ANNÉE SCOLAIRE 2023/2024

COURS D'ALGORITHMIQUE

Pape Abdoulaye **BARRO**
Docteur en Informatique et Télécommunications
Spécialiste en Télémétrie & Systèmes Intelligents

TABLEAUX

TABLEAUX
DÉFINITION

L'utilisateur donne 20 nombres entiers et puis notre algorithme va lui calculer la moyenne. **Comme suit**:

...
Lire N1
Lire N2
...
Lire N20
Moy←(N1+N2+...+N20)/20
...

- Le seul moyen dont nous disposons jusqu'à présent était de faire une boucle de saisie de notes et dedans, de tenter de faire les calculs au fur et à mesure.

TABLEAUX
DÉFINITION

- Maintenant, si nous savons le nombres de notes à saisir, ne serait-il pas plus simple de remplacer toutes les variables par une seule, mais qui pourrait contenir toutes les notes ?
  - L'idée serait donc d'avoir un nom de variable mais qui pourrait associer une note à un numéro.
  - **Exemple:**
    - Prenons la variable "note". Il suffirait alors de dire que "note 1 vaut 15, note 2 vaut 17, note 3 vaut 8, etc.".
- Un **ensemble de valeurs** représenté par le **même nom de variable** et où chaque valeur est identifiée par un numéro s'appelle un **tableau**.
  - Le numéro qui sert à identifier un élément (une valeur) du tableau s'appelle un indice.
  - Un élément du tableau est représenté par le nom de la variable auquel on accole l'indice entre crochets. **Exemple:** Note[numéro] .

TABLEAUX
DÉFINITION

- Un tableau est une liste d'éléments ayant le même type, désignés sous le même nom et accessibles par indices.
- Les tableaux peuvent être d'une, deux ou de plusieurs dimensions.
  - Pour un tableau à une dimension, la syntaxe est la suivante:

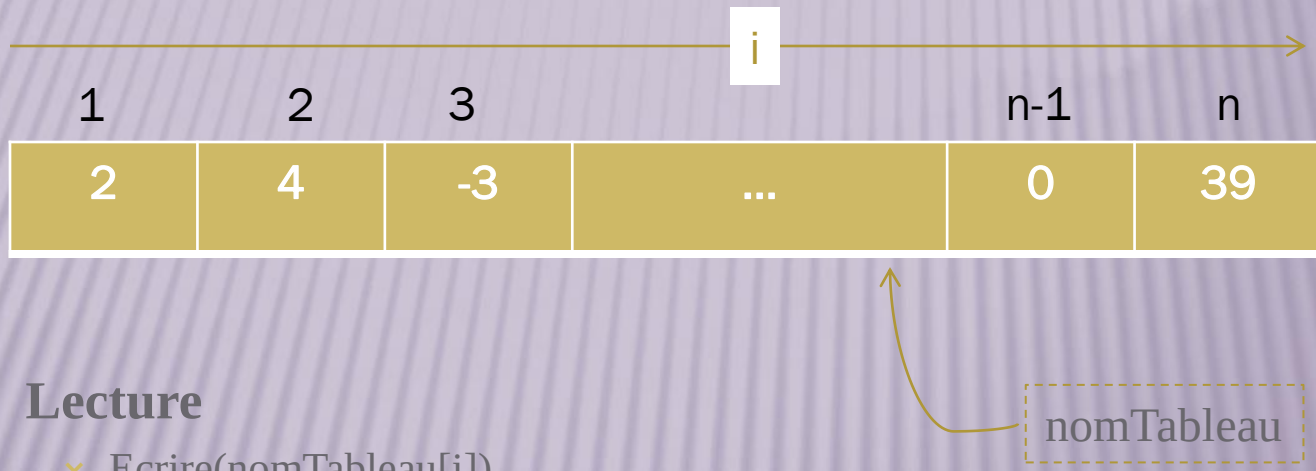
nomTableau: tableau[taille] de type

- **Exemple**:
  - Variable
    - notes: tableau[10] de réels
    - texte: tableau[255] de caractères
    - mois :tableau[12]<-{"janvier",...,"décembre"} de chaînes

TABLEAUX

TABLEAU À UNE DIMENSION>ACCÈS AUX ÉLÉMENTS

- Les éléments d'un tableau sont accessibles pas indice (commençant par 1) que cela soit en lecture ou en écriture.



