Les tableaux

Introduction

Imaginons que dans un algorithme, nous ayons besoin simultanément de 20 notes pour calculer une moyenne. Actuellement, la seule solution consiste à déclarer 20 variables, appelées par exemple Notea, Noteb, Notec, … ou bien N1, N2, N3… ensuite il faut écrire 20 instructions « écrire » et 20 instructions « lire » puis en arrive à l'instruction qui calcule la moyenne qui sera comme suit :

Moy← (N1+N2+N3+.....N20) / 20

C'est fatigant !!!!!! Imaginons un algorithme de gestion avec quelques centaines ou quelques milliers de valeurs à traiter alors là c'est le suicide direct. !!!!?????

Solution : L'algorithmique nous permet de rassembler toutes ces variables en une seule variable (appelé **tableau**), au sein de laquelle chaque valeur sera désignée par un numéro.

Définition

1. Un **tableau** est un ensemble de valeurs à clé (nom) unique telle que :
   - Le nombre d'éléments du tableau (**dimension** ou **taille**) est constant,
   - L'accès aux éléments s'effectue directement par la clé,
2. Un **tableau** est un objet structuré formée de zones mémoire contiguës. Les données dans un tableau ont toutes le même type. Ce dernier est précisé à la déclaration avec la taille du tableau (le nombre d'éléments) comme suit :

Syntaxe

Nom-de-tableau : Tableau [1..taille] : type_element

En Fortran

```
Type_element::nom-de-tableau(taille)
```

Exemple :

```
Variable
A : Tableau [1..10] d'entier
E : Tableau [1..500] de caractère
```

```
integer ::A(10)
character ::E(500)
```

Ou

```
Constant taille=500
Variable
T : Tableau [1..taille] de réel
```

```
Integer,parameter ::taille=500
integer ::T(taille)
```

Pour accéder à un élément d'un tableau on doit préciser son rang. Exemples :

Écrire (A(4)) **E(99) ← '$'** **A(9) ← 4 + A(2)**

Remarque:

- Chaque élément d'un tableau est une variable.
- Un tableau peut être vu comme un ensemble de variables qui ont le même nom.

Lecture et écriture d'un tableau:

La lecture (ou l'écriture) d'un tableau ne peut se faire qu'élément par élément en parcourant tout le tableau. Donc, il est pratique d'utiliser une boucle.

- Exemple : cet algorithme permet de lire un tableau puis de l'afficher

```
Algorithme lire_tableau
Constant max=50
Variable
T :tableau(max)d'entier
I :entier
Debut
Pour i de 1 à max faire
Lire (T(i))
Finpour
```

```
Algorithme Ecrire_tableau
Constant max=50
Variable
T :tableau(max)d'entier
I :entier
Debut
Pour i de 1 à max faire
Ecrire (T(i))
Finpour
```

Quelques exemples d'algorithmes

Stocker des valeurs dans un tableau

```
Algorithme lecture
Variable
T :tableau(10)de réel
I :entier
Début
Pour i de 1 à 20  faire
    Ecrire ("donnez la note numéro", i) ;
    Lire (Note(i));
finPour ;
Fin.
```

Echange d'éléments

```
Algorithme echange
Variable
T :tableau(10) de réel
I :entier
K:réel
Début
K= Note(1)
Note(1)= Note(2)
```

```
Note(2)=K
fin
```

Somme des éléments d'un tableau d'entiers

```
Algorithme Somme;
Var N : tableau[1..5]d'entier ;
    S, i: entier;
Début
S ← 0 ;
Pour i ← 1 à 5  faire
     S← S+ N[i] ;
finPour
Fin.
```

Recherche d'un élément

```
Algorithme recherhe;
Variable  N : tableau tableau[1..5]  d' entier
   K, i: entier;
   Trouv : booléen ;
Début
K← 15 ;
I←1 ;
Trouv←faux ;
TQ i <= 5 ET trouv=faux faire
    Si N[i]=K alors
       Trouv←Vrai ;
    Finsi ;
   i←i+1 ;
finTQ ;

Fin.
```

Tri d'un tableau

L'objectif du *tri* est d'ordonner une séquence de *N* entiers qui sont rangés dans un tableau.

