Chapitre 4 : Tableaux et matrices

1 Tableaux

1. 1 Principes et notations

Tableau : Variable portant un nom et divisée en cases alignées les unes aux autres sur un même axe.

Type tableau : toutes les valeurs contenues dans les cases d'un tableau ont obligatoirement le même type. C'est celui qui est appelé type du tableau (hormis dans de rares langages de programmation où la diversité du type est permise). Un tableau peut contenir des variables de tous types : entier, reel, booleen, etc.

Déclaration tableau :

Nom-Tableau : Tableau de Nombre-éléments type-éléments ;

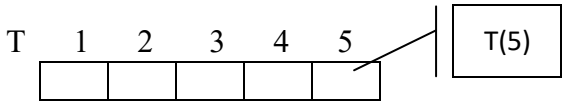
Le nombre d'éléments du tableau doit être clairement précisé lors de la déclaration.

Exemple : Pour calculer la moyenne précédemment traitée nous déclarons un tableau T comme suit : T : Tableau de 10 reel ;

Indice : les cases du tableau sont numérotées successivement. Ce numéro est appelé indice.
En programmation, il dépend du langage utilisé que la première case porte le numéro 0 ou 1 (ou autre) ; la case suivante porte le numéro suivant, etc.
Notes :
- Dans notre cour la première case a le numéro 1, la deuxième le 2 et ainsi de suite.
- L'indice doit être déclaré "entier" car il est impensable d'avoir une case avec le numéro 2.5.

Une valeur dans le tableau : est désignée par le nom du tableau et entre parenthèses l'indice de la case.

Exemple : la 5ème valeur du tableau T se trouve dans la 5ème case et est désignée par T(5).



Note : Il faut différencier le numéro de la case (indice I) de ce qu'il y a dans la case (T(I)).

Les boucles dans le traitement des tableaux : facilitent de beaucoup le travail sur les tableaux.
Exemple : Au lieu d'écrire 5 instructions de lecture pour saisir les valeurs du vecteur N :

```
Lire (N(1)) ; Lire (N(2)) ; Lire (N(3)) ;
Lire (N(4)) ;
Lire (N(5)) ;
```

Une boucle à 5 itérations ferait aussi bien l'affaire

```
Pour | I ← 1 a 5 Faire FinPour ;
     | Lire (N (I)) ;
```

A l'exécution l'indice I changerait à chaque fois de valeur en déplaçant en même temps la case à remplir :

I	N(I)	Case
1	N(1)	Numéro 1
2	N(2)	Numéro 2
3	N(3)	Numéro 3
4	N(4)	Numéro 4
5	N(5)	Numéro 5

L'énoncé cité au début du chapitre peut être traité comme suit lors de l'utilisation de la notion de tableaux :

```
Algorithme Moyenne ;
Déclaration
Variable
   N : Tableau de 10 reel ;
   M : réel ;
I : Entier
Debut
        Pour I ← 1 a 10 Faire
        |    Lire (N(I)) ;
        FinPour ;
        M ← 0 ;
        Pour I ← 1 a 10 Faire
        |    M ← M + N (I) ;
        FinPour ;
        M ← M /10 ;
        Ecrire (M) ;
Fin.
```

Notes :

- Remarquez que la valeur 10 (nombre d'éléments du tableau) est répétée 4 fois dans l'algorithme. S'il nous prenait l'envie de changer le tableau vers 100 éléments nous devrions changer la valeur à 4 endroits différents ; imaginez si l'on oubliait l'un d'eux !!

Une solution a été proposée par l'introduction de la rubrique "Constante" dans la partie déclaration (avant les variables). Il suffit alors de déclarée un objet auquel on assigne (=) une valeur et de travailler avec cet objet dans l'algorithme au lieu de sa valeur. A chaque changement de taille du tableau il suffit de changer la valeur de cet objet et le tour est joué. Mais il faut insister sue le fait que l'exécution étant enclenchée aucun changement de la valeur de la constante n'est possible.