- **Lecture**
  - Ecrire(nomTableau[i])
- **Ecriture**
  - Lire(nomTableau[i])
- **Affectation**
  - nomTableau[i] ← Valeur

TABLEAUX

TABLEAU À UNE DIMENSION>EXEMPLE

- **Exemple:**

```
Algorithme Parcours
Variable notes: tableau[10] de réels
                i      : entier
Début
        Ecrire("Remplir le tableau")
        Pour i allant de 1 à 10 faire
                Ecrire("Note[",i,"]= ")
                Lire(notes[i])
        FinPour

        {Affichage du tableau}
        Pour i allant de 1 à 10 faire
                Ecrire(notes[i])
        FinPour
  Fin
```

TABLEAUX

TABLEAU À UNE DIMENSION>EXEMPLE

- **Exemple:**

  Écrire un algorithme permettant de saisir 10 entiers, de les stocker dans un tableau nommé tab, de remplacer les éléments par leur carré, puis les afficher.

TABLEAUX

TABLEAU À UNE DIMENSION>EXEMPLE

- **Exemple:**
  Écrire un algorithme permettant de saisir 10 entiers, de les stocker dans un tableau nommé tab, de remplacer les éléments par leur carré, puis les afficher.

- **Solution:**

```
Algorithme tableau_dix_elements
Variable
        tab: tableau[10] d'entier
        i   : entier
Début
    Ecrire("Remplir le tableau")
    Pour i allant de 1 à 10 faire
        Ecrire("tab[",i,"]= ")
        Lire(tab[i])
    FinPour

    {Remplacer les éléments par leur carré}
    Pour i allant de 1 à 10 faire
        tab[i] ← tab[i]* tab[i]
    FinPour

    {Affichage du tableau}
    Pour i allant de 1 à 10 faire
        Ecrire(tab[i])
    FinPour
Fin
```

TABLEAUX
TABLEAUX À DEUX DIMENSIONS

- Pour un tableau à deux dimensions, la syntaxe est la suivante:

nomTableau: tableau[ligne][colonne] de type

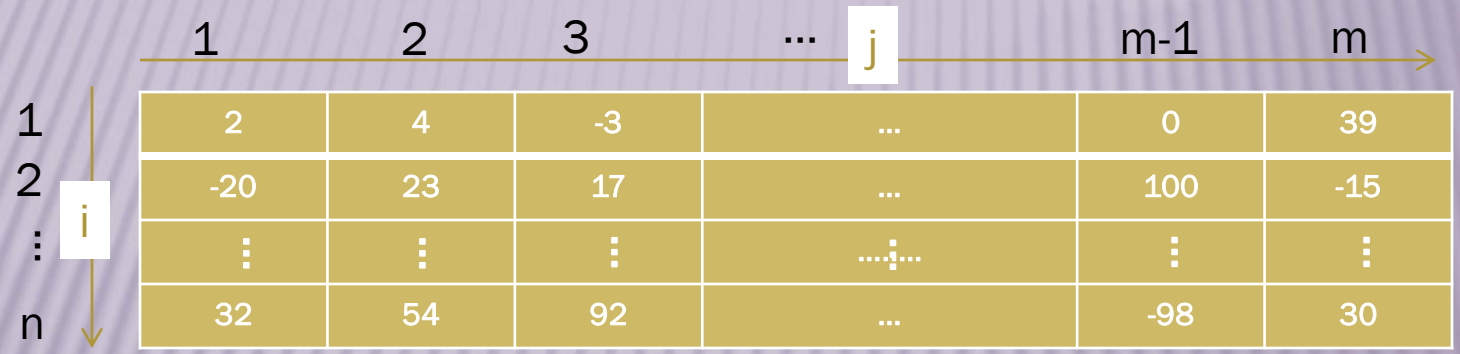
  - **Exemple**:
    - notes: tableau[10][20] de réels
    - texte: tableau[10][255] de caractères
    - matrice: tableau[3][4] de réels

  - **Remarque** : Nous pouvons utiliser autant de dimensions que souhaitées

TABLEAUX

TABLEAUX À DEUX DIMENSIONS

- Les éléments d'un tableau à deux dimensions sont accessibles pas indice ligne (commençant par 1) et colonne (commençant par 1 aussi) que cela soit en lecture ou en écriture.



