Algorithmique et Structures de Données 1
CH4: Les tableaux

Enseignant: Fethi Mguis
Sections: LGLSI1/LIRIS1

A.U: 2023/2024

1 Tableaux à une dimension

1.1 Définition

Un tableau T est une variable structurée formée d'un nombre entier N de variables simples de même type, qui sont appelées les composantes du tableau. Le nombre de composantes N est alors la dimension du tableau.
On dit encore que T est un vecteur de dimension N.

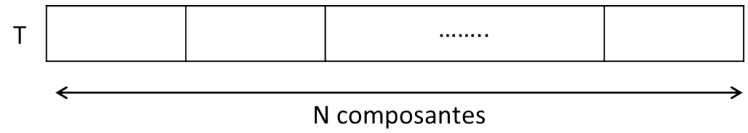

Figure 1 – Exemple d'un tableau.

1.2 Utilité

- Un tableau est une structure de données constituée d'un nombre fini d'éléments de même type.
- Lorsque plusieurs données de même type, généralement destinées au même traitement doivent être accessibles le long d'un programme, on propose d'utiliser la structure d'un tableau.

1.3 Composantes

Nom : identificateur d'un tableau.
Type_élément : Les éléments d'un tableau sont caractérisés par leur type (entier, réel, caractère,...).
Indice : Tout type dont les éléments possèdent un successeur (les types scalaires), généralement de type entier.

1.4 Déclaration

Nom_tab : **Tableau [premind . . deuxind] de type_élément**

Exemples

T1 : **Tableau [1..50] d'entier**
T2 : **Tableau [1..20] de réel**
T3 : **Tableau [1..20] de caractère**

Remarque Il est également possible de définir un type tableau comme dans l'exemple suivant :

```
Const :
  nmax : 50
Type :
  tab : Tableau [1..nmax] d'entier
Var :
  T : tab
```

1.5 Accès aux composantes d'un tableau

Considérons un tableau T de dimension N
- L'accès au premier élément du tableau se fait par T[1]
- L'accès au dernier élément du tableau se fait par T[N]

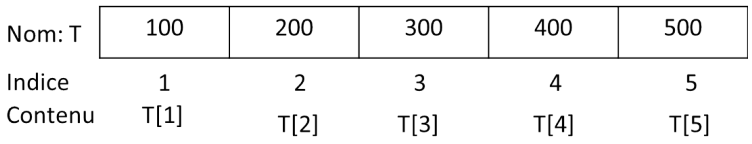

FIGURE 2 – Accès aux éléments d'un tableau.

Exemple

1.6 Chargement d'un tableau

```
Procédure CHARGEMENT(T : tab ; N :entier)
Var
   i : Entier
Début
  Pour i de 1 à N faire
    Écrire ("T[",i,"] :")
    Lire (T[i])
  Fin Pour
Fin
```

1.7 Affichage du contenu d'un tableau

```
Procédure AFFICHAGE(T : tab ; N :entier)
Var
   i : Entier
Début
  Pour i de 1 à N faire
    Écrire (T[i]))
  Fin Pour
Fin
```

1.8 Méthodes de tri dans un tableau

1.8.1 Tri par sélection (par minimum)

Principe : Le principe de cette méthode est simple. Elle consiste à :
- Chercher l'indice du plus petit élément du tableau T[1..n] et permuter l'élément correspondant avec l'élément d'indice 1 ;
- Chercher l'indice du plus petit élément du tableau T[2..n] et permuter l'élément correspondant avec l'élément d'indice 2 ;
- ...
- Chercher l'indice du plus petit élément du tableau T[n-1..n] et permuter l'élément correspondant avec l'élément d'indice n-1 ;

