

Chapitre 4 : Structures de données de base

1 Tableaux

L'algorithme suivant est celui que nous écrirons afin de calculer la moyenne de 10 valeurs que nous devons lire avant de commencer le calcul – en considérant ce que nous avons déjà vu - :

Algorithme Moyenne ;

Déclaration

Variable

N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, M : réel ;

Debut

```

Lire (N1) ;
Lire (N2) ;
Lire (N3) ;
Lire (N4) ;
Lire (N5) ;
Lire (N6) ;
Lire (N7) ;
Lire (N8) ;
Lire (N9) ;
Lire (N10) ;
M ← (N1+ N2+ N3+ N4+ N5+ N6 + N7 + N8 + N9 + N10)/10 ;
Ecrire (M) ;

```

Fin.

Bien que le copier-coller m'ait facilité l'écriture de ce code il reste astreignant de manipuler plusieurs variables (imaginez si nous avons 100 ou 100000 variables !!!!).

Une solution trouvée était de regrouper toutes les valeurs dans un seul objet qui à un moment "t" possède donc non pas une seule valeur mais plus d'une. La notion de tableau a été alors inventée.

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 ;

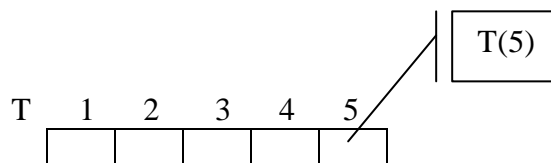
Indice : les cases du tableau sont numérotées successivement. Ce numéro est appelé indice. En programmation, il dépend du langage utilisé que la première case porte le numéro 0 ou 1 (ou autre) ; la case suivante porte le numéro suivant, etc.

Notes :

- Dans notre cours 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 qui la contient.