- Il est tout à fait possible (et recommandé lorsqu'on est un informaticien expérimenté) de réaliser le travail de lecture et calcul dans une seule boucle (comme présenté dans l'algorithme suivant) :

```
Algorithme Moyenne2 ;
Déclaration Constante
Variable
        x = 10 ;
   N : Tableau de x reel ;
   M : réel ;
I : Entier
Debut
        M ← 0 ;
      Pour I ← 1 a x Faire
        Debut
            Lire (N (I)) ;
            M ← M + N (I) ;
        Fin
      FinPour ;
        M ← M / x ;
        Ecrire (M) ;
Fin.
```

Mais il est très important de faire très attention en regroupant les boucles particulièrement lorsque l'indice ne doit pas se déplacer de la même manière dans les différentes phases des traitements.

Note générale : !!! Dans le traitement des tableaux tout est dans la manipulation des indices. L'objectif de ce TP est d'implémenter, compiler et exécuter les algorithmes donnés en pseudo code.

1.2 Exercices

Exercice 1 :

Ecrire un algorithme qui permet de lire les valeurs d'un tableau de 50 entiers puis de calculer la somme de ses éléments. Utiliser un sous algorithme pour chaque traitement

```
Algorithme Exo1 ;
Declaration
Constante
        N= 50 ;
Variable
        T : Tableau de N entier ;
        I, S : entier ;
Procedure Lecture (Var A : Tableau de N entier) ;
Debut
        Pour I ← 1 à N Faire
            Lire (A(I)) ;
        FinPour
Fin ;
Fonction Somme (A : Tableau de N entier) : entier ;
Variable
        SS : entier ;
Debut
        SS ← 0 ;
        Pour I ← 1 à N Faire
            SS ← SS + A(I) ;
        FinPour
        Somme ← SS ;
Fin ;
Debut
        Lecture (T) ;
        S ← Somme (T) ;
        Ecrire ('La somme des éléments = ', S) ;
Fin.
```

Note : La déclaration du type tableau a été répétée 3 fois (2 sous-algorithmes et 1 pour l'algorithme) cette répétition d'un texte assez long peut être évitée en utilisant le même principe que celui des constantes sauf que ce n'est plus une valeur que l'on déclare mais un nouveau nom de type (voir l'utilisation du type dans les exercices suivants)

Exercice 2 :

Ecrire un algorithme qui permet de lire les valeurs d'un tableau de 1000 caractères puis d'inverser ces valeurs de façon à ce qui se trouve dans la case 1 soit échangé avec la valeur de la case 1000 puis la 2ème case avec la case 999 et ainsi de suite jusqu'à arriver au milieu du tableau.

```
Algorithme Exo2 ;
Declaration
Constante
        N= 50 ;
Type
        XX = Tableau de N caractere ;
Variable
        T : XX ;
        I : entier ;
Procedure Lecture (Var R : XX) ;
Debut
        Pour I ← 1 à N Faire
        Lire (R(I)) ;
        FinPour
Fin ;
Procedure Ecriture (Var R : XX) ;
Debut
        Pour I ← 1 à N Faire
        Ecrire (R(I)) ;
        FinPour
Fin ;
Procedure Echange (Vary R : XX) ;
  Variable
        J : entier ;
        W : caractere ;
  Debut
  I     ← 1 ;
  J     ← N ;
   Tantque I < J Faire
   Debut
                W ← R(I) ;
      R(I)      ←      R(J)      ;
      R(J) ← W ;
      I ← I+1 ;
      J ← J- 1 ;
     Fin
     FinTantque
  Fin ;
  Debut
        Lecture (T) ;
        Echange (T) ;
        Ecriture (T) ;
  Fin.
```

Exercice 3 :

Ecrire une procédure permettant d'afficher la plus grande valeur réelle dans un tableau en précisant quelle position elle occupe dans le tableau.

```
Procedure Exo3 (P : Tableau de x reel) ;
Déclaration
Variable
   V : réel ;
   I, J : Entier
Debut
      V ← P(1) ;
      J ← 1 ;
      Pour I ← 2 a x Faire
        Debut
          Si P(I) > V Alors
            Debut
              J ← I ;
              V ← P(I) ;
            Fin ;
          FinSi
        Fin
      FinPour ;
        Ecrire ('La plus grande valeur est = ', V, 'Elle se trouve à la position ', J) ; Fin.
```

Note : x étant déclaré constante dans l'algorithme.

