Ministère de l'enseignement supérieur et de la recherche scientifique
Université Djillali BOUNAAMA - Khemis Miliana (UDBKM)
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Chapitre 4

Les Structures itératives : Boucles

MI-L1-UEF121 : Algorithmiques et Structures de Données I

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz

Plan du Cours

1. Introduction
2. La boucle « Pour »
3. La boucle « Tant que »
4. La boucle « répéter »
5. Les boucles imbriquées

Introduction

Problématique

- ✓ Ecrire un algorithme qui affiche la table de multiplication d'un entier entre 1 et 10

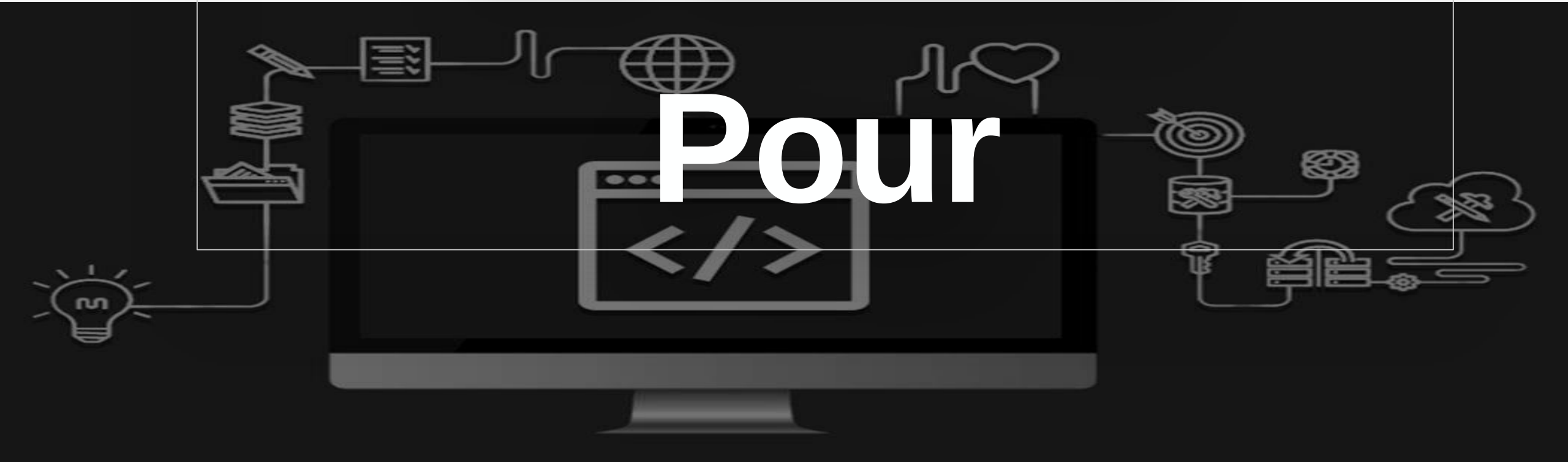
Définition

- ✓ Une **structure répétitive** est une structure qui répète un même traitement autant de fois que l'on veut en fonction d'une **condition** d'exécution.
- ✓ Une **Structure Répétitive** est aussi appelée **Structure Itérative** ou encore la **Boucle.**
- ✓ Une boucle se compose de quatre éléments essentiels :
  - ➢ Un **bloc** d'instruction, qui sera exécuté un certain nombre de fois ;
  - ➢ Une **condition**, qui porte sur au moins une ***variable*** dite de ***boucle***.
  - ➢ Une **initialisation** de la ***variable de boucle***.
  - ➢ Une **modification** de la ***variable de boucle*** pour arrêter la boucle.

Définition

- ✓ Il existe 3 formes de **structures** (boucles) **répétitives**
  1. La boucle **POUR**
  2. La boucle **TANT QUE**
  3. La boucle **REPETER**
- ✓ C'est structures ont le même pouvoir; mais par convention le choix d'une boucle dépend essentiellement de la nature du problème à résoudre
- ✓ **Boucle infinie** : est une boucle dont la condition ne change jamais, ce qui provoque un nombre infini de répétitions.
- ✓ **Itération** : est la réalisation d'un cycle complet de boucle en incluant le bloc de la boucle, le test de condition, et aussi la modification.

