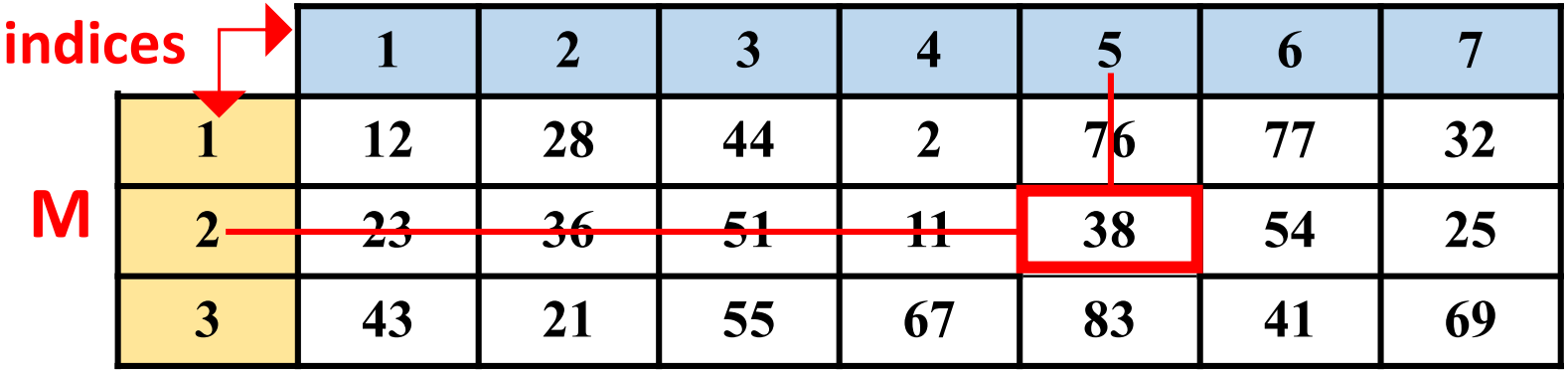
Tableaux à deux dimensions

1. Introduction

- Les **langages de programmation** permettent de déclarer des **tableaux** dans lesquels les valeurs ne sont pas repérées par une **seule indice**, mais par **deux indices**.
- Ces tableaux à **deux dimensions** sont appelés des **Matrices**.

Exemple d'un **tableau à deux dimensions** :



- C'est une **matrice** de valeurs entiers de **3 lignes et 7 colonnes**.
- Les **éléments du tableau** sont repérés par leur **numéro de ligne et leur numéro** de colonne : **M[2][5] = 38**

Tableaux à deux dimensions

2. Déclarations

Pour **déclarer un tableau à deux dimensions**, il faut indiquer:

- ✓Le **nom de la variable** tableau à 2 dimensions : **Nom_Tableau**
- ✓Le **nombre maximum** de Ligne: **Nbligne**
- ✓Le **nombre maximum** de Colonne: **Nbcolonne**
- ✓Le **type** de base des éléments du tableau.

Syntaxe :

Var Nom_Tableau : tableau [1...Nbligne, 1...Nbcolonne] de type_éléments

ou bien

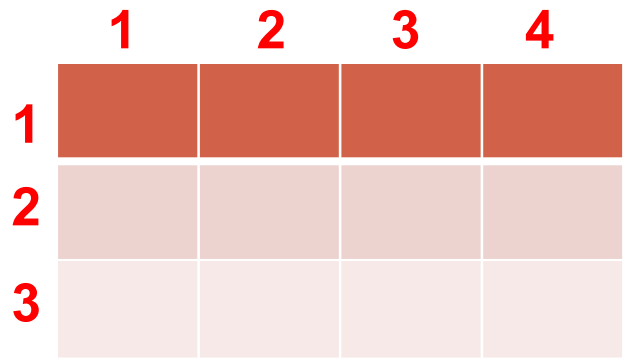
Var Nom_Tableau : tableau [Nbligne, Nbcolonne] de type_éléments

Tableaux à deux dimensions

Exemple de déclaration :

Var M : tableau[1..3, 1..4] d'entiers

M est une **variable tableau** à **deux dimensions** de **3 lignes** et **4 colonnes**



Espace mémoire réservé

Autres exemples de déclarations :

