


# Architecture des ordinateurs & Algorithmique

## Partie II : Algorithmique

## Chapitre 5 : Tableaux

1ère Année Cycle Préparatoire Intégré / Semestre 1

ENSA Khouribga

Pr. DARGHAM Abdelmajid

Année universitaire : 2018/2019

# Sommaire

- Introduction
- Définition d'un tableau
- Utilité des tableaux
- Représentation mémoire d'un tableau
- Tableau monodimensionnel
- Tableau à deux dimensions
- Tableau dynamique
- Traitements sur les tableaux
- Exercices corrigés

# Introduction

- Lorsqu'un ingénieur veut **résoudre un problème d'ingénierie**, il est important qu'il soit en mesure de **visualiser les données et les résultats** associées à son problème.
- Dans certains cas, la **donnée est simple** et consiste en une **unique valeur individuelle**, comme par exemple le **rayon d'un cercle** qui est un **simple nombre**.
- Dans d'autres cas, la **donnée** peut consister en une **paire de valeurs**, comme par exemple les **coordonnées planes** (x, y) d'un point.

# Introduction

- Mais dans la plupart des cas, il est indispensable de **manipuler un ensemble de valeurs similaires**.
- Par exemple, supposez que l'on a un ensemble de **100 mesures de température**, et que l'on voudrez calculer la **température moyenne**.
- Évidemment, l'utilisation de **100 variables de noms différents** pour représenter les **100 différentes mesures** est **inefficace**.
- Nous avons besoin d'un **moyen pour manipuler un groupe de valeurs** en utilisant un **nom unique**.

# Définition d'un tableau

- Une solution à ce problème consiste à utiliser une **structure de données** appelée « **Tableau** ».
- Un **tableau** est une **structure de données** formée d'une **collection d'éléments** de **même type**.
- Chaque **élément d'un tableau** est **accessible** par un **indice** (ou **rang**), qui est un entier.
- Selon les langages de programmation, le premier indice est soit 0 (C/Java), soit 1 (Pascal).
- Dans ce cours, on va considérer que le **premier indice d'un tableau** est **1**.

# Définition d'un tableau

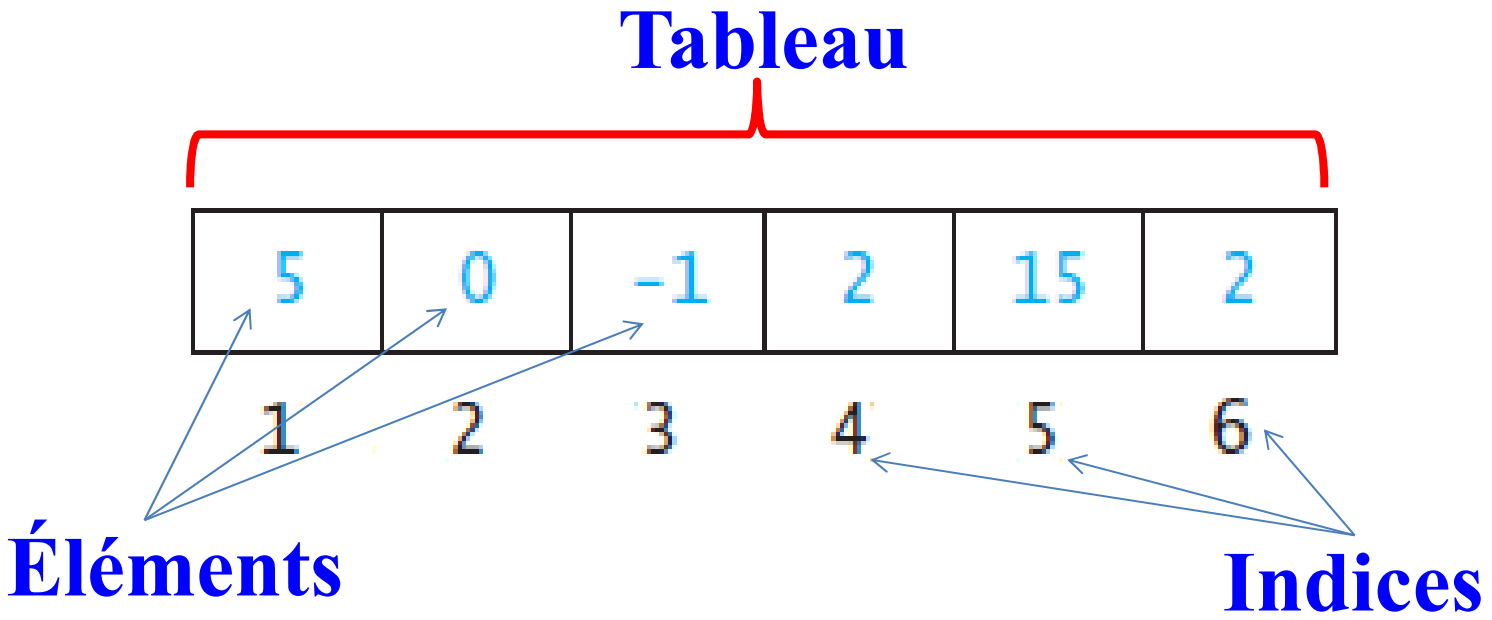
- Un **tableau** est caractérisé par :
  - Un **nom** (le ***nom du tableau***).
  - Une **longueur** (le ***nombre des éléments du tableau***).
  - Un **type** (le ***type des éléments du tableau***).
  - Une **dimension** (1, 2, 3, etc).
  - Une **taille mémoire** (le ***nombre d'octets occupés par le tableau***). Elle est égale à la longueur du tableau multipliée par la taille mémoire du type.

