Sommaire

Chapitre 3 : Tableaux et Matrices 1
3.1. Introduction à la notion de tableaux 1
3.1.1. Déclaration d'un tableau 2
3.1.2. Lecture et écriture d'un tableau 3
3.1.3. Exercices 4

Chapitre 3 : Tableaux et Matrices

3.1. Introduction à la notion de tableaux

Supposons qu'on veut écrire un algorithme qui demande cinq (5) notes, et affiche les notes qui sont supérieures à la moyenne. Dans ce cas, on doit commencer par calculer la moyenne, puis d'afficher dans un deuxième temps, les valeurs supérieures à cette moyenne, comme suit :

```
Algo Moy
Variables
        a, b, c, d, e, moy : Réel
Début
        Lire(a,b,c,d,e)
        Moy = (a+b+c+d+e)/5
        Si a ≥ Moy :
                Ecrire(a)
        FSi
        Si b ≥ Moy :
                Ecrire(b)
        FSi
        Si c ≥ Moy :
                Ecrire(c)
        FSi
        Si d ≥ Moy :
                Ecrire(d)
        FSi
        Si e ≥ Moy :
                Ecrire(e)
        FSi
Fin
```

Imaginons qu'on veut faire le même traitement, mais sur 100 notes au lieu de 5. Dans ce cas, il sera pratiquement impossible de travailler avec 100 variables. La solution est d'utiliser un nouveau type de donnée, à savoir le tableau, qui permet la sauvegarde de 100 notes dans une même variable composée **T** :

```
Algo Moy_Tab
Variables
        T : Tableau de 100 Réel
        Moy : Réel
Début
        Pour i ← 1 à 100 :
                Lire(T(i))
        FPour
        Moy ← 0
        Pour i ← 1 à 100 :
                 Moy ← Moy + T(i) /100
        FPour
        Pour i ← 1 à 100 :
                Si T(i) ≥ Moy :
                        Ecrire(T(i))
                FSi
        FPour
Fin
```

3.1.1. Déclaration d'un tableau

On déclare le tableau de l'exemple précédent comme suit :

Un tableau donné peut contenir des éléments de n'importe quel type, mais d'un seul type à la fois ; on ne peut pas avoir des éléments de différents types dans le même tableau.

L'un des avantages de ce type de donnée (les tableaux) est que les algorithmes écrits pour un tableau de 100 éléments par exemple, peuvent facilement être modifiés pour fonctionner avec un tableau d'une autre taille, il suffit de remplacer toute occurrence de la taille **100** par sa nouvelle valeur.
Afin de rendre cette tâche de modification plus simple, on fait recours à la notion de déclaration de constantes. A la différence d'une variable, une constante est déclarée pour ne pas être modifiée par la suite dans le corps de l'algorithme. On la déclare dans un bloc à part comme suit :

```
Algo Moy_Tab_Cons
Constantes
        D = 100
Variables
        T : Tableau de D Réel
        Moy : Réel
Début
        Pour i ← 1 à D :
                Lire(T(i))
        FPour
        Moy ← 0
        Pour i ← 1 à D :
                Moy ← Moy + T(i) /D
        FPour
        Pour i ← 1 à D :
                Si T(i) ≥ Moy :
                        Ecrire(T(i))
                FSi
        FPour
Fin
```

Par la suite, si on veut modifier l'algorithme pour fonctionner sur un tableau de 1000 éléments, il suffit de remplacer la valeur de **D** dans le bloc **Constantes** par **1000**.

L'algorithmique offre d'autre part, un outil permettant de définir de nouveaux types, comme suit :

Déclaration sans définition de nouveaux types	Déclaration avec définition d'un nouveau type **TAB**
Constantes D = 100 **Variables** T : Tableau de D Réel Moy : Réel	**Constantes** D = 100 **Types** **TAB = Tableau de D Réel** **Variables** T : **TAB** Moy : Réel

3.1.2. Lecture et écriture d'un tableau

Dans l'exemple précédent, la première boucle permet de lire le tableau, élément par élément :

```
Pour i ← 1 à D :
    Lire(T(i))
FPour
```

Il n'est pas possible de lire un tableau d'un coup comme un seul bloc. L'instruction **Lire(T)** n'est pas correcte. Il faut procéder élément par élément, généralement à travers une boucle.

