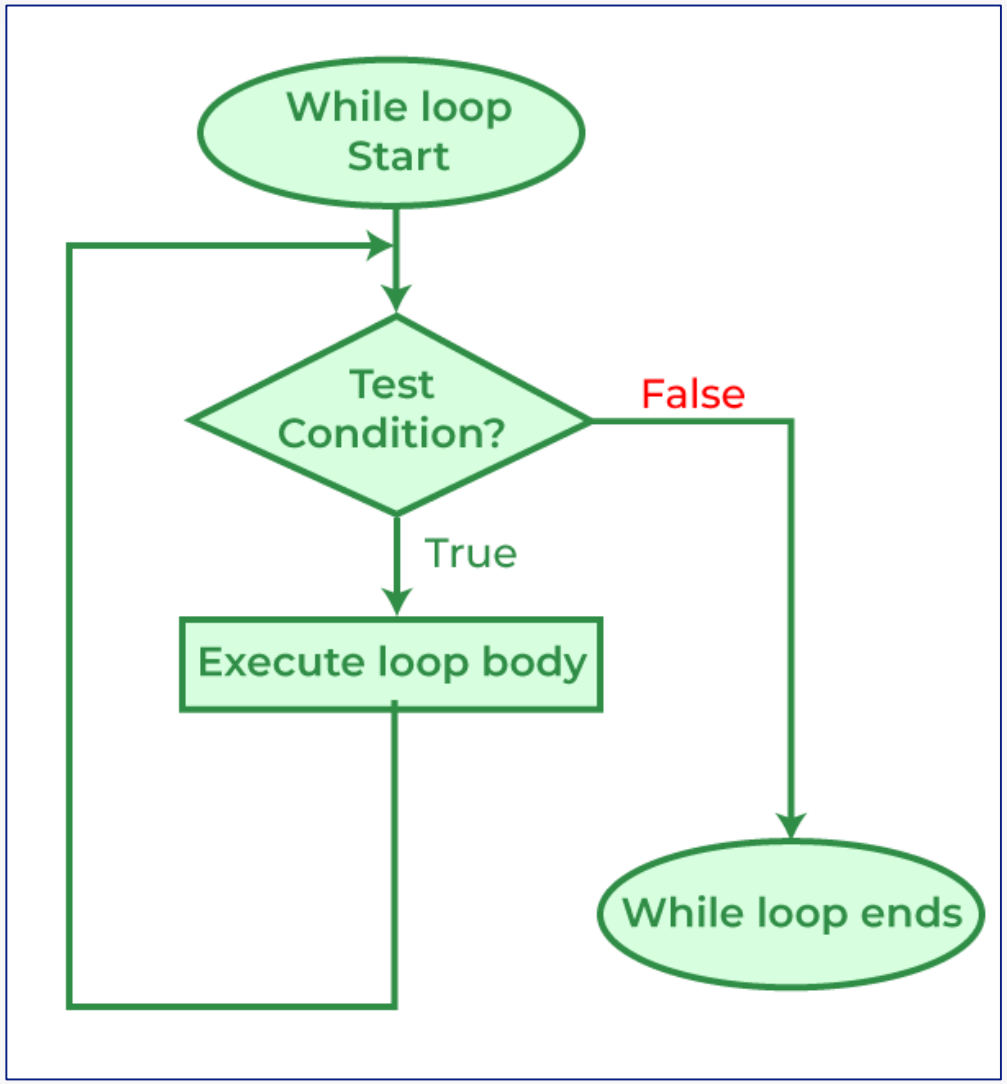
If it is verified then go to the ***3rd step***, otherwise go to the ***5th step***.

3rd step: Execution of the loop **statements block.**

4th step: Changing (**updating**) the value of the Loop Variable and returning to the ***2nd step***.

5th step: Exit from the loop and continue execution of the program starting from the first instruction which comes after ***endwhile***.

while loop Flowchart



Example1 : Multiplication table of a number

Multiplication table

Write an algorithm which asks the user for an integer N (1 < N < 10), and which then writes the multiplication table of this number

Analyse :

```
Algorithm  table_multiplication;

Var  N, i, p: integer ;

begin
    read (N);
     i ← 1;
     while i <= 10 do
          p ← N * i
          write (N, 'x', i, ' = ', p)
          i ← i + 1
    endwhile

end.
```

Exemple2 : GCD calculation

GCD calculation

Write an algorithm that calculates the GCD of two integers a and b by applying the recurrence relation GCD(a,b) = GCD(b, a MOD b) until the remainder of the division of a by b is zero .