Exercice 4 :

Ecrire un algorithme permettant de recherche le nom d'un étudiant dans un tableau contenant les noms des 20 étudiants reçus dans un concours ; on suppose qu'il n'y a pas d'homonymes.

```
Algorithme Exo4 ;
    Declaration
    Constante
        N= 20 ;
    Variable
     T : Tableau de N chaines de caractere ;
     I : entier ;
        Nom : chaine de caractere ;
    Debut
          Pour I ← 1 à N Faire
              Lire (T(I)) ;
          FinPour
         I ← 1 ;
         Lire (Nom) ;
          Tantque (I ≤ N) et (T(I) ≠ Nom) Faire
                I ← I + 1 ;
          FinTantque ;
              Si I > N Alors
                  Ecrire ('l"étudiant n"est pas admis dans le concours')
          Sinon
            Ecrire ('l"étudiant est admis dans le
         concours') FinSi Fin.
```

2 Matrices

L'informatique nous offre la possibilité de déclarer des tableaux dans lesquels les valeurs ne sont pas repérées par une seule, mais par plusieurs coordonnées (2, 3, …)
Cependant les tableaux multidimensionnels à plus de 3 dimensions et plus sont laissés aux mathématiciens alors que nous simples mortels nous contentons des tableaux (une dimension) et matrices (deux dimensions). Le pourquoi se résumerait dans le fait que pour travailler nous avons besoin de gribouiller des schémas pour trouver des solutions aux problèmes ce gribouillage est tout à fait possible pour les tableaux et matrices, devient très fastidieux (besoin de beaucoup d'imagination) pour les tableaux à 3 dimensions et est carrément impossible pour les dimensions supérieures. Résultat on s'arrête aux matrices.

2. 1 Notations

Matrice : Variable portant un nom et divisée en cases alignées sur deux axes. Une matrice peut être considérée comme le regroupement de plusieurs tableaux de même taille. Regrouper trois tableaux ayant chacun 7 éléments aboutira à une matrice de 3 lignes (tableaux) et 7 colonnes (Cases).

Type matrice : De même qu'un tableau le type d'une matrice est celui de ses éléments vu qu'ils doivent tous être du même type.

Déclaration matrice :

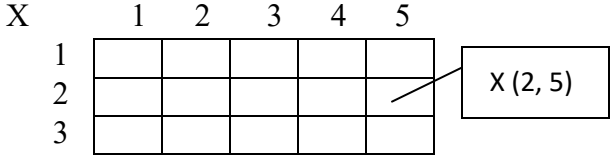
Nom-Matrice **:** Tableau de Nombre-lignes * Nombre -colonnes type-éléments ;

Les nombres de lignes et colonnes doivent être clairement précisés lors de la déclaration.

Exemple : Pour calculer la moyenne des étudiants de 4 sections chaque section ayant 120 étudiants nous déclarons une Matrice M comme suit : ***M : Tableau de 4 * 120 reel ;***

Indice : les lignes d'une matrice sont numérotées par des entiers successifs de même que les cases de chaque ligne. Ces numéros sont appelés indices de ligne et de colonne. En programmation, il dépend du langage utilisé que la première case porte le numéro 0 ou 1 (ou autre) ; la case suivante porte le numéro suivant, etc. Nous respecterons la consigne donnée pour les tableaux de débuter par "1"

Une valeur dans la matrice : est désignée par le nom de la matrice et entre parenthèses l'indice de la ligne suivi de celui de la colonne.
C'est comme quand on arrive en visite chez un ami qui habite un bâtiment il faut monter à l'étage puis frapper à la bonne porte.
Exemple : la 5ème Case de la 2ème ligne de la matrice X est désignée par X (2, 5).