De même pour écrire un tableau, on ne peut pas agir directement en une seule instruction : **Ecrire(T)**, mais il faut procéder élément par élément, généralement à travers une boucle.

```
Pour i ← 1 à D :
    Ecrire(T(i))
FPour
```

Lecture d'un tableau	Exemple de traitement d'un tableau	Affichage d'un tableau

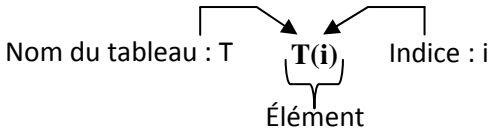
Pour i ← 1 à D :
  Lire(T(i))
FPour
	Moy ← 0
	Pour i ← 1 à D :
	  Moy ← Moy + T(i) /100
	FPour
		Pour i ← 1 à D :
		  Ecrire(T(i))
		FPour

Un tableau à une dimension (appelé aussi vecteur) est donc, une variable composée de plusieurs éléments repérés par des indices :

T	**indices**	1	2	3	4	5	6	7	8	9	10
	Éléments	**11.5**	**13.25**	**10**	**8.75**	**15**	**9**	**7.5**	**12**	**14**	**10.25**

Le tableau précédent appelé **T**, se compose de dix (10) éléments (cases) repérés par des indices (de **1** à **10**)[1], chaque élément contient une valeur réelle. T(1) par exemple, est le premier élément du tableau, il contient la valeur **11.5**, T(2) est le deuxième, il contient la valeur **13.25** et T(10) quant à lui, est le dixième élément, il contient **10.25**. Le nombre d'éléments, ici 10, représente la **taille** du tableau.

Il faut donc différencier entre l'indice d'un élément (son numéro) et l'élément lui-même qui contient la valeur proprement dite.



Dans l'exemple précédent, on peut fusionner les deux premières boucles pour permettre la lecture et le calcul de la moyenne en même temps, mais on ne pourra pas fusionner la dernière boucle d'affichage dans ce cas, puisqu'il faut calculer la moyenne en premier, et faire une deuxième passe pour afficher les notes qui sont supérieures à la moyenne :

[1] De **0** à **9** en langage C

```
Algo Moy_Tab_Cons
Constantes
        D = 10
Variables
        T : Tableau de D Réel
        moy : Réel
Début
        Moy ← 0
        Pour i ← 1 à D :
                Lire(T(i))
                Moy ← Moy + T(i) /D
        FPour
        Pour i ← 1 à D :
                Si T(i) ≥ Moy :
                        Ecrire(T(i))
                FSi
        FPour
Fin
```

3.1.3. Exercices

Exercice 01:

a- Écrire un algorithme qui permet à l'utilisateur de remplir un tableau de 30 caractères (dans une boucle à part), et affiche le nombre de voyelles de ce tableau. **Note** : les voyelles sont : a, e, i, o, u, y.

b- Écrire un algorithme qui permet à l'utilisateur de remplir un tableau de 25 entiers (dans une boucle à part), et affiche le nombre, la somme et le pourcentage d'éléments impairs du tableau.

Exercice 02:

Écrire un algorithme qui permet à l'utilisateur de remplir un tableau de 50 réels (dans une boucle à part), et d'afficher :

a- la plus grande valeur.

b- la plus grande valeur et son indice. S'il y a plusieurs éléments qui donnent la même plus grande valeur, l'algorithme devra afficher le plus petit indice.

c- la plus grande valeur et son indice. S'il y a plusieurs éléments qui donnent la même plus grande valeur, l'algorithme devra afficher tous les indices correspondants.

Exercice 03:

Écrire un algorithme qui demande de remplir deux tableaux A et B, de 45 réels chacun (dans deux boucles à part), et de les additionner dans un troisième tableau C, élément par élément, d'afficher le contenu de C, puis de calculer et d'afficher leur produit scalaire.

$$\sum_{i=1}^{45} A(i) \times B(i)$$

Exercice 04:

Écrire un algorithme qui demande de remplir un tableau de 30 caractères (dans une boucle à part), et faire un décalage circulaire du tableau :

a- vers la gauche d'une position.

b- vers la droite d'une position.

c- vers la droite de K positions.

d- vers la gauche d'une position à partir du n^{ème} élément.

On doit à la fin, afficher pour chaque cas le contenu du tableau après décalage.

Exercice 05:

Écrire un algorithme qui demande de remplir un tableau de 20 chaines de caractères (dans une boucle à part), et d'inverser les valeurs de ce tableau (la valeur du 1ère élément sera rangée dans le dernier, celle du 2ème élément sera rangée dans l'avant dernier et ainsi de suite):

a- dans un autre tableau.

b- dans le même tableau initial, sans utiliser aucun autre tableau intermédiaire.

L'algorithme doit à la fin, afficher le tableau résultant.