Analyze :

PGCD(18,12) = PGCD(12, 6) = PGCD(6, 0) = 6

```
Algorithm calcul_pgcd;

Var a, b, r: integer ;

begin
    read (a,b);

    while b <> 0 do
        r ← MOD b
        a ← b
        b ← r
    endwhile
    write ('the GCD is of a and b =', a)
end.
```

the WHILE loop

Declaration

Examples

Algorithm

Syntax:

WHILE condition **DO**
    Begin ... End

```
program Example_while;

var
        a, b, r : Integer;

begin
    Read(a, b);    // Input

    While b <> 0 do
      begin
        r := a mod b;
        a := b;
        b := r;
      end;

    Write('The GCD of a and  b is ', a);

end.
```

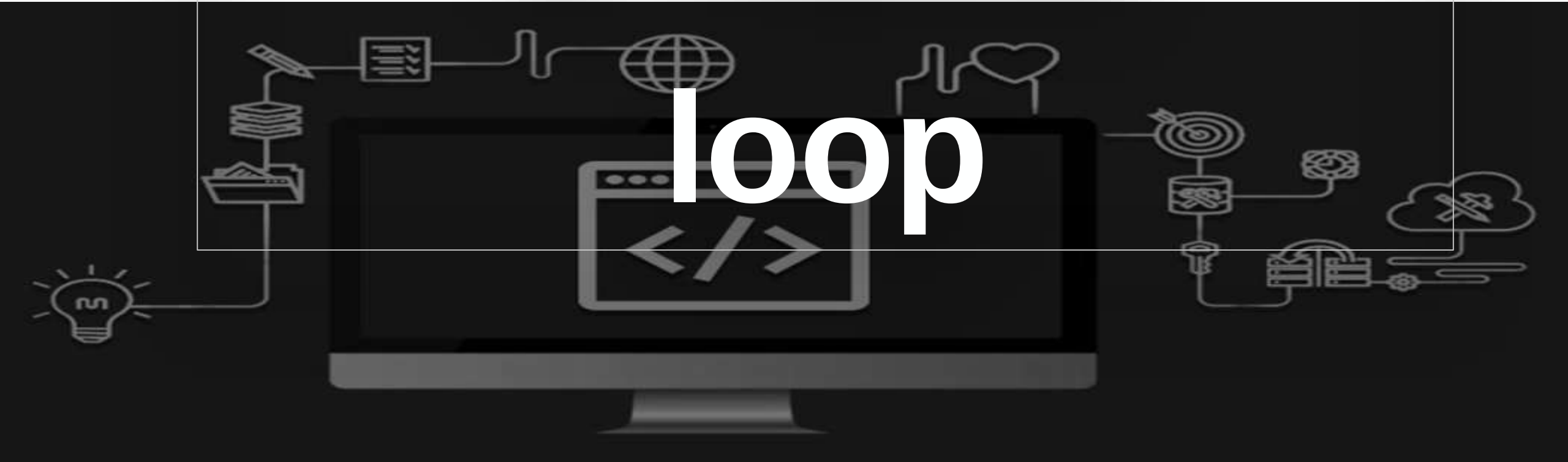
C

Syntax:

while (condition**) { ... }**

```
#include <stdio.h>

int main (){

    int a, b, r; // declaration
    scanf("%d %d",&a, &b);

    while (b != 0)
    {
        r = a%b;
        a = b;
        b = r;
    }

    printf(" The GCD of a and  b is  %d", a);

    return 0;
}
```

4. repeat loop



the REPEAT loop

- ✓ The Repeat loop allows you to enter the loop regardless of the condition and repeats the execution until the condition is true.

Syntax

```
Repeat
        Block of statements
    update loop variable
until  stop condition
...
```

Running REPEAT

Step 1: Go inside the “Repeat” loop and execute the associated **block**.

Step 2: Update the variables involved in the **stopping condition**.