2	4	-3	...	0	39
-20	23	17	...	100	-15
⋮	⋮	⋮	⋱	⋮	⋮
32	54	92	...	-98	30

nomTableau

- **Lecture**
  - Ecrire(nomTableau[i][j])
- **Ecriture**
  - Lire(nomTableau[i][j])
- **Affectation**
  - nomTableau[i][j] ← Valeur

TABLEAUX

TABLEAU À DEUX DIMENSIONS>EXEMPLE

- **Exemple:**

```
Algorithme Parcours
Variable
                notes : tableau[2][10] de réels
                  i, j   : entier
Début
        Ecrire("Remplir le tableau")
        Pour i allant de 1 à 2 faire
            Pour j allant de 1 à 10 faire
                Ecrire("Note[",i,"][",j,"]= ")
                Lire(notes[i][j])
            FinPour
        FinPour

        {Affichage du tableau}
        Pour i allant de 1 à 2 faire
            Pour j allant de 1 à 10 faire
                Ecrire(notes[i][j])
            FinPour
        FinPour
  Fin
```

TABLEAUX

TABLEAU À DEUX DIMENSIONS>EXEMPLE

- **Exemple:**

  Ecrire un algorithme qui permet de remplir une matrice M de 10 lignes et 20 colonnes. Ensuite, de remplacer les éléments par leur carré. Enfin, d'afficher le contenu du tableau.

TABLEAUX

TABLEAU À DEUX DIMENSIONS>EXEMPLE

- **Exemple:**

  Ecrire un algorithme qui permet de remplir une matrice M de 10 lignes et 20 colonnes. Ensuite, de remplacer les éléments par leur carré. Enfin, d'afficher le contenu du tableau.

- **Solution:**

```
Algorithme Factorielle
Variable M: tableau[10][20] d'entier
                i , j , val : entiers
Début
        val ← 1
        {Remplissage du tableau}
        Pour i allant de 1 à 10 faire
            Pour j allant de 1 à 20 faire
                    M[i][j] ← val
                    val ← val+1
            FinPour
        FinPour
        {Remplacer les éléments par leur carré}
        Pour i allant de 1 à 10 faire
            Pour j allant de 1 à 20 faire
                    M[i][j] ← M[i][j] * M[i][j]
            FinPour
        FinPour
        {Affichage du tableau}
        Pour i allant de 1 à 10 faire
            Pour j allant de 1 à 20 faire
                    Ecrire(M[i][j])
            FinPour
        FinPour
Fin
```

TABLEAUX
TABLEAU DYNAMIQUE

- Si nous ne connaissons pas par avance le nombre d'éléments de votre tableau, nous avons deux possibilités:
  - On peut fixer un nombre d'éléments suffisamment grand à l'avance pour être sûr d'en avoir assez.
  - Ou alors de redimensionner notre tableau à la bonne taille dès que le nombre d'éléments nous est connu.

- Il existe en **pseudo-code algorithmique** une instruction appelée "**Redim**" qui permet de redimensionner un tableau dont le nombre d'éléments n'est pas connu à l'avance.
  - Cependant, il est souvent conseillé d'éviter de l'utiliser.
  - Cette instruction trouve son utilité dans le fait qu'en pseudo-code les variables et les tableaux sont déclarés avant le début du programme, ce qui induit l'impossibilité d'initialiser le nombre d'éléments d'un tableau suivant la valeur d'une variable.

TABLEAUX
TABLEAU DYNAMIQUE

- Si nous devons utiliser des tableaux dynamiques, alors nous ne devons pas indiquer de nombres d'éléments dans la déclaration. Cela sera fait dans l'instruction de redimensionnement.

- **Exemple:**

```
Algorithme Redimensionnement
Variable
    elements :tableau[] d'entiers
    nb:entier
Debut
    Ecrire( "Combien d'éléments ?")
    Lire(nb)
    Redim elements[nb]
Fin
```

TABLEAUX
CAS PRATIQUES N°5

- **Application 21:**

Écrire un algorithme permettant de saisir 10 notes et qui affiche la moyenne de ces notes.

- **Application 22:**

Écrire un algorithme permettant de saisir 10 entiers et qui affiche le maximum de ces entiers.

- **Application 23:**

Écrire un algorithme permettant de saisir 10 entiers dans un tableau, et de calculer le nombre d'occurrences d'un élément N dans ce tableau. Où N saisi par l'utilisateur.

- **Application 24:**

Ecrire un algorithme qui calcule la somme des éléments d'une matrice.

- **Application 25:**

Ecrire un algorithme qui calcule la somme des lignes d'une matrice.

Affaires à suivre