# Utilité des tableaux

- On utilise un **tableau** pour représenter des **collections d'éléments** de **même type**.
- Avec les tableaux, on peut manipuler des **données composées**, comme les **vecteurs** et les **matrices**.
- Par exemple, on peut utiliser les tableaux pour représenter :
  - Une **liste de notes**;
  - Un **polynôme**;
  - Une **matrice de distances entre des villes**;
  - etc;

# Représentation mémoire d'un tableau

- Un **tableau** est **stocké en mémoire** dans une **zone contigüe** : les éléments du tableau sont stockés en mémoire **l'un après l'autre**, du **premier élément** jusqu'au **dernier élément**.



# Tableau monodimensionnel

- **Définition**
  - Un **tableau monodimensionnel** est un tableau ayant une **seule dimension**.
  - Avec ce type de tableaux, on peut manipuler des **données vectorielles**.
- **Déclaration d'un tableau** :
  - Un tableau se déclare dans la section <**Var**>.
  - **Syntaxe :**

    Nom : **Tableau de** <valeur> <Type>;

# Tableau monodimensionnel

- **Exemples**

  **A : Tableau de 10 Entier;**

  **B : Tableau de 20 Réel;**

  **C : Tableau de 4 Booléen;**

- **Une autre déclaration :**
  - La longueur du tableau peut être une **constante symbolique**.
  - **Avantage** : si l'on veut modifier la longueur du tableau, on ne modifie que la valeur de la constante.

# Tableau monodimensionnel

- **Exemple**

```
Const
   N : Entier = 10;
Var
   A : Tableau de N Réel;
```

# Tableau monodimensionnel

- **Déclaration d'un type tableau**
  - Il est possible de **déclarer un type tableau** en utilisant le mot-clé <**Type**>.
  - Cela permet de simplifier l'écriture d'une déclaration d'un tableau.
- **Exemple**

**Type**

**Tab** : **Tableau de 10 Entier**;

# Tableau monodimensionnel

- **Déclaration d'un type tableau**
  - Dans l'exemple précédent, « **Tab** » désigne le **nom d'un type** (et **non pas un tableau**).
  - Autrement dis, « **Tab** » **n'est pas une variable**.
  - Cependant, on peut déclarer une variable de type tableau de 10 entiers, en utilisant la déclaration suivante :

```
Var
   A : Tab;
```