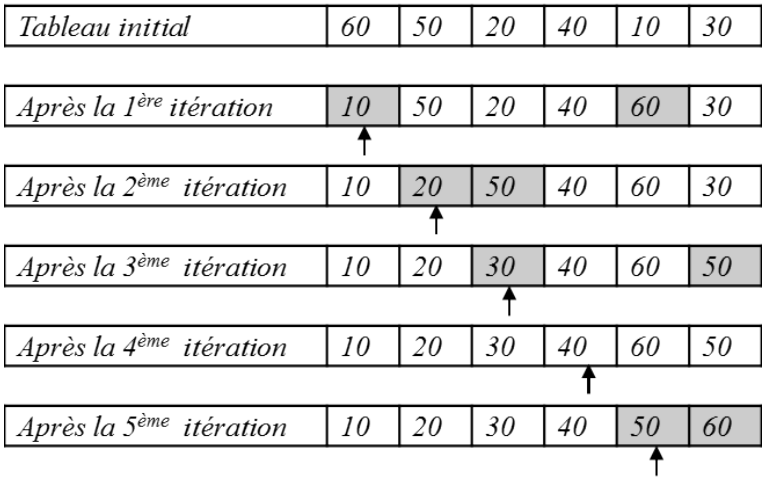

FIGURE 3 – Tri par sélection.

```
Fonction IND_MIN(T : tab ; N,d :entier) : Entier
Var
   posmin,j : Entier
Début
  posmin←d
  Pour j de d+1 à N faire
    Si (T[j]<T[posmin]) Alors
      posmin←j
    Fin Si
  Fin Pour
  IND_MIN←posmin
Fin
Procédure TRISELECTION(T : tab ; N :entier)
Var
   i, j, aux, posmin : Entier
Début
  Pour i de 1 à N-1 faire
    posmin←IND_MIN (T,N,i)
    Si ( posmin<>i)) Alors
      aux←T[i]
      T[i]←T[posmin]
      T[posmin]←aux
    Fin Si
  Fin Pour
Fin
```

1.8.2 Tri à bulles

Principe Cet algorithme porte le nom de tri bulle car, petit à petit, les plus grands éléments du tableau remontent, par le jeu des permutations, enfin de tableau. La méthode de tri à bulles consiste à répéter le traitement suivant :
Parcourir les éléments du tableau de 1 à (n-1) ; si l'élément i est supérieur à l'élément (i+1) , alors on les permute.
Le programme s'arrête lorsque aucune permutation n'est réalisable après un parcours complet du tableau.

```
Procédure TRIBULLES(T : tab ; N :entier)
Var
   i, aux, permute : Entier
Début
  Répéter
    Permute←0
    Pour i de 1 à N-1 faire
      Si ( T[i] > T[i + 1]) Alors
        aux←T[i]
        T[i]←T[i+1]
        T[i+1]←aux
        permute←1
      Fin Si
    Fin Pour
  Jusqu'à (permute=0)
Fin
```

Tableau initial	*50*	*30*	*20*	*40*	*10*
1ère étape	*30*	*50*	*20*	*40*	*10*
	30	*20*	*50*	*40*	*10*
	30	*20*	*40*	*50*	*10*
	30	*20*	*40*	*10*	*50*
2ème étape	*20*	*30*	*40*	*10*	*50*
	20	*30*	*10*	*40*	*50*
3ème étape	*20*	*10*	*30*	*40*	*50*
4ème étape	*10*	*20*	*30*	*40*	*50*

FIGURE 4 – Tri à bulles.

1.8.3 Tri par insertion

Méthode : Trier le tableau de gauche à droite en insérant à chaque fois l'élément i+1 dans le tableau (déjà trié) des i premiers éléments.

- Comparer et permuter si nécessaire T[1] et T[2] de façon à placer le plus petit dans la case d'indice 1
- Comparer et permuter si nécessaire l'élément T[3] avec ceux qui le précèdent dans l'ordre (T[2] puis T[1]) afin de former une sous-liste triée T[1..3]
- ...
- Comparer et permuter si nécessaire l'élément T[n] avec ceux qui le précèdent dans l'ordre (T[n-1] puis T[n-2]) afin de former un tableau trié .

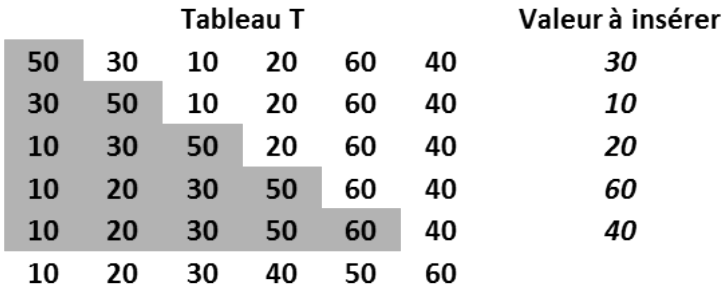

FIGURE 5 – Tri par insertion.

```
Procédure TRIINSERTION(T : tab ; N :entier)
Var
   i, j, aux : Entier
Début
  Pour i de 2 à N faire
    aux←T[i]
    j←i
    Tant que ( (T[j − 1] > aux) ET (j > 1)) faire
      T[j]←T[j-1]
      j←j-1
    Fin Tq
    T[j]←aux
  Fin Pour
Fin
```

1.9 Méthodes de recherche dans un tableau

1.9.1 La recherche séquentielle

Problème Déterminer la première position d'une valeur donnée dans un tableau de N élément. Résoudre ce problème en utilisant la notion de procédures/Fonctions.

```
Fonction RECH_SEQ(T : tab ; N, val :entier) : entier
Var
   i, pos : entier
Début
  pos←0
  i←1
  Tant que (i <= N et pos = 0) faire
    Si (T[i] = val) Alors
      pos←i
    Sinon
      i←i+1
    Fin Si
  Fin Tq
  RECH_SEQ←pos
Fin
```

1.9.2 La recherche dichotomique

Problème Déterminer la première position d'une valeur donnée dans un tableau de N éléments triés dans l'ordre croissant.

Principe Le principe est de décomposer le tableau T en deux sous tableaux. Trois cas peuvent se produire :

- Si la valeur recherchée = T[milieu] alors val est trouvé et la recherche est terminée.
- Si la valeur recherchée < T[milieu] alors on va chercher val dans la partie gauche du tableau T.
- Si la valeur recherchée > T[milieu] alors on va chercher val dans la partie droite du tableau T.

On poursuit la recherche tant que la dimension de sous tableau reste valide est tant que T[milieu] est différent de la valeur recherchée.

```
Fonction RECH_DICH(T : tab ; N, val :entier) : entier
Var
   i, pos, mil, inf, sup : entier
Début
   pos←0
   inf←1
   sup←N
   Tant que (inf <= sup et pos = 0) faire
      mil←(inf + sup) div 2
      Si (T[mil] = val) Alors
         pos←mil
      Sinon
         Si (val < T[mil]) Alors
            sup←mil - 1
         Sinon
            inf←mil + 1
         Fin Si
      Fin Si
   Fin Tq
   RECH_DICH←pos
Fin
```

2 Tableaux à deux dimensions

2.1 Définition

Un tableau à deux dimensions A et à interpréter comme un tableau (unidimensionnel) de dimension L dont chaque composante est un tableau (unidimensionnel) de dimension C. On appelle L le nombre de lignes du tableau et C le nombre de colonnes du tableau. Un tableau à deux dimensions contient L*C composantes.

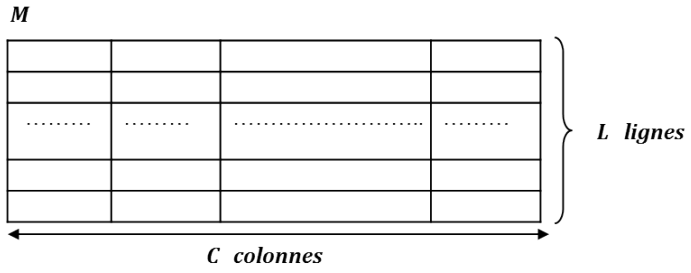

FIGURE 6 – Représentation d'une matrice.

2.2 Déclaration

Nom_Mat : **Tableau [premind . . deuxind][premind . . deuxind] de type_élément**

Exemples

M1 : **Tableau [1..50][1..30] d'entier**
M2 : **Tableau [1..20][1..20] de réel**
M3 : **Tableau [1..20][1..50] de caractère**

Remarque Il est également possible de définir un type matrice comme dans l'exemple suivant :

```
Const :
  lmax : 50
  cmax : 50
Type :
  Mat : Tableau [1..lmax][1..cmax] d'entier
Var :
  M : Mat
```

2.3 Accès aux composantes d'une matrice

Considérons un tableau M de L lignes et C colonnes.

- Les indices du tableau varient de 1 à L, respectivement de 1 à C.
- La composante de la N^{ieme} ligne et M^{ieme} colonne est notée : A[N][M].

Syntaxe

```
Nom_Matrice[ligne][colonne]
```

Exemple Considérons une matrice de 3 lignes et 4 colonnes

	1	2	3	4
1	A[1][1]	A[1][2]	A[1][3]	A[1][4]
2	A[2][1]	A[2][2]	A[2][3]	A[2][4]
3	A[3][1]	A[3][2]	A[3][3]	A[3][4]

FIGURE 7 – Accès aux éléments d'une matrice.

2.4 Chargement d'une matrice

```
Algorithme Chargement
Var
   M : Tableau [1.. 3][1..4] d'entier
i ,j : Entier
Début
  Pour i de 1 à 3 faire
    Pour j de 1 à 4 faire
      Écrire("M[", i, ", ", j, "] :")
      Lire(M [i, j])
    Fin Pour
  Fin Pour
Fin
```

2.5 Affichage du contenu d'une matrice

```
Algorithme Afficher
Var
   M : Tableau [1.. 3][1..4] d'entier
i ,j : Entier
Début
  Pour i de 1 à 3 faire
    Pour j de 1 à 4 faire
      Écrire(M [i, j])
    Fin Pour
  Fin Pour
Fin
```

Exemple 1 Soient M1 et M2 deux matrices à n lignes et m colonnes. On veut écrire une procédure qui calcule les éléments de la matrice M3 = M1 + M3

$$\underset{M1}{\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}} + \underset{M2}{\begin{bmatrix} a' & b' & c' & d' \\ e' & f' & g' & h' \\ i' & j' & k' & l' \end{bmatrix}} = \underset{M3}{\begin{bmatrix} a+a' & b+b' & c+c' & d+d' \\ e+e' & f+f' & g+g' & h+h' \\ i+i' & j+j' & k+k' & l+l' \end{bmatrix}}$$

FIGURE 8 – Calcul de la somme de deux matrices.

Rappel

```
Procédure SOMME(M1 , M2 : MAT; VAR M3 : MAT;n, m : entier)
Var
   i ,j : Entier
Début
   Pour i de 1 à n faire
      Pour j de 1 à m faire
         M3[i, j]←M1[i, j] + M2[i, j]
      Fin Pour
   Fin Pour
Fin
```

Exemple 2 Ecrire un algorithme qui effectue la transposition tA d'une matrice A de dimensions N et M en une matrice de dimensions M et N. La matrice A sera transposée par permutation des éléments. Résoudre ce problème en utilisant la notion de procédures/Fonctions.

$$tA = t\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} = A\begin{bmatrix} a & e & i \\ b & f & j \\ c & g & k \\ d & h & l \end{bmatrix}$$

FIGURE 9 – Détermination de la transposée d'une matrice.

```
Fonction SAISIE_TAILLE() : Entier
Var
   nb : entier
Début
   Répéter
      Ecrire("Donner la taille de T :") Lire(nb)
   Jusqu'à (nb > 1 et nb <= nmax)
   SAISIE_TAILLE←nb
Fin
Procédure REMP(A : MAT ; N, M : entier)
Var
   i, j : entier
Début
   Pour i de 1 à N faire
      Pour j de 1 à M faire
         Ecrire("A [" , i , ", " , j , "] :")
         Lire(A[i, j])
      Fin Pour
   Fin Pour
Fin
Procédure AFF(A : MAT ; N, M : entier)
Var
   i, j : entier
Début
   Pour i de 1 à N faire
      Pour j de 1 à M faire
         Ecrire(A[i, j])
      Fin Pour
   Fin Pour
Fin
```

```
Procédure TRANS(A : MAT ; N, M : entier)
Var
   i, j, Dmax, aux : entier
Début
   Si (N > M) Alors
      Dmax←N
   Sinon
      Dmax←M
   Fin Si
   Pour i de 1 à Dmax faire
      Pour j de 1 à i faire
         aux←A[i, j]
         A[i , j]←A[j , i]
         A[j , i]←aux
      Fin Pour
   Fin Pour
Fin
Algorithme CHANGEMENT
Const
   Nmax : 50
Var
   MAT : Tableau [1.. Nmax, 1.. Nmax] d'entier
Début
   A : MAT
   N , M : entier
Fin
Ecrire (" Saisie des tailles")
N←SAISIE_TAILLE()
M←SAISIE_TAILLE()
Ecrire(" Chargement de A ")
REMPLIR(A, N, M)
Ecrire("Affichage de A avant transposition")
AFFICHE(A, N, M)
TRANS(A, N, M)
Ecrire("Affichage de A après transposition")
AFFICHE(A, M, N)
```