La boucle Pour



La boucle POUR

- ✓ La boucle **Pour** est une structure répétitive qui itère le même traitement pour une ***plage de valeurs*** comprises entre une borne inférieure et une borne supérieure.
- ✓ La boucle **Pour** est utilisée lorsque le nombre répétitions est ***connu*** à l'avance

Syntaxe

```
Pour  Variable de boucle  de  Valeur initiale  à  Valeur finale  Faire
                          Bloc
Fpour
...
```

Exécution de POUR

1ère étape : **Initialisation** de la *Variable de boucle* avec la *Valeur initiale* .

2ème étape : évaluer la **condition** et vérifier si la valeur de la *Variable de boucle* est dans l'intervalle ou non.

- Si elle est dans l'intervalle alors aller à la ***3ème étape***, sinon aller à la ***5ème étape***.

3ème étape : Exécution du **bloc** de la boucle.

4ème étape : **Modification** (Incrémentation) automatique de la valeur de la *Variable de boucle* en fonction de la valeur du Pas (par défaut : Pas = 1) et retour à la ***2ème étape.***

5ème étape : Sortie de la boucle et suite de l'exécution du programme à partir de la première instruction qui vient après Fpour.

Exemple1 : Table de multiplication d'un nombre

Table de multiplication

Ecrire un algorithme qui demande un nombre entier N à l'utilisateur (1 < N < 10), et qui ensuite écrit la table de multiplication de ce nombre

Analyse :

```
Algorithme table_multiple;

Var N, i, p: entier ;

Début
    Lire (N);

    Pour i de 1 à 10 Faire
        p ← N * i
        Ecrire (N, 'x', i, ' = ', p)

    Fpour

Fin.
```

Exemple2 : la puissance

La puissance

Ecrire un algorithme calcule a^b tel que a et b deux entier positif donnés par l'utilisateur.

Analyse :

```
Algorithme puissance;

Var a, b : entier ;
    p, i : entier;
Début
    Lire (a, b);
    p ← 1;
    Pour i de 1 à b Faire

        p ← p * a;

    Fpour

    Ecrire ('la puissance =', p)

Fin.
```

La boucle POUR

Structure itératives

Déclaration

PASCAL

Syntaxe:

FOR compt := val_initial TO val_final DO
Begin … End

C

Syntaxe:

for (initialisation; condition; modification)
{ … }

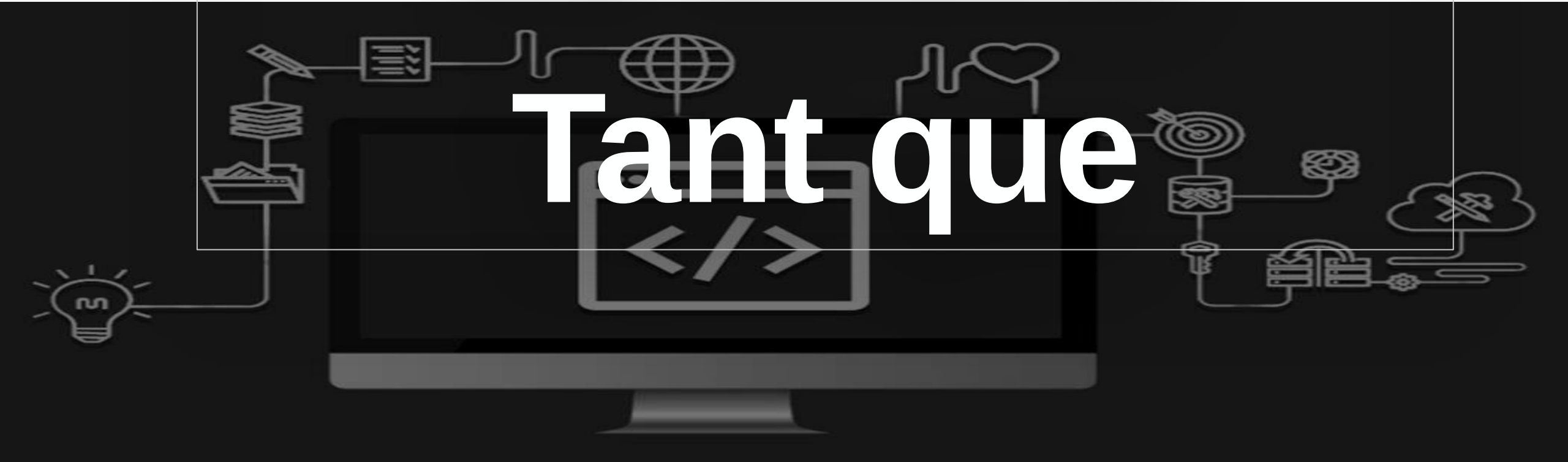
Exemples

```pascal
program Exemple_pour;

var
        a, b, p, i : Integer;

begin
    ReadLn(a, b);   // lecture
    p := 1;         // initialisation

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

    WriteLn('La puissance = ', p);

end.
```

```c
#include <stdio.h>

int main (){

    int a, b, i; // déclaration

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

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

    printf("La puissance = %d", p);

    return 0;
}
```