# Tableau monodimensionnel

- **Initialisation d'un tableau au moment de sa déclaration**

**Var**

**A : Tableau de 3 Entier = {3, -7, 19} ;**

**B : Tableau de 2 Réel = {1.5, -5.75} ;**

**C : Tableau de 3 Caractère = {'x', '0', 'A'};**

**D : Tab = {0, 9, 1, 8, 2, 7, 3, 6, 4, 5};**

# Tableau monodimensionnel

- **Accès à un élément d'un tableau**
  - Soient « **T** » **le nom d'un tableau** de « **N** » éléments, et « **i** » un **entier**.
  - L'élément de « **T** » d'indice « **i** » est représenté par la notation « **T[i]** ».
  - Si « $i \leq 0$ » ou « $i > N$ », l'écriture «**T[i]** » n'a pas de sens (c'est une erreur).
  - Ainsi :
    - « **T[1]** » est le **premier élément** de **T**.
    - « **T[N]** » est le **dernier élément** de **T**.

# Tableau monodimensionnel

- **Accès à un élément d'un tableau**
  - Si « **T** » est un **tableau** dont les éléments sont de type « **X** », alors « **T[i]** » est considéré comme une variable de type « **X** ».
  - Toute **opération valable** avec une variable de type « **X** », est valable avec « **T[i]** ».
  - En particulier, on peut :
    - Écrire la valeur de « **T[i]** »;
    - Lire une valeur pour « **T[i]** »;
    - Affecter une valeur à « **T[i]** »;

# Tableau monodimensionnel

- **Exemple 1**

```
Algorithme  Tableau;
Var
   A : Tableau de 5 Entier = {2, 3, 5, 7, 11};
Début
   Ecrire(A[1]);   Ecrire(A[5]);
   A[3] := A[1] + A[5];
   Ecrire(A[3]);
Fin
```

# Tableau monodimensionnel

- **Boucle de lecture d'un tableau**

```
Pour i := 1 A 5 Faire
    Ecrire("Entrez l'élément d'indice ", i);
    Lire(A[i]);
FinPour
```

- **Boucle d'écriture d'un tableau**

```
Pour i := 1 A 5 Faire
    Ecrire("L'élément d'indice ", i, " : " , A[i]);
FinPour
```

# Tableau monodimensionnel

- **Boucle de parcours gauche->droite d'un tableau**

```
Pour i := 1 A N Faire
    Traiter(A[i]); { traiter élément par élément}
FinPour
```

- **Boucle de parcours droite->gauche d'un tableau**

```
Pour i := N A 1 Pas = -1 Faire
    Traiter(A[i]);
FinPour
```

# Tableau monodimensionnel

- **Affectation globale entre tableaux**
  - Soient « **A** » et « **B** » deux **tableaux de même longueur** N et **même type**.
  - L'instruction « **A := B;** » est une **affectation globale entre tableaux**.
  - Cette instruction se traduit par une **affectation élément par élément** :

```
Pour i := 1 A N Faire
    A[i] := B[i];
FinPour
```

# Tableau à deux dimensions

- **Définition**
  - Un **tableau bidimensionnel** est un tableau ayant **deux dimensions**.
  - Avec ce type de tableaux, on peut manipuler des **données matricielles**.
- **Déclaration d'un tableau bidimensionel** :
  - **Syntaxe :**

    **Nom : Tableau de <val1> × <val2> <Type>;**
  - <**val1**> est le **nombre de lignes** et <**val2**> est le **nombre de colonnes**.

# Tableau à deux dimensions

- **Exemples**

  **A : Tableau de 5 × 5 Entier;**

- **Une autre déclaration :**

**Const**

**N : Entier = 5; { Nombre de lignes }**

**M : Entier = 5; { Nombre de colones }**

**Var**

**A : Tableau de N × M Entier;**

# Tableau à deux dimensions

- **Déclaration d'un type tableau**
  - Il est possible de **déclarer un type tableau à deux dimensions** en utilisant le mot-clé <**Type**>.
- **Exemple**

**Type**

**Matrice : Tableau de 3 × 3 Entier;**

# Tableau à deux dimensions

- **Initialisation d'un tableau à deux dimensions au moment de sa déclaration**

```
Var
    A : Tableau de 2 × 2 Entier = {{3, 7}, {1, 9}} ;
    D : Matrice = {{0, 9, 1}, {8, 2, 7}, {3, 6, 4}};
```

# Tableau à deux dimensions

- **Accès à un élément d'un tableau à 2 dimension**
  - Soient « **T** » **le nom d'un tableau bidimensionnel** de « **N × M** » éléments, et « **i** » et « **j** » deux **entiers** (« **i** » : numéro de ligne, « **j** » : numéro de colonne).
  - L'élément de « **T** » d'indice de ligne « **i** » et d'indice de colonne « **j** » est noté « **T[i, j]** ».

