# LES TABLEAUX

# Les Tableaux

## 1- Introduction

**Imaginons que dans un programme, nous ayons besoin simultanément de 25 valeurs (par exemple, des notes pour calculer une moyenne). La solution consiste à déclarer 25 variables réelles, appelées par exemple n1,n2,n3,…,n25 et la variable moyenne réelle.**

**moyenne = (n1+n2+n3+…+n25) / 25**

**l'ordinateur va réserver 25*4=100 octets pour les valeurs réelles des 25 variables et 25*4=100 octets pour les adresses de ces 25 variables.**

**La programmation nous permet de rassembler toutes ces variables en une seule, appelé note[], qui est un ensemble de valeurs portant le même nom de variable et repérées par un nombre, cet ensemble s'appelle un tableau, et le nombre qui sert à repérer chaque valeur s'appelle un indice.**

**Un tableau contient des variables de tous types : tableaux d'entiers, tableaux de réels, tableaux de caractères ou tableaux de booléens, mais on ne peut pas faire un mixage de types différents au sein d'un même tableau.**

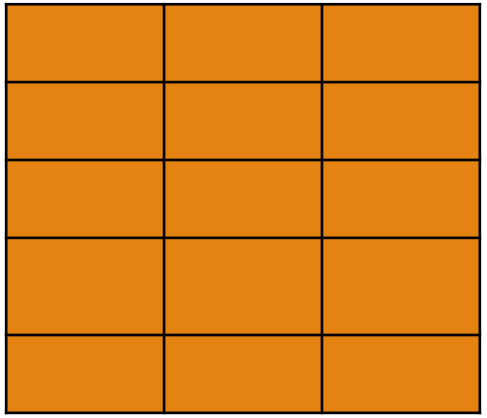
# Les Tableaux

**Il existe deux types de tableaux :**

**Tableaux à une dimensions**

- **Les tableaux statiques**
- **Les tableaux dynamiques**

**Tableaux à deux dimensions**



# Les Tableaux

## 2- Tableaux à une dimensions

### A- Remplir et afficher un tableau

**Un tableau de taille n est une structure très simple constituée de n emplacements consécutifs en mémoire.**

**Il n'est cependant pas possible de modifier la taille d'un tableau, on dit que cette structure est statique.**

**Pour déclarer un tableau en algorithmique :**

**Tableau Nom_tab[taille] : type**

# Les Tableaux

**Exp :**

**Tableau Note[25] : réels**

**Chaque élément est repéré par son indice (position de l'élément dans le tableau), la première position porte le numéro 0. Dans notre exemple, les indices vont de 0 a 24.**

**Le premier élément du tableau sera désigné par Note [0], le deuxième par Note[1],et le dernier par Note[24]. L'utilisation de Note [25] déclenchera une erreur.**

**L'élément qui se trouve à la position i sera désignée par : Note[i]**

# Les Tableaux

**Exemple :**

```
Variables: tableau Note[20],i,somme,n : entier
                moyenne : réel
début
pour i allant de 0 à 19 faire
  ecrire("donner une valeur :")                    //Remplir un Tableau
  lire(Note[i])
Finpour
pour i allant de 0 jusqu'à 19
  écrire(Note[i])                                  // Afficher un Tableau
Finpour
Fin
```

# Les Tableaux

**Remarque : "Les tableaux dynamiques"**

**Il arrive fréquemment que l'on ne connaisse pas à l'avance le nombre d'élément que devra comporter un tableau.**

**Pour résoudre ce problème :**

**En algorithmique on a la possibilité de déclarer le tableau sans préciser au départ son nombre d'éléments. Ce n'est que dans un second temps, au cours du programme, que l'on va fixer ce nombre :**

**Redim(T[n]), il faut préciser le nom du tableau et le nombre d'éléments, et a la fin on doit libérer l'espace réservé par liber(T).**

**Notez que tant qu'on n'a pas précisé le nombre d'éléments d'un tableau, d'une manière ou d'une autre, ce tableau est inutilisable.**