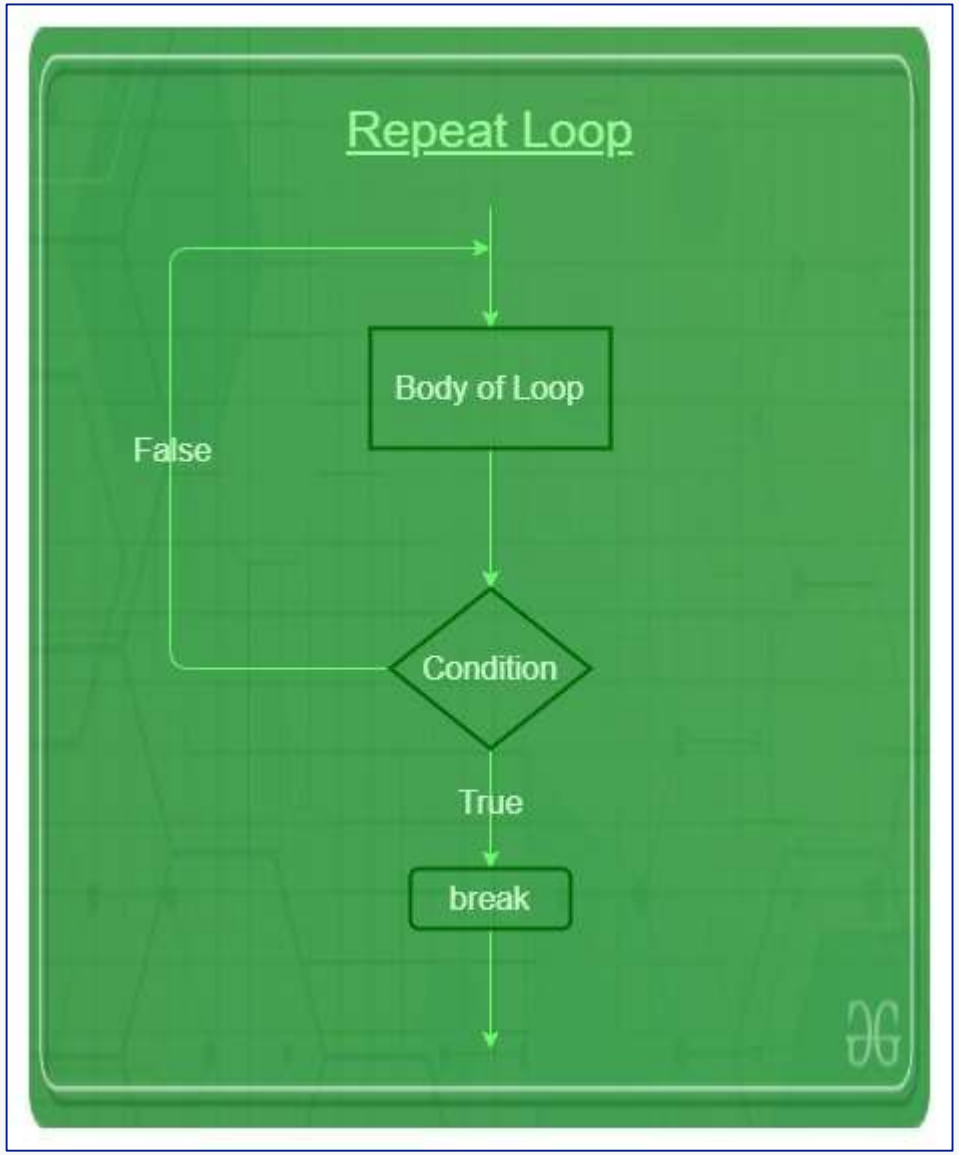
Step 3: Evaluate and test the **stopping condition**.

- If it is checked then go to the ***4th step***, otherwise go back to the ***1st step***.

Step 4 : Changing (**updating**) the value of the Loop Variable and returning to the ***2nd step***.

Step 5 : the **stopping condition** having been reached, we exit the Repeat loop and continue the execution of the program from the first instruction which comes after Until.

REPEAT loop Flowchart



Example1 : Multiplication Table

Multiplication table

Write an algorithm which asks the user for an integer N (1 < N < 10), and which then writes the multiplication table of this number

Analyse :

```
Algorithm table_multiplication;

Var  N, i, p: integer ;

begin
    read (N);
     i ← 1;
     Repeat
          p ← N * i
          write (N, 'x', i, ' = ', p)
          i ← i + 1
     Until  (i > 10)

end.
```

Example2 : GCD calculation

GCD calculation

Write an algorithm that calculates the GCD of two integers a and b by applying the recurrence relation GCD(a,b) = GCD(b, a MOD b) until the remainder of the division of a by b is zero .