Exemple : la 5^{ème} valeur du tableau T se trouve dans la 5^{ème} 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 à 5 Faire
  Lire (N (I)) ;
FinPour ;
```

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 de la moyenne 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 réel ;

M : réel ;

I : Entier ;

Debut

Pour I \leftarrow 1 à 10 Faire

 Lire (N(I)) ;

FinPour ;

M \leftarrow 0 ;

Pour I \leftarrow 1 à 10 Faire

 M \leftarrow M + N (I) ;

FinPour ;

M \leftarrow 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 entête (avant les variables). Il suffit alors de déclarer un objet auquel on assigne (avec le symbole =) 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 dans la partie déclaration et le tour est joué. Mais il faut insister sur le fait que dès que l'exécution est 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

$x = 10 ;$

Variable

N : Tableau de x réel ;

M : réel ;

I : Entier ;

Debut

$M \leftarrow 0 ;$

Pour I $\leftarrow 1$ a x Faire

Lire (N (I)) ;

$M \leftarrow M + N (I)$

FinPour ;

$M \leftarrow M / x ;$

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.

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 ;

Procédure 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 constante 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 que la valeur qui se trouve dans la case 1 soit échangée (permutée) avec la valeur de la case 1000 puis la 2^{ème} 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

W ← R(I) ;

R(I) ← R(J) ;

R(J) ← W ;

I ← I+1 ;

J ← J- 1

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.

Procédure Exo3 (P :XX) ;

Déclaration

Variable

V : réel ;

I, J : Entier

Debut

V ← P(1) ;

J ← 1 ;

Pour I ← 2 à x Faire

Si P(I) > V Alors

J ← I ;

V ← P(I) ;

FinSi

FinPour ;

Ecrire ('La plus grande valeur est = ', V, 'Elle se trouve à la position ', J) ;

Fin.

Note : XX est déclaré type dans l'algorithme comme suit : XX = Tableau de x reel) ; x étant le nombre d'éléments du tableau.

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, mais 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 que pour 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 débiter 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 5^{ème} Case de la 2^{ème} ligne de la matrice X est désignée par X (2, 5).

X		1	2	3	4	5
1						
2						
3						

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 à 3 Faire
  Pour J ← 1 à 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 de façon à ce que la boucle de l'indice colonnes soit celle mise à l'extérieur. 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 quelque 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) tels 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 : Tab) : reel ;

Declaration

Variable

I, J : entier ;

Max : Reel ;

Debut

Max \leftarrow M (1, 1) ;

Pour I \leftarrow 1 a 12 Faire

Pour J \leftarrow 1 a 8 Faire

Si Max < M (I, J) Alors

Max \leftarrow M (I, J) ;

FinSi

FinPour

FinPour ;

Maximum \leftarrow Max

Fin.

Note :

Type

Tab = Tableau de 12*8 reel

Exercice 2 :

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

Procédure Zero-diag (var W : Tableau de N*N entier) ;

Declaration

Variable

I: entier ;

Debut

Pour I \leftarrow 1 a N Faire

W (I, I) \leftarrow 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 est 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.

Exercice 4 :

Ecrire une fonction ou procédure qui permet de calculer le produit de deux matrices A (4 x 5) et B (5 x 3) dans une troisième matrice C.

Le produit de deux matrices ne peut se définir que si le nombre de colonnes de la première matrice est le même que le nombre de lignes de la deuxième matrice. De plus, en général, le produit matriciel n'est pas commutatif.

Exemple :

$$\begin{bmatrix} 1 & 0 \\ -1 & 3 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} (1 \times 3 + 0 \times 2) & (1 \times 1 + 0 \times 1) \\ (-1 \times 3 + 3 \times 2) & (-1 \times 1 + 3 \times 1) \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 3 & 2 \end{bmatrix}$$

Procédure Produit_matriciel (A : Mat1, B: Mat2 ; Var C : Mat 3) ;

Variable

I, J, K : entier ;

Debut

Pour I ← 1 a N Faire

Pour J ← 1 a L Faire

C (I, J) ← 0 ;

Pour K ← 1 a M Faire

C (I, J) ← C (I, J) + A (I, K) * B (K, J)

FinPour

FinPour

FinPour

Fin ;

Note :

Constante

N = 4 ;

M = 5 ;

L = 3 ;

Type

Mat 1 : Tableau de N * M Entier ;

Mat 2 : Tableau de M * L Entier ;

Mat 3 : Tableau de N * L Entier ;

3 Type énuméré

Un type énuméré est un type dont on énumère, c'est-à-dire dont on donne la liste effective, les éléments de ce type. On voit immédiatement des cas où cela peut être intéressant, par exemple pour traiter :

- les mois de l'année (janvier, février...);
- les indications d'état civil (célibataire, marié, divorcé, veuf).
- les notes de musique ;
- les marques de voiture ;

Bien entendu il ne peut être question que des types correspondants à un ensemble fini d'éléments.

Type

Nom-enum = {valeur 1, valeur 2, Valeur n}

Donc un type énuméré ne peut avoir qu'un nombre limité de valeurs.

Exemples :

Jour = {samedi, dimanche, lundi, mardi, mercredi, jeudi, vendredi};

Mois = {janv, fev, mars, avril, mai, juin, juillet, aout, sept, oct, nov, dec}

Couleurs-feux = {rouge, orange, vert}

Notes : On ne peut pas utiliser les instructions lire et écrire sur une variable de type énuméré mais sont utilisables l'affectation et les conditions avec de nouvelles actions telles que :

- Succ (x) : qui permet de passer de la valeur de x vers celle qui la suit en respectant l'ordre du type énuméré déclaré sachant que les valeurs de ce dernier sont cycliques (ç-à-d que si x = la dernière valeur du type énuméré le suivant (succ) sera la première valeur du type)
- Pred (X) est l'inverse de Succ ç-à-d que c'est la valeur qui vient avant.

Exemple :

Si on déclare

Type

Couleurs-feux = {rouge, orange, vert}

Variable

A : Couleurs-feux ;

Et que

A ← orange ;

Succ (A) = vert & Pred (A) = rouge

4 Ensemble

Les ensembles en algorithmique gardent le même principe que ceux utilisés en mathématiques avec les mêmes lois et les mêmes opérateurs. L'utilisation des ensembles n'ont d'intérêt que pour certains.

Un ensemble est une variable qui contient un nombre fini d'éléments de même type. Ce type doit être de type ordinal c'est-à-dire qu'il ne doit être ni de type réel, ni de type chaîne de caractères, ni tout autre type complexe. Ce type doit être énuméré et ne doit pas excéder 256 combinaisons possibles. La déclaration d'une telle variable se fait par l'utilisation de la syntaxe suivante :

Type

W = ensemble de (valeur1, valeur2, valeur N) ;

Ou encore en version intervalle :

Z = ensemble de valeur-min .. Valeur-max ;

Une variable x déclarée de ce type peut contenir une combinaison d'une, deux ou toutes les valeurs du type ensemble.

Exemple :

Type

Binaire = ensemble de 0..1 ;

Ou encore

Type

Prenom = (Ahmed, Wahiba, Guermia) ;

Variable

club : ensemble de (Ahmed, Wahiba, Guermia) ;

Dans cet exemple, la variable club est un ensemble de type prenom, c'est-à-dire que cet ensemble nommé club ne peut contenir que les éléments déclarés dans le type prenom. Donc la variable club pourra contenir une combinaison de ces trois éléments.

Ainsi déclaré, le type apparaît comme une constante dont on peut connaître l'index des éléments avec la fonction Ord ().

Ici, Ord (Ahmed) = 0, Ord (Wahiba) =1 et Ord (Guermia) = 2. Le type fonctionne un peu comme la table ASCII... l'index du premier élément étant zéro.

Les ensembles ne sont pas ordonnés. Donc il n'existe pas d'ordre d'apparition des éléments dans une variable ensemble. On peut tout juste être capable de comparer le contenu de deux ensembles de même type, et de déterminer si un élément est inclus ou non dans un ensemble. De plus, un même élément n'apparaît qu'une seule fois dans un ensemble. Et il n'existe pas de fonction qui renvoie le nombre d'éléments d'un ensemble.