# Tableau à deux dimensions

- **Boucle de lecture d'un tableau bidimensionnel**

```
Pour i := 1 A N Faire
    Pour j := 1 A M Faire
        Lire(A[i, j]);
    FinPour
FinPour
```

# Tableau à deux dimensions

- **Boucle d'écriture d'un tableau bidimensionnel**

```
Pour i := 1 A N Faire
    Pour j := 1 A M Faire
        Ecrire(A[i, j]);
    FinPour
FinPour
```

# Tableau à deux dimensions

- **Affectation globale entre tableaux bidimensionnel**
  - Soient « **A** » et « **B** » deux **tableaux bidimensionnel de même longueur** N × M et **même type**.
  - L'instruction « **A := B;** » est une **affectation globale entre tableaux**.

- **Tableau dynamique**
  - Il arrive fréquemment que l'on ne connaisse pas à l'avance le nombre d'éléments d'un tableau.
  - Une première solution, **moins efficace**, consiste à déclarer un tableau ayant une longueur assez suffisante pour contenir les éléments.
  - Normalement, on impose une **borne supérieure de la longueur du tableau**.

# Tableau dynamique

- **Première solution**

```
Const
   MAX : Entier = 1000;
Type
   Tab : Tableau de MAX Réel;
Var
   A : Tab;
```

# Tableau dynamique

- **Inconvénients**
  - Parfois, il est **difficile de connaître parfaitement une borne supérieure** du nombre d'éléments ➔ **erreur de programmation**.
  - Parfois, **l'espace du tableau est très grand et l'espace réellement utilisé plus petit** ➔ **gaspillage de la mémoire**.

# Tableau dynamique

- **Seconde solution : tableau dynamique**
  - Elle consiste à donner la possibilité de **déclarer un tableau sans préciser au départ sa longueur**.
  - Ensuite, et dans un second temps, au cours de l'exécution de l'algorithme, on fixe la longueur du tableau via une **instruction d'allocation de la mémoire**.
  - Le tableau crée s'appelle un **tableau dynamique**, car il est crée au moment de l'exécution de l'algorithme.

# Tableau dynamique

- **Instruction d'allocation dynamique de la mémoire**
  - L'instruction « **Allouer(<nom>, <nombre>, <type>)** » permet de créer un **tableau dynamique** de nom <**nom**>, ayant un nombre d'éléments égal à <**nombre**> et de type <**type**>.
  - 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**.

# Tableau dynamique

- **Libération de la mémoire allouée dynamiquement**
  - Il ne faut pas oublier de libérer le tableau à la fin de son utilisation avec l'instruction :

    « **Libérer(nom)** »
  - Le paramètre « **nom** » étant le tableau à libérer.
- **Exemple** :
  - On veut faire saisir des notes pour un calcul de moyenne, mais on ne sait pas combien il y aura de notes à saisir.

# Tableau dynamique

- **Solution :**

```
Algorithme Moyenne_Notes;
   Var
        Notes : Tableau de Réel;
        { Tableau dynamique }
        M, S : Réel; { Moyenne et Somme }
        i, N : Entier;
Début
   Ecrire("Nombre de notes à saisir : ");
   Lire(N);
```

# Tableau dynamique

```
{ Allouer N nombres de types réels }
Allouer(Notes, N, Réel);
{ Saisir les notes }
Ecrire("Entrer les ", N, " notes : ");
Pour i := 1 A N Faire
    Lire(Notes[i]);
Finpour
{ Calculer la somme des notes }
S := 0.0 ;
```