La boucle

Tant que



La boucle TANT QUE

- ✓ La boucle **Tant que** permet d'exécuter le corps de la boucle lorsque la ***condition*** d'exécution est ***vérifiée*** ; on s'arrêtera dès que la condition n'est plus vérifiée..

Syntaxe

```
initialisation Variable de boucle
TQ  condition d'exécution  Faire
                Bloc
        modification Variable de boucle
Ftq
...
```

Exécution de TANT QUE

1ère étape : **Initialisation** de la *Variable de boucle* avant la boucle.

2ème étape : évaluer et tester la **condition d'exécution** .

- Si elle est vérifiée alors aller à la ***3ème étape***, sinon aller à la ***5ème étape***.

3ème étape : Exécution du **bloc** de la boucle.

4ème étape : **Modification** (mise à jour) de la valeur de la *Variable de boucle* et retour à la ***2ème étape.***

5ème étape : Sortie de la boucle et suite de l'exécution du programme à partir de la première instruction qui vient après Ftq.

Exemple1 : Table de multiplication

Table de multiplication

Ecrire un algorithme qui demande un nombre entier N à l'utilisateur (1 < N < 10), et qui ensuite écrit la table de multiplication de ce nombre

Analyse :

```
Algorithme table_multiple;

Var N, i, p: entier ;

Début
    Lire (N);
    i ← 1;
    TQ i <= 10 Faire
        p ← N * i
        Ecrire (N, 'x', i, ' = ', p)
        i ← i + 1
    Ftq

Fin.
```

Exemple2 : Calcul de PGCD

Table de multiplication

Ecrire un algorithme qui calcul le PGCD de deux entier a et b en appliquant la relation de récurrence PGCD(a,b) = PGCD(b, a MOD b) jusqu'à ce que le reste de la division de a sur b soit nul.

Analyse :

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

```
Algorithme calcul_pgcd;

Var a, b, r: entier ;

Début
    Lire (a,b);

    TQ b <> 0 Faire
        r ← a MOD b
        a ← b
        b ← r
    Fpour
    Ecrire ('Le PCGD de a et b =', a)
Fin.
```

La boucle TANT QUE

Structure itératives

Déclaration

PASCAL

Syntaxe:

WHILE condition **DO**
    Begin … End

C

Syntaxe:

