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:

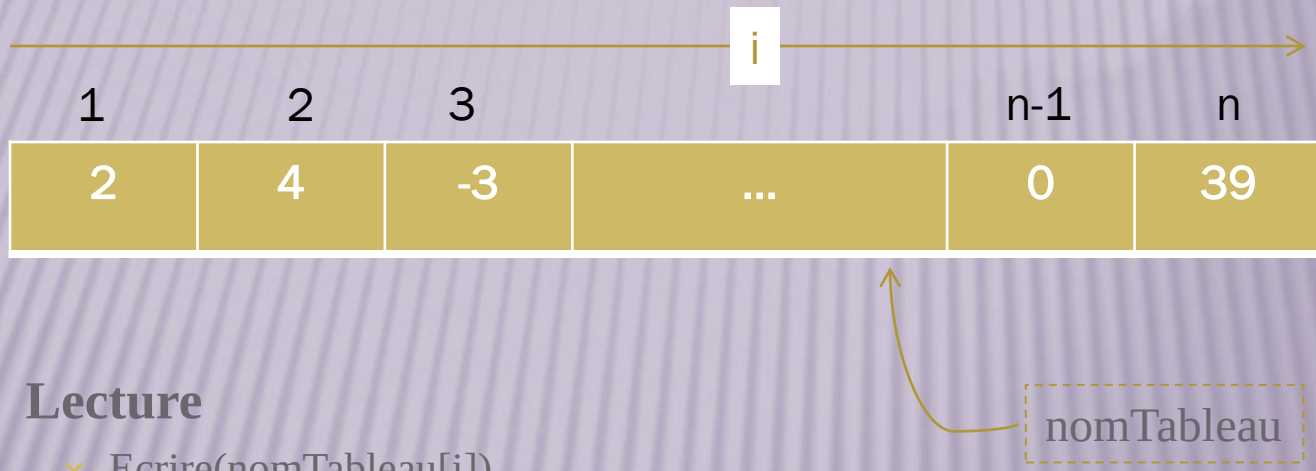
nomTableau: tableau[taille] de type

- **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

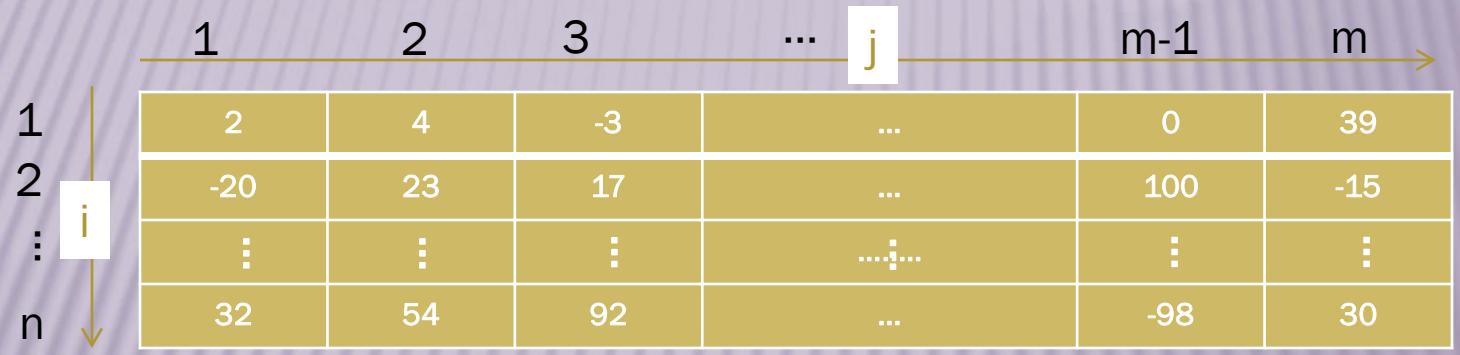
  - **Exemple**:
    - notes: tableau[10][20] de réels
    - texte: tableau[10][255] de caractères
    - matrice: tableau[3][4] de réels

  - **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
REPRÉSENTATION EN MÉMOIRE

- Représentation linéaire:

Les éléments d'un tableau sont placés dans des cases contiguës en mémoire c'est-à-dire de manière successif et sans espace entre les éléments. Prenons le tableau ci-dessous:

Case	35141	35142	35143	35144	35145	35146	35147	35148	35149	35150
Indice	1	2	3	4	5	6	7	8	9	10
Valeur	11	1	8	-13	9	2	6	14	11	15

- Les numéros des cases mémoire (ou adresses) n'ont pas de rapport avec l'indice, mis à part le fait qu'ils sont contigus.
- Soit un tableau de dimension n, si la position du premier élément du tableau est m, celle du deuxième sera alors m+1, celle du troisième sera m+2, ainsi de suite jusqu'au nième qui sera m+n+1.

TABLEAUX
REPRÉSENTATION EN MÉMOIRE

Si représenter un tableau de scalaires à une dimension en mémoire est simple, celle de deux ou n dimensions est un peu moins évident.

Soit un tableau notes de 3 lignes et 5 colonnes, représentant ainsi quinze (15) valeurs. En représentation mémoire cela peut donner:

case	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157
Indice	1,1	1,2	1,3	1,4	1,5	2,1	2,2	2,3	2,4	2,5	3,1	3,2	3,3	3,4	3,5
Valeur	10	7	14	8	12	11	5	12	13	18	20	2	0	17	16

Un tableau à deux dimensions peut être facilement transposé en tableau à une seule dimension.

- Un tableau à n dimensions peut donc être représenté de manière totalement linéaire en mémoire.
- Connaissant la position du premier élément du tableau, nous pouvons connaitre la position de tous les autres éléments en faisant: **p=m+(x*My)+y**; avec:
  - **m** la position connue du premier élément,
  - **x** l'indice de la première dimension moins 1,
  - **y** l'indice de la seconde dimension moins 1,
  - **My** la taille maximale de la première dimension.

TABLEAUX
REPRÉSENTATION EN MÉMOIRE

- Représentation par référence:

Si on considère un tableau de chaine de caractère, tous les caractères de la chaîne sont représentés par une suite de valeurs numériques: les codes ASCII (ou unicode, selon le cas). Si nous souhaitons représenter le mot "Bonjour" par exemple.

- Voici leur valeur en ASCII:

Lettre	B	o	n	j	o	u	r
ASCII	66	111	110	106	111	117	114

- La longueur d'une chaîne de caractères n'est pas forcément connue à l'avance, celle-ci se termine souvent, suivant les langages, par un caractère **nul**. Ainsi en mémoire, nous obtiendrons ceci :

Adresse	3254	3255	3256	3257	3258	3259	3260	3261
Contenu	66	111	110	106	111	117	114	0

  - Il n'est pas évident de représenter les indices des chaînes de caractères.
    - Pour ce repérer, il suffit juste de rechercher le caractère **nul** (0) afin de parcourir toute la chaine.

TABLEAUX
REPRÉSENTATION EN MÉMOIRE

La longueur d'une chaîne de caractères n'étant pas fixe, comment réserver à l'avance l'espace contigu nécessaire au stockage de n chaînes dans un tableau de n éléments ?

- Une solution non optimale consisterait à fixer une valeur arbitraire pour la taille de notre chaîne de caractères. Ceci engendrera une perte de place si votre chaîne ne fait que deux caractères pour deux cents réservés !
- Si maintenant nous affectons une chaîne de caractères à une variable, la longueur de cette chaîne est alors connue (ex: a ← "Salut", sa longueur est 5) et l'espace mémoire contigu nécessaire lui sera alloué pour le stockage de cette chaîne. la variable sera alors la position en mémoire de la chaîne de caractères (son adresse).
  - Soit un tableau de cinq chaînes de caractères :
  - Sa représentation en mémoire sera:

Adresses	3007 ->3009	3010 ->3012	3013 ->3017	3018 ->3021	3022 ->3026
Contenu	Il	Ne	Fait	Pas	Beau

```
Algorithme TabChaines
Variable
    msg:tableau[5] de chaînes
DEBUT
    msg[1]←"il"
    msg[2]←"ne"
    msg[3]←"fait"
    msg[4]←"pas"
    msg[5]←"beau"
FIN
```

- Il faut noter qu'un octet est ajouté pour le caractère nul en fin de chaîne et donc une chaîne de longueur n occupe n+1 octets en mémoire.

TABLEAUX
RECHERCHE D'UN ÉLÉMENT

- Soit un tableau de n éléments correspondant aux prénoms des élèves de la classe. Nous cherchons à savoir si un prenom donné est bien dans ce tableau ou pas.
  - Le principe consiste à balayer l'intégralité du tableau à l'aide d'une structure itérative et à en sortir dès que l'élément a été trouvé ou que le nombre maximal d'indice a été dépassé.
  - À la sortie de la boucle, il faudra de nouveau vérifier pour savoir si oui ou non l'élément a été trouvé: il se peut en effet que tout le tableau ait été parcouru et que ce soit la raison de la sortie de la boucle.

TABLEAUX
RECHERCHE D'UN ÉLÉMENT

- Soit un tableau de n éléments correspondant aux prénoms des élèves de la classe. Nous cherchons à savoir si un prenom donné est bien dans ce tableau ou pas.
  - Le principe consiste à balayer l'intégralité du tableau à l'aide d'une structure itérative et à en sortir dès que l'élément a été trouvé ou que le nombre maximal d'indice a été dépassé.
  - À la sortie de la boucle, il faudra de nouveau vérifier pour savoir si oui ou non l'élément a été trouvé: il se peut en effet que tout le tableau ait été parcouru et que ce soit la raison de la sortie de la boucle.

```
Algorithme RechercheElement
Variable
        noms: tableau[10] de chaînes  de caractère
        rech: chaîne de caractère
        i: entier
DEBUT
        i←1
        TantQue i<=10 et noms[i]<>rech Faire
                i←i+1
        FinTantQue
        i←i-1
        Si noms[i]=rech Alors
                Ecrire("Trouvé")
        Sinon
                Ecrire("Absent")
        FinSi
FIN
```

TABLEAUX
LE PLUS GRAND/PETIT, MOYENNE

- Soit un tableau de n éléments. Nous cherchons à déterminer le plus petit, le plus grand d'une série de notes saisies par l'utilisateur, ainsi que la moyenne.

TABLEAUX
LE PLUS GRAND/PETIT, MOYENNE

- Soit un tableau de n éléments. Nous cherchons à déterminer le plus petit, le plus grand d'une série de notes saisies par l'utilisateur, ainsi que la moyenne.

```
Algorithme MinMaxMoy
Variable
    notes: tableau[10] de réels
    min,max,moy: réels
    i: entier
DEBUT
    min←notes[1]
    max←notes[1]
    moy←0
    Pour i de 1 à 10 faire
        moy=moy+note[i]
        Si note[i]>max Alors
            max←note[i]
        FinSi
        Si note[i]<min Alors
            min←note[i]
        FinSi
    FinPour
    moy←moy/10
    Ecrire(min,max,moy)
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