# Tableau dynamique

```
Pour i := 1 A N Faire
    S := S + Notes[i];
Finpour
{ Calculer et afficher la moyenne des notes }
M := S / N ;
Ecrire("La moyenne des notes est : ", M);
{ Libérer le tableau Notes }
Libérer(Notes);
Fin
```

# Traitements sur les tableaux

- **Créer** des tableaux
- **Ranger** des valeurs dans un tableau
- **Récupérer**, **consulter** des valeurs rangées dans un tableau
- **Rechercher** si une valeur est dans un tableau
- **Tester** si un tableau vérifie une certaine propriété
- **Mettre à jour** des valeurs dans un tableau
- **Modifier** la façon dont les valeurs sont rangées dans un tableau (par exemple : les trier de différentes manières)

# Traitements sur les tableaux

- Effectuer des **opérations entre tableaux** : comparaison de tableaux, multiplication,...

# Exercices corrigés

- **Exercice 1**
  - Écrire un algorithme qui lit un tableau de 10 entiers, puis calcul la plus grande distance entre ses éléments.
- **Exercice 2**
  - Écrire un algorithme qui lit une matrice carrée d'ordre 5, puis teste si cette matrice est antisymétrique.
- **Exercice 3**
  - Écrire un algorithme qui lit un nombre X (non nul) et qui stocke ses chiffres dans un tableau (de droite à droite). Ensuite, l'algorithme créera l'image de X à partir de ce tableau.

# Corrigé exercice 1

```
Inclure <Math>
Algorithme  Plus_grande_distance;
Const
   N : Entier = 10;
Var
   T : Tableau de N Entier;
   d : Entier; { d = plus grande distance }
   i, j, dij : Entier; { dij : distance entre T[i] et T[j] }
Début
   Ecrire("Entrez les éléments du tableau T : ");
```

# Corrigé exercice 1

```
Pour i := 1 A N Faire
    Lire(T[i]);
Finpour
d := 0;
Pour i := 1 A N - 1 Faire
    Pour j := i + 1 A N Faire
        dij := Val_abs(T[i] - T[j]);
        Si(d < dij) Alors
            d :=  dij;
        FinSi
```

# Corrigé exercice 1

```
        FinPour
    FinPour
    Ecrire("La plus grande distance est : " ,d);
Fin
```

# Corrigé exercice 2

```
Algorithme  Matrice_antisymétrique;
Const
   N : Entier = 5;
Var
   M : Tableau de N × N Réel;
   Antisym : Booléen;
   i, j : Entier;
Début
   Ecrire("Entrez les coefficients de la matrice : ");
```

# Corrigé exercice 2

```
Pour i := 1 A N Faire
    Pour j := 1 A N Faire
        Lire(M[i, j]);
    FinPour
FinPour
Antisym := Vrai;
i := 1;
Tant que (Antisym = Vrai et i ≤ N) Faire
    j := i;
    Tant que (Antisym = Vrai et j ≤ N)
        Si(M[i, j] ≠ -M[j, i]) Alors
```

# Corrigé exercice 2

```
                Antisym := Faux;
            Sinon
                j := j + 1;
            FinSi
        FinTQ
        Si(Antisym = Vrai) Alors
            i := i + 1;
    FinTQ
    Si(Antisym = Vrai) Alors
        Ecrire("Matrice antisymétrique");
```

# Corrigé exercice 2

```
    Sinon
        Ecrire("Matrice non antisymétrique");
    FinSi
Fin
```

# Corrigé exercice 3

```
Algorithme  Image_Nombre;
Var
   X, CX, Y, N, i : Entier;
   T : Tableau de Entier; { Tableau dynamique }
Début
   Ecrire("Entrez un nombre X : ");
   Lire(X);
   CX := X; N := 0;
   Tant que (CX > 0) Faire
        N := N + 1;
```

# Corrigé exercice 3

```
        CX := CX div 10;
    FinTQ
    Allouer(T, N, Entier);
    Y := 0;   CX := X;
    Pour i := 1 A N Faire
        T[i] := CX mod 10;
        X := CX div 10;
        Y := 10 * Y + T[i];
    FinPour
    Ecrire("L'image de ", X, " est : ", Y);
    Libérer(T);
Fin
```