while (condition**) { … }**

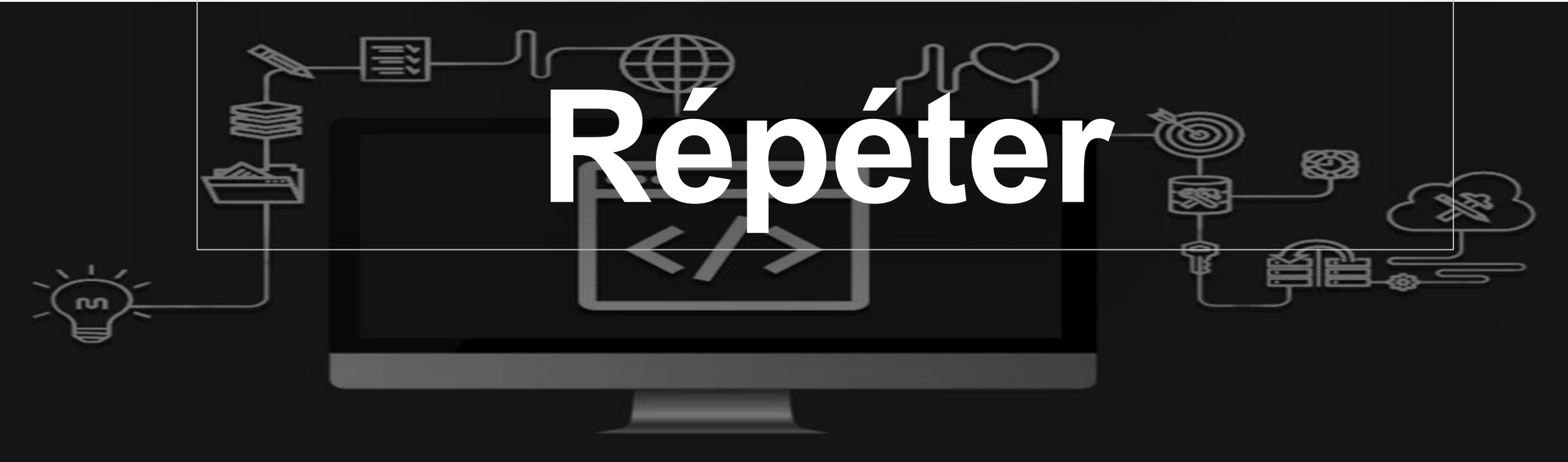
Exemples

```pascal
program Exemple_tq;

var
        a, b, r : Integer;

begin
    ReadLn(a, b);   // lecture

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

    WriteLn('Le PGCD de a et b est ', a);

end.
```

```c
#include <stdio.h>

int main (){

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

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

    printf("Le PGCD de a et b = %d", a);

    return 0;
}
```

La boucle

Répéter



La boucle RÉPÉTER

- ✓ La boucle **Répéter** permet de rentrer dans la boucle quelque soit la condition et réitère l'exécution ***jusqu'à ce*** que la ***condition*** soit vérifiée.

Syntaxe

```
Répéter
        Bloc
        modification Variable de boucle
Jusqu'à  condition d'arrêt
...
```

Exécution de RÉPÉTER

1ère étape : Passer à l'intérieur de la boucle « Répéter » et exécuter le **bloc** associé.

2ème étape : mettre à jour les variables impliquées dans la **condition d'arrêt**.

3ème étape : évaluer et tester la **condition d'arrêt**.

- Si elle est vérifiée alors aller à la ***4ème étape***, sinon revenir à la ***1ère étape***.

4ème étape : **Modification** (mise à jour) de la valeur de la *Variable de boucle* et retour à la ***2ème étape*.**

5ème étape : la **condition d'arrêt** étant atteinte, on sort de la boucle Répéter et continuer l'exécution du programme à partir de la première instruction qui vient après Jusqu'à.

Exemple1 : Table de multiplication

Table de multiplication

Ecrire un algorithme qui demande un nombre entier N à l'utilisateur (1 < N < 10), et qui ensuite écrit la table de multiplication de ce nombre

Analyse :

```
Algorithme table_multiple;

Var N, i, p: entier ;

Début
    Lire (N);
    i ← 1;
    Répéter
        p ← N * i
        Ecrire (N, 'x', i, ' = ', p)
        i ← i + 1
    Jusqu'à (i > 10)

Fin.
```

Exemple2 : Calcul de PGCD

Table de multiplication

Ecrire un algorithme qui calcul le PGCD de deux entier a et b en appliquant la relation de récurrence PGCD(a,b) = PGCD(b, a MOD b) jusqu'à ce que le reste de la division de a sur b soit nul.

Analyse :

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

```
Algorithme calcul_pgcd;

Var a, b, r: entier ;

Début
    Lire (a,b);

    Répéter
        r ← a MOD b
        a ← b
        b ← r
    Jusqu'à (b = 0)
    Ecrire ('Le PCGD de a et b =', a)
Fin.
```

La boucle RÉPÉTER

Structure itératives

Déclaration

PASCAL

Syntaxe:

REPEAT bloc **UNTIL** (condition)

C

Syntaxe:

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

Exemples

```pascal
program Exemple_repeter;

var
        a, b, r : Integer;

begin
    ReadLn(a, b);   // lecture

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

    WriteLn('Le PGCD de a et b est ', a);

end.
```

```c
#include <stdio.h>

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

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

    printf("Le PGCD de a et b = %d", a);

    return 0;
}
```

Ministère de l'enseignement supérieur et de la recherche scientifique
Université Djillali BOUNAAMA - Khemis Miliana (UDBKM)
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Chapitre 4

Les Structures itératives : Boucles

MI-L1-UEF121 : Algorithmiques et Structures de Données I

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz