Université A. Mira Béjaia
Faculté de Sciences Exactes
Département Mathématiques
L1 Math (tronc commun)
2025/2026

Module : Algorithmique et Structures de Données 1 (ASD1)

Chapitre 5 : Les Tableaux

1. Introduction

Une variable de type structuré regroupe sous un même nom plusieurs valeurs appartenant à des types constituants déjà définis. Ces types constituants peuvent à leur tour être structurés.

2. Les tableaux à une dimension

2.1. Définition

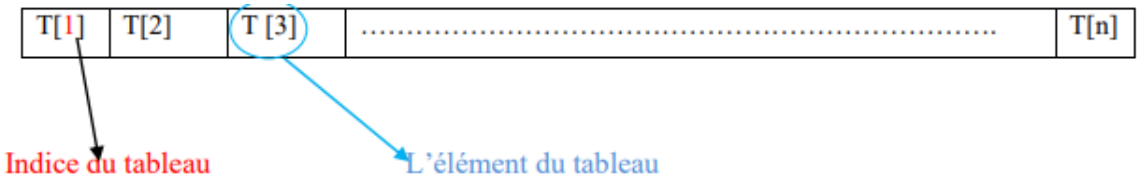
Un tableau est un ensemble de données qui sont toutes du même type et qui possèdent un identificateur unique (nom du tableau) et se identifiés dans le tableau par leur numéro d'indice.

- Chaque élément d'un tableau est accessible par un indice, qui est un entier.
- Selon les langages de programmation, le premier indice est soit 0 (Le C), soit 1 (Pascal)
- Dans ce cous, on va considérer que le premier indice du tableau est 1.

Un tableau est caractérisé par :

- Un nom (le nom du tableau).
- Une longueur (le nombre des éléments du tableau).
- Un type (le type des éléments du tableau).
- Une dimension (1, 2, 3, etc).

On peut représenter un tableau par un ensemble de cases repérées par leurs indices (positions, rangs) dans le tableau.



2.2. Déclaration

La syntaxe de déclaration d'une variable de type ***tableau***, on peut utiliser l'un de deux méthodes de déclaration :

Type nom_type_tableau = **Tableau** [valeur_indice_minimum**..**valeur_indice_maximum] **de** type_des_éléments ;

Var nom_tableau : nom_type_tableau ;
Ou directement :
Var nom_tableau : **Tableau** [valeur_indice_minimum**..**valeur_indice_maximum] **de** type_des_éléments ;

- **Syntaxe en langage C :**

Type_elements **nom_tableau** [Taille] ;

La Taille du tableau est **le nombre de ses éléments**. Elle ne peut être une variable. Elle doit être une **constante définie** avant **ou une valeur**.

Exemple:

Type Tab = **Tableau** [1..10] **de** entier; //Défini un **nouveau type** appelé ***Tab***, qui est un tableau **de 10 entiers.**

Var T : Tab ; // Déclare une variable **T de type Tab**.

- **En langage C**

int T[10] ;

2.3. Les opérations sur les tableaux

a. L'accès aux éléments

L'accès aux éléments d'un tableau se fait en donnant **le nom de la variable tableau** suivi entre deux crochets ([ ]) de **l'indice de l'élément**.

Le fait que les éléments constituant un tableau soient indicés permet de parcourir tous les éléments avec **une boucle**. La boucle **« Pour »** est généralement la boucle la plus adaptée, puisque l'on **connaît le nombre de fois** qu'on doit effectuer le traitement (une fois par élément).

Exemple :

Type Tab = **tableau** [1..100] de entier ;
Var T : Tab ;
- T [1] : désigne le premier élément du tableau T.
- T [100] : désigne le dernier élément du tableau T.
- T [101] : n'est pas un élément du tableau T.

b. Accès en affichage

Ecrire (T[i]) : le contenu du tableau à l'indice i est affiché à l'écran.