# Les Tableaux

**Exercice d'application:**

**on veut saisir des notes pour un calcul de moyenne, mais on ne sait pas combien il y aura de notes a saisir. L'algorithme sera :**

```
variables tableau Note[], moyenne, somme : réels
  i, n : entiers

début

écrire("entrer le nombre de notes à saisir : ")

lire(n)

/* allouer n nombres de types reels */

Redim(Note[n])

/* saisir les notes */

pour i allant de 0 à n-1 faire
  écrire(« donner une note : ")
  lire(Note[i])

finpour

/* effectuer la moyenne des notes */

somme ←0

pour i allant de 0 à n-1 faire

somme ←somme + Note[i]

finPour

moyenne = somme / n

/* affichage de la moyenne */

écrire("la moyenne des ",n," notes est :",moyenne)

/* liberer le tableau Notes */

libère(Notes)

fin
```

# Exercices

# Les Tableaux

**B- la recherche dans les tableaux**

***Recherche séquentielle***

**La méthode de recherche séquentielle consiste à comparer chaque élément du tableau avec l'élément recherché en allant du premier élément au dernier. Cette méthode est la plus simple qui puisse exister, mais aussi la plus lente.**

```
variables i ,n, x,pos : entier
                tableau Tab[]:entier
début
Pos <-  -1
i ←0
Tant que ( i < n et Tab[i] != x)
i ← i + 1
fintantque
    si (i<n) alors
    pos <- i
    finsi
   si( pos<>-1)
    Ecrire(" l'element existe est sa position :",pos)
    Sinon
    Ecrire("l'element n'existe pas")
    Finsi
fin
```

## B- la recherche dans les tableaux

### *Recherche séquentielle*

**La méthode de recherche séquentielle consiste à comparer chaque élément du tableau avec l'élément recherché en allant du premier élément au dernier. Cette méthode est la plus simple qui puisse exister, mais aussi la plus lente.**

# Les Tableaux

***Recherche dichotomique***

**La recherche dichotomique fait partie des méthodes de recherche rapide et on appelle recherche rapide les algorithmes dont la croissance du nombre d'opérations à faire, diminue avec le nombre d'éléments.**

**La grande particularité de cette méthode de recherche est que les éléments doivent être triés !**

**Le principe est de comparer la valeur recherchée X par celle du milieu du tableau T[milieu] ;**

- **Si T[milieu]=X on arrête car on a trouvé ce qu'on cherche,**
- **Si X<T[milieu], refaire la recherche sur la 1ère partie du tableau**
- **Si X>T[milieu], refaire la recherche sur la 2ème partie du tableau.**
- **Ainsi de suite**

# Les Tableaux

**Réponse:**

```
variables d, f, mil : entières
début
d ← 0
f ←taille – 1
Pos ← -1
Tantque (d<=f)
mil ←(d + f)/2
si (valRech = Tab[mil])
        pos← mil
si (valRech<Tab[mil])
        f ← mil-1
sinon
        d ← mil + 1
Fintantque

Fin
```

# Les Tableaux

**Le tri des tableaux**

**Trier un tableau revient à réorganiser une suite d'enregistrements, de telle sorte que les valeurs soient ordonnées.**

**On tri les éléments d'un tableau dans l'ordre croissant ou décroissant pour rendre plus efficaces les opérations de recherche et, par conséquent, les opérations d'insertion, et de suppression, etc.**

**Il existe plusieurs méthodes de tris :**

- **Tri par insertion**
- **Tri par sélection**
- **Tri par permutation**

# Les Tableaux

**Tri par sélection**

**Soit T un tableau de N éléments. On cherche le plus petit élément du tableau et on le place à la première position. Après, on cherche le plus petit dans les (N-1) qui reste et on le place en deuxième position et ainsi de suite.**

**Il faut :**

- **1 boucle pour parcourir le tableau et sélectionner tous les éléments ;**
- **1 boucle pour rechercher le minimum parmi les éléments non triés.**

# Les Tableaux

```
Variables i, j, min, tmp : entier
Début

    Pour i de 0 à N-2                    /* sélection d'un élément */
        min <- i
        Pour j de i+1 à N-1              /* recherche du minimum */
            Si t[ j ] < t[ min ] alors
                min <- j
            Fin si
        Fin pour
        tmp <- t[ i ]                    /* échange */
        t[ i ] <- t[ min ]
        t[ min ] <- tmp
    Fin pour
Fin
```

# Les Tableaux

**Tri par insertion**

**Chaque fois que le joueur prend une nouvelle carte, il l'introduit dans sa main de telle sorte qu'elle reste ordonnée. Le tri par insertion est basé sur cette idée d'insertion dans une liste ordonnée.**

**On considère les éléments les uns après les autres en insérant chacun à sa place parmi les éléments déjà triés.**

**Il faut :**

- **1 boucle pour parcourir le tableau et sélectionner l'élément à insérer ;**
- **1 boucle pour décaler les éléments plus grands que l'élément à insérer ;**
- **insérer l'élément**

# Les Tableaux

```
Variables i, j, mem: entier

Début

    Pour i de 1 à N-1                          /* sélection de l'élément à insérer*/
        mem <- t[ i ]
        j <- i
        Tant que j>0 et t[j-1]>mem             /* décalage des éléments plus grands */
            t[ j ] <- t[ j-1 ]
            j <- j - 1
        Fin tant que
        t[ j ] <- mem                          /* insertion */
    Fin pour
Fin
```

# Les Tableaux

**Tri par permutation**

**L'algorithme de permutation simple est basé sur le principe de la comparaison et de l'échange de couples d'éléments adjacents jusqu'à ce que tous les éléments soient triés.**

**Parcourir le tableau en comparant deux a deux les éléments successifs, permuté s'ils ne sont pas dans le bon ordre.**

**Répéter tant que des permutations sont effectuées.**

# Les Tableaux

```
Variables : tableau tab[] :entier
                                    taille ,i, j, tmp :
entiers
début
pour i allant de 2 à n faire
                pour j allant de n à i par –1 faire
                                    Si Tab[j-1]>Tab[j]
alors
tmp = Tab[j-1]
Tab[j-1] = Tab[j]
Tab[j] = tmp
                                    finsi
                finpour
finpour
fin
```

```
Amélioration du tri par permutation
Variables : tableau tab[] :entier
                                    taille ,i, j, tmp :
entiers
                                    permute :
boolean
début
Repeter
Permute= faux
                pour j allant de n à i par –1 faire
                                    Si Tab[j-1]>Tab[j]
alors
tmp = Tab[j-1]
Tab[j-1] = Tab[j]
Tab[j] = tmp
Permute=vrai
                                    finsi
                finpour
Jusqu’ à (permute=faux)
Fin
```

# Les Tableaux

## 3- Tableaux à deux dimensions

**Pour représenter les matrices dans un ordinateur, un tableau ne suffit pas, puisque chaque ligne de la matrice est en effet un tableau, donc une matrice de taille n×k peut être représenté par n tableaux de k éléments chacun.**

**L'informatique nous offre la possibilité de déclarer des tableaux dans lesquels les valeurs ne sont pas repérées par un seul indice, mais par deux indices.**

$$\text{Matrice de dimension } m \times n;\ A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

# Les Tableaux

**1- Remplir et afficher une matrice (tableaux à deux dimensions)**

**Pour déclarer un tableau en algorithmique :**

**tableau T[nbr-ligne][ nbr-col] : type**

**Cette déclaration alloue donc en mémoire pour l'objet T un espace de 30 × 4 octets.**

**Les éléments d'un tableau sont toujours numérotés de 0 à nbr_ligne -1 et nbr_col-1.**

**T[0][0] est le premier élément de la matrice T, qui se trouve à l'intersection de la ligne 0 et la colonne 0.**

**T[i][j] est l'élément de la matrice qui se trouve a l'intersection de la ligne i et la colonne j.**

# Les Tableaux

```
variable Tableau T[10][10],i,j,n,m : entiers
début
écrire("donner le nombre de ligne de la matrice :")
lire(n)
écrire("donner le nombre de colonne de la matrice :")
lire(m)
pour i allant de 0 à n faire                    //Lecture d'une matrice(remplissage)
pour j allant de 0 à m faire
ecrire("donner une valeur:")
lire(matrice[i][j])
finpour
fipour
fin
```

# Les Tableaux

```
variables Tableau Mat[10][10],i,j,n,k : entier
début
//demander la taille n et m
pour i allant de 1 à n faire
pour j allant de 1 à k faire
écrire(Mat[i][j]," ")
finpour
    écrire("\n") /* retour à la ligne */
finpour
fin
```

**//affichage d'une matrice :**

# Les Tableaux

## 2- Quelques propriétés

Matrice de dimension $m \times n$; $A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$

Exemple:

Matrice de dimension 2 x 3 $\begin{matrix} 3 & 1 & 0 \\ 6 & 9 & 4 \end{matrix}$

**Matrice nulle :**

Tous ses éléments sont nuls

**Matrice carrée d'ordre n :** nombre de lignes = nombre de colonnes = n

Exemple:

matrice carrée de taille 3 $\begin{matrix} 3 & 1 & 4 \\ 8 & 5 & 2 \\ 1 & 0 & 2 \end{matrix}$

# Les Tableaux

**Matrice Diagonal :**

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{mn} \end{pmatrix}$$

Exemple:

Matrice diagonale de taille 3

$$\begin{matrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 2 \end{matrix}$$

**Matrice identitée :**

$$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

# Les Tableaux

**Matrice triangulaire supérieure :**

Pour i > j on a T[i][j]=0

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{mn} \end{pmatrix}$$

Exemple:

Matrice triangulaire supérieure de taille 3

$$\begin{matrix} 3 & 2 & 1 \\ 0 & 5 & 8 \\ 0 & 0 & 2 \end{matrix}$$

# Les Tableaux

**Matrice triangulaire inférieure :**

Pour i < j on a T[i][j]=0

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Exemple:

Matrice triangulaire inférieure de taille 3

$$\begin{matrix} 3 & 0 & 0 \\ 1 & 5 & 0 \\ 3 & 6 & 2 \end{matrix}$$

# Les Tableaux

## Matrice Transposée :

La transposée d'une matrice $A$ s'obtient en remplaçant les lignes de la matrice par ses colonnes. Si la matrice $A$ est de dimension $m \times n$, la transposée $AT$, sera de dimension $n \times m$.

$$A^T = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}$$

$A^T[i][j] = A[j][i]$

Matrice de taille 3

$$\begin{matrix} 3 & 2 & 1 \\ 4 & 5 & 8 \\ 9 & 0 & 2 \end{matrix}$$

Matrice transposée

$$\begin{matrix} 3 & 4 & 9 \\ 2 & 5 & 0 \\ 1 & 8 & 2 \end{matrix}$$

# Les Tableaux

**Matrice symétrique**

Lorsque les éléments de cette matrice sont symétriques par rapport à la diagonale.

Pour i # j on a T[i][j]=T[j][i]

**Exemple :**

La matrice suivante est donc symétrique :

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 0 & 5 \\ 3 & 5 & 6 \end{pmatrix}$$

**Trace d'une matrice :**

*Trace d'une matrice carrée d'ordre n, $A = (a_{ij})$ (notée tr (A)) :*

Somme des éléments de la diagonale principale i.e. $\mathrm{tr}(A) = a_{11} + a_{22} + \cdots + a_{nn}$

**Exemple:**

Matrice de taille 3

$$\begin{matrix} 3 & 2 & 1 \\ 4 & 5 & 8 \\ 9 & 0 & 2 \end{matrix}$$

Trace de la matrice = 3+5+2=10

# Exercices