Le tri par sélection

Le but de tri par sélection soit de trier un tableau dans l'ordre croissant, on met le plus petit élément dans la case numéro 1. Puis en met dans la case numéro 2 le plus petit élément suivant. Et ainsi de suite jusqu'au dernier :

45	122	12	3	21

3	122	12	45	21

3	12	122	45	21

3	12	21	45	122

```
Algorithme   tri_selction ; ;
Var     N : tableau [1.. 10] de entier ;
          i,  j, posmini, temp : entier;

Début
 Pour i ← 1 à 9 faire
   posmini ← i

     Pour j ← i + 1  à 10 faire
        Si  t(j) < t(posmini) Alors
           posmini ← j
        Finsi
     FinPour

    temp ← t(posmini)
    t(posmini) ← t(i)
    t(i) ← temp
  FinPour
Fin.
```

Le tri à bulles

Consiste à faire remonter progressivement les plus grands éléments d'un tableau.

```
Algorithme tri_bulle ;
Var N : tableau [1.. 10] de entier ;
    i,j,  temp : entier;
   échange: booleen;
Début
Répéter
  échange←Faux ;
  Pour i ← 1 à 9 faire
      Si t[i] > t[i+1] Alors
         temp ← t[i] ;
         t[i] ← t[i+1] ;
         t[i+1] ← temp ;
        Echange← vrai ;
      Finsi ;
   FinPour ;
Jusqu'à échange=faux ;
Fin.
```

Les matrices

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).

Une matrice est un tableau à deux dimensions.

Déclaration:

```
Var
Nom-de-matrice:Tableau [1..nbr_lignes, 1..nbr_colonnes] : type
```

Exemple :

```
Var  M : Tableau [1..10, 1..20] : entier
```

En Fortran

```
integer ,dimension(n,m) :: M
```

Pour accéder à un élément d'une matrice on doit préciser sa position (la ligne et la colonne).

Exemple : `M[5, 13] ← 1400`

Pour la lecture (ou l'écriture) d'une matrice il est pratique d'utiliser deux boucles imbriquées.

Quelques exemples d'algorithmes

Initialisation d'une matrice :

```
Algorithme Init ;
Var Note : tableau [1.. 10, 1..20] de réel ;
    i,j: entier;
Début
     Pour i ← 1 à 10  faire
        Pour j ← 1 à 20  faire
          N[i,j]←0 ;
        finPour ;
     finPour ;
Fin.
```

Somme de deux matrices

```
Algorithme Somme;
Var N,M,L : tableau [1.. 10, 1..20] de réel ;
    i,j: entier;
Début
     Pour i ← 1 à 10  faire
        Pour j ← 1 à 20  faire
```

N[i,j]←M[i,j] +L[i,j];
finPour ;
finPour ;
Fin.

```
Pour i de 1 à 10 faire
Pour j de 1 à 20 faire
Lire (M[i, j])
Finpour
Finpour
```

Exercice

Soit une matrice d'entiers M(NxN).

Écrire un algorithme qui initialise la diagonale de la matrice à zéro.

```
Algo diagonal
Constant N=10 ;
Var M :tableau[1..N,1..N]d'entier ;
Var i :entier
Debut
Pour i←1à N faire
M(i,i)=←0 ;

Finpour
```

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.

Exemple de manipulation des matrices en FORTRAN

```
program matrice
        implicit none
        integer,parameter::n=3,m=2
        real ,dimension(n,m) :: A,B,C
        integer:: i,j
       print*,'Saisir les éléments de la matrice A'
        do i=1,n
          do j=1,m
          read *, A(i,j)
        enddo
        enddo
        print*,'Saisir les éléments de la matrice B'
        do i=1,n
          do j=1,m
          read *, B(i,j)
        enddo
        enddo
```

```
   print*,'les éléments de la matrice C :'
    do i=1,n
     do j=1,m
C(i,j)=A(i,j)+B(i,j)
   enddo
   enddo


 print *, 'la nouvelle matrice sera:'
 print *,C
  end program matrice
```