Exemple : pour afficher le contenu du tableau en entier :

Pour i allant de **1** jusqu'à **100** faire
    Ecrire (T[i]) ;
Finpour;

En lngage C

```
For (i = 0 ; i < = (N-1) ; i++)
        printf ("%d", &T[i]) ;
```

c. Accès en initialisation ou modification:

1. **T[i] ←valeur :** la valeur est placée dans le tableau à l'indice **i**.
2. **Lire (T[i]) :** la valeur entrée par l'utilisateur est enregistrée dans le tableau à l'indice **i**.

Exemple : pour remplir le tableau en entier :

En langage C

```
For (i =0 ; i < = (N-1) ; i++)
    scanf ("%d", &T[i]) ;
```

Pour i allant de **1** jusqu'à **100** faire
  Lire (T[i]) ;
Finpour.

Exemple : écrire un algorithme permettant remplir un tableau de N éléments et de l'afficher.

Algorithme lire_tableau ;
Type Tab = tableau [1..100] de entier ;
Var T : Tab ;
N, i : entier ;
Debut
  Repeter
    Ecrire ('introduire le nombre des éléments du tableau') ;
    Lire (N) ;
  Jusqua ((N>=1) et (N<=100)) ;
Ecrire ('introduire les éléments du tableau T') ;
Pour i allant de 1 jusqua N **faire**
  Lire (T[i]) ;
FinPour ;
Pour i allant de 1 jusqua N **faire**
  Ecrire (T[i]) ;
FinPour ;

Fin.

Exercice : Ecrire un algorithme qui permet de lire un tableau de 100 éléments entiers, puis de calculer et d'afficher la moyenne de ces éléments.

```
Algorithme  moyenne_tableau ;
Type  tab100= tableau[1..100] de entier ;
Var   tab :tab100 ;
i, somme : entier ;
moy :réel ;
début
|     pour i allant de 1 jusqu'à 100 faire
|     |       lire(tab[i]) ;
|     finpour ;
|       somme←0 ;
|     pour i allant de 1 jusqu'à 100 faire
|     |       somme←somme + tab[i];
|     finpour ;
|       moy←(somme/100) ;
|       ecrire('moyenne=', moy) ;
|
fin.
```

3. Manipulation de tableaux à une seule dimension

3.1. Somme de deux tableaux T1 et T2

La somme de deux tableaux (vecteurs) T1 et T2 de N éléments, donne un vecteur T de même taille (nombre d'éléments).

```
Algorithme somme ;
Var T1, T2, T : tableau [1..100] de réel ;
N, i : entier ;
Debut
    Repeter
      Ecrire ('introduire le nombre des éléments du tableau') ;
      Lire (N) ;
    Jusqua ((N>=1) et (N<=100)) ;
    Pour i allant de 1 jusqua N faire
     Lire (T1[i]) ;
    FinPour ;
    Pour i allant de 1 jusqua N faire
       Lire (T2[i]) ;
    FinPour ;
    Pour i allant de 1 jusqua N faire
      T[i] ← T1[i] + T2[i] ;
    FinPour
    Pour i allant de 1 jusqua N faire
     Ecrire (T[i]) ;
    FinPour
Fin.
```

3.2. Un algorithme de recherche

Exercice :

Ecrire un algorithme qui permet de lire un tableau **de taille n**, donnée par l'utilisateur, puis de chercher **une valeur val,** donnée par l'utilisateur aussi, dans ce tableau ainsi **que son indice.**

- **Paramètres d'entrée** : un tableau de **n** entiers et une valeur **val**.
- **Paramètres de sortie** : le booléen **trouve** et l'entier **indice**.
- **Spécification** : le booléen **trouve** doit être à vrai si **val** se trouve dans le tableau. La valeur d'indice vaut alors le plus petit indice de la case contenant la valeur **val** dans le tableau.

```
Algorithme    Recherche ;
Type  tab100 = tableau [1..100] de entier ;
Var             i, indice, n, val : entier ;   trouve : booléen ;     tab :tab100 ;
début :
|       Ecrire('donner la taille du tableau') ;
|     Lire(n) ;
|       pour  i  allant de  1  jusqu'à  n  faire
|       | lire (tab[i]) ;
|       | finpour ;
|       Ecrire ('donner moi la valeur à trouver') ;
|       Lire(val) ;
|       i ← 1 ;
|       trouve ← faux ;
|       tantque  ((i≤n) et non trouve) faire
|       |       si  (val=tab[i])  alors
|       |       |                       début
|       |       |                       |       trouve ← vrai ;
|       |       |                       |       indice ← i ;
|       |       |                       |fin ;
|       |       |sinon
|       |       |       i← i+1 ;
|       |       |finsi ;
|       |fintantque ;
|       si  (trouve = vrai)  alors
|       |                       début
|       |                       |       ecrire ('valeur trouvé') ;
|       |                       |       ecrire ('indice=',indice);
|       |                       |fin ;
|       |sinon
|       |       ecrire( 'valeur n'existe pas dans le tableau') ;
|       |finsi ;
|fin.
```

3.3. Un algorithme de tri

Trier un tableau c'est réorganiser ses éléments suivant un ordre déterminé croissant ou décroissant pour les nombres ou les caractères.

Exercice :

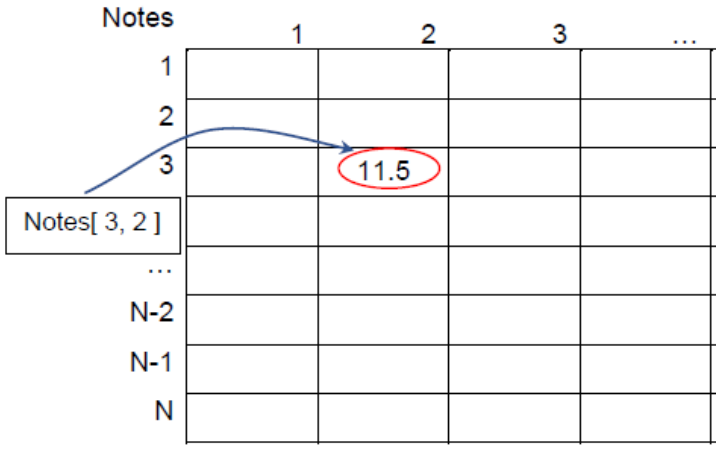
Ecrire un algorithme qui remplit un tableau de **10 entiers** et le trier selon l'ordre croissant de ces éléments.

```
Algorithme    tri_tableau ;
Type  tab100 = tableau [1..10] de réel ;
Var              i, j : entier ;   perm :réel ;    tab : tab10 ;
début :
|       pour  i  allant de  1  jusqu'à  10  faire
|       | lire (tab[i]) ;
|       finpour ;
|       pour  i  allant de  1  jusqu'à  9  faire
|       |      pour  j  allant de  i+1  jusqu'à  10  faire
|       |       |          si  (tab[j]<tab[i])  alors
|       |       |          |                          début
|       |       |          |                          |     perm ← tab[i] ;
|       |       |          |                          |     tab[i] ← tab[j];
|       |       |          |                          |      tab[j] ← perm;
|       |       |          |                          | fin ;
|       |       |          finsi ;
|       |       |
|       |       finpour ;
|       |
|       finpour ;
|       pour  i  allant de  1  jusqu'à  10  faire
|       |     ecrire (tab[i],' ') ;
|       finpour ;
|
fin.
```

4. Les tableaux à deux dimensions (les matrices)

4.1. Définition

L'algorithmique nous offre la possibilité de déclarer et d'utiliser des tableaux dans lesquels les valeurs ne sont pas repérées par un seul indice comme les tableaux à une seule dimension, mais par deux indices, le premier indice sert à représenter les lignes et le second indice pour les colonnes. Ce type de tableau est appelé Matrice. Le schéma ci-dessous représente un tableau Notes à deux dimensions.