Analyse :

PGCD(18,12) = PGCD(12, 6) = PGCD(6, 0) = 6

```
Algorithm calcul_pgcd;

Var a, b, r: integer ;

begin
    read (a,b);

    Repeat
        r ← a MOD b
        a ← b
        b ← r
    Until (b = 0)
    write ('Le PCGD de a et b =', a)
end.
```

the REPEAT loop

Declaration

Algorithm

Syntax:

REPEAT bloc **UNTIL** (condition)

C

Syntax:

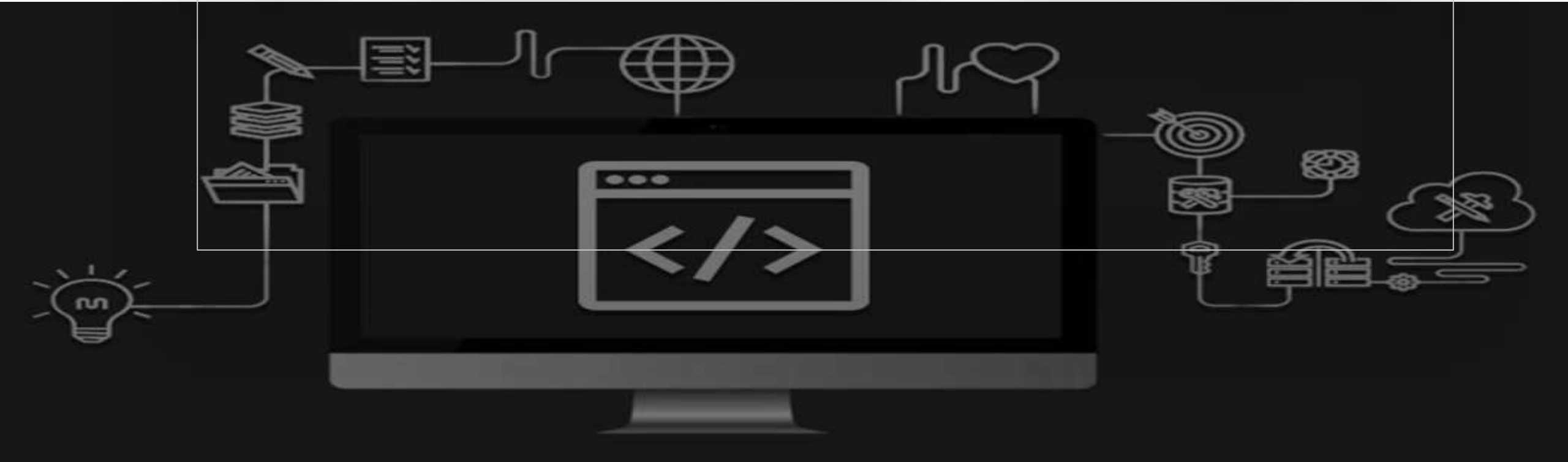
do { ... } **while** (condition)

Examples

```
program Exemple_repeter;

var
        a, b, r : Integer;

begin
    Read(a, b);   // input

    repeat
        r := a mod b;
        a := b;
        b := r;
    until (b = 0);

    Write('The GCD of a and b is  ', a);

end.
```

```
#include <stdio.h>

int main (){
    int a, b, r; // declaration
    scanf("%d %d",&a, &b);

    do
    {
        r = a%b;
        a = b;
        b = r;
    }
    while (b != 0);

    printf(« The GCD of a and b = %d", a);

    return 0;
}
```

5. Nested loops



What are Nested Loops?

A nested loop is when you have one loop inside another loop. The inner loop completes all its iterations for each single iteration of the outer loop.

the FOR loop

Declaration

Examples

Algorithm

Syntax:

FOR compt **:=** val_initial **TO** val_final **DO**
Begin ... End

```
program Example_nested_for;
var
         a : Integer;
begin
    a=15;
    For i:= 1 to 5 do
      begin
           For i:= 1 to 5 do
              begin
                write (a);
              end;

      end;

    end.
```

C

Syntax:

for (initialization**;** condition**;** increment**)** **{**
... **}**

```
#include <stdio.h>

int main (){

    int a; // declaration
    a=15;


    for (i = 1; i <= 5; i++)
    {
       for (i = 1; i <= 5; i++)
    {
        printf("%d ", a);

    }

    }

    printf(« the power = %d", p);
```

Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Matter Science and Computer Science
Department of Mathematics

Chapter : 4

Loops (in algorithmic language and in C)

MI-L1-UEF121 : Algorithms and Data Structures I

Ali Khalfi

Khalfiali.udbkm@gmail.com