Les boucles dans le traitement des matrices : Le traitement de plusieurs cases de plusieurs lignes d'une matrice se fait en général par des boucles imbriquées.

Exemple : Si nous avons à remplir les cases da matrice X ci-dessus nous aurons à utiliser les boucles suivantes :

```
Pour I ← 1 a 3 Faire
    Pour J ← 1 a 5 Faire
        Lire (X (I, J)) ;
    FinPour ;
FinPour ;
```

A l'exécution l'indice I serait incrémenté après que l'instruction de lecture aurait été exécutée 5 fois (l'indice J aurait changé en 1 à 5). A chaque nouvelle valeur, l'indice I bloquera pour permettre à l'indice J de faire un tour complet.

I	J	X(I,J)	Case
1	1	X(1, 1)	Ligne 1 colonne 1
1	2	X(1, 2)	Ligne 1 colonne 2
1	3	X(1, 3)	Ligne 1 colonne 3
1	4	X(1, 4)	Ligne 1 colonne 4
1	5	X(1, 5)	Ligne 1 colonne 5
2	1	X(2, 1)	Ligne 2 colonne 1
2	2	X(2, 2)	Ligne 2 colonne 2
2	3	X(2, 3)	Ligne 2 colonne 3
2	4	X(2, 4)	Ligne 2 colonne 4
2	5	X(2, 5)	Ligne 2 colonne 5
3	1	X(3, 1)	Ligne 3 colonne 1
3	2	X(3, 2)	Ligne 3 colonne 2
3	3	X(3, 3)	Ligne 3 colonne 3
3	4	X(3, 4)	Ligne 3 colonne 4
3	5	X(3, 5)	Ligne 3 colonne 5

La matrice est alors traitée (ici remplie) ligne par ligne. Il est possible de la traiter colonne par colonne (ç-à-d, traiter les cases numéro 1 de la ligne 1 puis de la ligne 2 et ainsi de suite jusqu'à la dernière ligne avant de traiter les cases avec numéro 2 de la ligne 1 puis Ligne 2 et ainsi de suite). Il suffit pour cela d'inverser les boucles da façon à ce que la boucle de l'indice colonnes soit celle mise à l'extérieure. Exécutez les boucles suivantes pour constater cet état :

```
Pour J ← 1 a 5 Faire
    Pour I ← 1 a 3 Faire
        Lire (X (I, J)) ;
    FinPour ;
FinPour ;
```

Note : Entre les parenthèses le premier indice est toujours celui des lignes et le deuxième celui des colonnes quelques soit la manipulation que l'on fait sur la matrice.

Exemple : l'algorithme suivant permet de remplir automatiquement toutes les cases d'une matrice A (2, 3) par des valeurs entières successives partant du 0. Le résultat de ce traitement donnera une matrice comme la suivante :

A

0	1	2
3	4	5

```
Algorithme Val-succ ;
Declaration
Constante
        N = 2 ;
        M = 3 ;
Variable
        A : Tableau de N*M entier ;
        Val, I, J : entier ;
Debut
        Val ← 0 ;
        Pour I ← 1 a N Faire
            Pour J ←  1 a M Faire
              Debut
                A (I, J) ← Val ;
                Val ← Val + 1
              Fin
            FinPour
        FinPour ;
        Pour I ← 1 a N Faire
            Pour J ←  1 a M Faire
                Ecrire (A (I, J) ;
            FinPour
          FinPour ;
  Fin.
```

Note : Certains traitements ne peuvent être exécutés que sur des matrices carrées (Nombre de lignes = nombre de colonnes). Tel que les traitements sur la diagonale principale ou les triangles inférieurs et/ ou supérieurs.