4.2. Déclaration

La syntaxe de déclaration d'une variable de type ***matrice*** peut être l'une des deux suivantes :

Type nom_type_tableau = **Tableau**
[valeur_indice_ligne_minimum**..**valeur_indice_ligne_maximum , valeur_indice_colonne_minimum**..**valeur_indice_colonne_maximum] **de** type_des_éléments;
Var matrice : nom_type_tableau ;
Ou directement :

Var matrice : **Tableau** [valeur_indice_ligne_minimum**..**valeur_indice_ligne_maximum , valeur_indice_colonne_minimum**..**valeur_indice_colonne_maximum] **de** type_des_éléments;

- **En langage C :**

Type_elements identificateur_tableau [nombre_lignes][nombre_colonnes] ;

Exemple : matrice (tableau à deux dimensions) :
Type matrice = **tableau** [1..10,1..20] **de** réel ;
Var M : matrice ;

- **En langage C :**

float M [10][20] ;

4.3. Accès aux éléments d'un tableau à deux dimensions

Pour accéder aux éléments d'une matrice, il suffit de donner entre crochets l'indice de chaque dimension séparée par une virgule.

Nom_du_matrice [i, j] ;

Exemple : dans la matrice M précédente :

M [2,3] ← 3 signifie mettre la valeur 3 dans la case de numéro de ligne 2 et de numéro de colonne 3.

En considérant le cas où x est une variable de type réel, x ← M [2,3] signifie mette 3 dans x.

- **En langage C :**

M [2][3] = 3;

4.4. Lecture et affichage d'une matrice

L'algorithme suivant permet de lire et d'écrire une matrice de **n lignes et m colonnes** :

Algorithme lire_afficher_matrice;
Type Mat = tableau [1..50, 1..50] de entier ;
Var A : Mat ;
N, M, i : entier ;
Debut
Repeter
Ecrire ('introduire le nombre de ligne de la matrice') ;
Lire (N) ;
Jusqua ((N>=1) et (N<=50)) ;
Repeter
Ecrire ('introduire le nombre de colonnes de la matrice') ;
Lire (M) ;
Jusqua ((M>=1) et (M<=50)) ;
Pour i allant de 1 jusqua N **faire**
Pour j allant de 1 jusqua M **faire**
Lire (A[i,j]) ;
FinPour ;
FinPour ;

```
  Pour i allant de 1 jusqua N faire
     Pour j allant de 1 jusqua M faire
       Ecrire(A[i,j]) ;
     FinPour ;
  FinPour ;
Fin.
```

Exercice : Soit **A et B deux matrices** à valeurs entières. Ecrire un algorithme qui calcule et affiche la somme **des deux matrices**.

```
Algorithme   somme ;
     Type  matrice = tableau [1..5,1..3] de entier ;
     Var   A,B,C :matrice ;   i,j : entier ;
début :
|      ecrire ('introduire  les éléments de A') ;
|       Pour i allant de 1 jusqu'à  5  faire
|       |     Pour   j  allant de 1 jusqu'à  3 faire
|       |     |    lire (A [i,j]) ;
|       |     |Finpour ;
|       |Finpour ;
|
|      ecrire('introduire  les éléments de B') ;
|       Pour i allant de 1 jusqu'à  5  faire
|       |     Pour   j  allant de 1 jusqu'à  3 faire
|       |     |    lire (B [i,j]) ;
|       |     |Finpour ;
|       |Finpour ;
|
|       Pour i allant de 1 jusqu'à  5  faire
|       |     Pour   j  allant de 1 jusqu'à  3 faire
|       |     |    C[i,j] ← A [i,j]+ B[i,j] ;
|       |     |     ecrire(C[i,j]) ;
|       |     |Finpour ;
|       |Finpour ;
|fin.
```

5. Les tableaux comme paramètres dans les actions paramétrées

Dans les sous-programmes, si on veut passer des variables de type tableau comme paramètres à une fonction ou procédure, il faut définir au préalable les types tableaux utilisés pour la déclaration des paramètres formels.

Exemple :

```
Algorithme    exemple ;
        Type  tab1d = tableau [1..100] de entier ;
        Var   tab : tab1d ;
        Procedure   lecture_tableau(var  tab1 :tab1d) ;
var   i :entier ;
        début
        |       pour  i  allant de  1  jusqu'à  10  faire
        |       |     lire (tab1[i]) ;
        |       |finpour ;
        |fin ;
début
   |    lecture_tableau(tab) ;
   |
   |fin.
```