2.2 Exercices

Exercice 1 :

Soit un tableau T à deux dimensions (12, 8) préalablement rempli de valeurs réelles. Écrire un sous-algorithme qui recherche la plus grande valeur au sein de ce tableau.

```
Fonction Maximum (M : Tableau de 12*8 reel) : reel ;
Declaration
Variable
        I, J : entier ;
        Max : Reel ;
Debut
        Max ← M (1, 1) ;
        Pour I ← 1 a 12 Faire
            Pour J ←  1 a 8 Faire
                Si Max < M (I, J) Alors
                    Max ← M (I, J) ;
                FinSi
            FinPour
        FinPour ;
        Maximum ← Max
  Fin.
```

Exercice 2 :

Soit une matrice d'entiers W (N, N). Écrire une procédure qui initialise la diagonale de la matrice à 0.

```
Procedure Zero-diag (var W : Tableau de N*N entier) ;
Declaration
Variable
        I: entier ;
Debut
          Pour I ← 1 a N Faire
                                W (I, I) ← 0 ;
          FinPour ;
  Fin.
```

La diagonale est constituée d'un élément par ligne. (1, 1) ; (2, 2) ; (3, 3) ; etc. Une seule boucle (pour passer d'une ligne à une autre et traiter ainsi toutes les lignes) est nécessaire au traitement de la diagonale.

Exercice 3 :

Soit une matrice d'entiers M (N, N). Écrire un algorithme pour vérifier si la matrice est triangulaire supérieure ou non.

Note : Matrice triangulaire supérieure c.-à-d. que les informations essentielles sont dans le triangle supérieur (les éléments se trouvant en dessus de la diagonale) alors que tous les éléments du triangle inférieur (les éléments se trouvant en dessous de la diagonale) sont à zéro. (L'inverse est correct pour la triangulaire inférieure)

Pour traiter les éléments du triangle inférieur il faut avant tout les identifier pour comprendre la manière selon laquelle doivent être écrites les boucles et manipuler les indices correctement.
Prenons le cas d'une matrice 4*4 ; les cases qui sont dans le triangle inférieur sont les suivantes :

Ligne 1 Pas d'élément
Ligne 2 Colonne 1 Ligne 3 Colonnes 1 et 2
Ligne 4 Colonnes 1, 2 et 3

On remarque que bien que la ligne 1 ne soit pas concernée par le travail plusieurs lignes doivent être traitées et donc la boucle des lignes et nécessaire avec un indice qui va de 2 à N.
L'indice des colonnes se déplace quand à lui de 1 à indice-ligne - 1. Cette étude sera utilisée dans la fonction vérif.

```
Algorithme TS ;
Declaration
Constante
        N = 100 ;
Type
        Type1 = Tableau de N*N entier ; Variable
        M : Type1.
        I, J : entier ;
        C : booleen ;
Fonction Verif (X : Type1) : booleen ;
Declaration
Variable
        B : Booleen ;
Debut
B ← vrai ;
I ← 2 ;
Tantque (I ≤ N) et (B) Faire
Debut
J ← 1 ;
   Tantque (J ≤ I -1) et (B) Faire
      Debut
                Si X (I, J) ≠ 0 Alors
                   B ← Faux
                FinSi
                J ← J + 1 ;
      Fin
    FinTantque ;
    I ← I + 1 ;
  Fin
FinTantque
 Verif ← B ;
Fin ;
Procedure Lecture (Var A : Type1) ;
Debut
        Pour I ← 1 a N Faire
        |   Pour J ← 1 a N Faire
        |   |       Lire (A (I, J)) ;
        |   FinPour ;
        FinPour ;
Fin ;
Debut (* Algo*)
   Lecture (M) ;
    C ← Verif (M) ;
    Si   C   Alors
    |       Ecrire (‘La matrice est triangulaire supérieure’)
    Sinon
    |       Ecrire (‘La matrice n"est pas triangulaire supérieure’)
    FinSi Fin.
```