M1 : tableau[1..5, 1..10] de entier;	On déclare une matrice **M1** de **5 lignes et 10 colonnes** dont les éléments sont des entiers
M2 : tableau[15, 3] de réel;	On déclare un tableau **M2** qui stockera 15*3 valeurs réelles
M3 : tableau[30, 4] de caractère;	On déclare un tableau **M3** qui stockera 30*4 valeurs de types caractères

Tableaux à deux dimensions

3. Accès aux composantes d'un tableau à deux dimensions

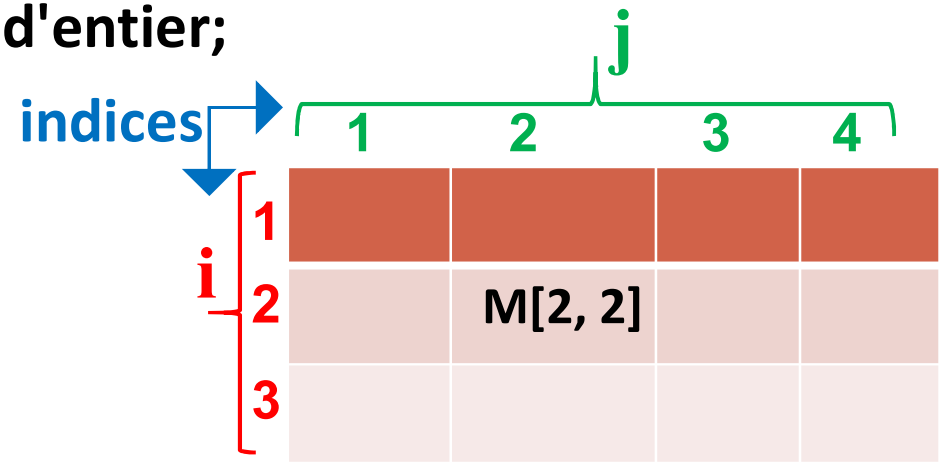
L'**objectif** de la **déclaration** est de r**éserver** un **espace mémoire** nécessaire pour **stocker les éléments** du tableau **ligne par ligne.**

Pour **accéder à un élément** d'un tableau à **deux dimensions**, on écrit:

Nom_tableau[**Numéro_ligne**, **Numéro_Colonne**]

Par exemple :

Var M : Tableau [3, 4] d'entier;



M[**i**][**j**] : permet d'**accéder à l'élément**, de la matrice **M**, qui se trouve à **l'intersection** de la ligne **i** et de la colonne **j**

Tableaux à deux dimensions

Exemple :

Avec la **déclaration** suivante:

Var M: tableau[5, 7] d'entiers;

On définit un **tableau M de 5*7 éléments :**

Tableau M

	1	2	3	4	5	6	7
1							
2		M[2,2]					
3				M[3,4]			
4							
5				M[5,4]			

- L'élément de la 3ième ligne et la 4ième colonne est notée: **M[3,4]**
- L'élément de la 5ième ligne et la 4ième colonne est notée: **M[5,4]**
- L'élément de la 2ième ligne et la 2ième colonne est notée: **M[2,2]**

Tableaux à deux dimensions

Exercice 1 :

Écrire un **algorithme** qui remplit un tableau de **6x13**, avec des **zéros.**

```
Algorithme tableau_Exo1;
  Var
      i, j : entier;
      T : tableau[6, 13] d'entiers;
   Début
       Pour i ← 1 à 6 faire
            Pour j ← 1 à 13 faire
                 T[i, j] ← 0;
            FinPour j
        FinPour i
Fin
```

Le même principe que dans un tableau à une dimension, sauf qu'ici le balayage requiert deux boucles imbriquées, au lieu d'une seule boucle.

Tableaux à deux dimensions

Exercice 2 :

Quel résultat produira cet algorithme ?

Cet algorithme remplit et un tableau de 2x3, de la manière suivante:

X[1, 1] = 1
X[1, 2] = 2
X[1, 3] = 3
X[2, 1] = 4
X[2, 2] = 5
X[2, 3] = 6

1	2	3
4	5	6

```
Algorithme tableau_Exo2;
   Var
      i, j, val : entier;
      X : tableau[2, 3] d'entiers;
    Début
          Val ← 1;
          Pour i ← 1 à 2 faire
              Pour j ← 1 à 3 faire
                    X[i, j] ← Val;
                    Val ← Val + 1;
               FinPour j
          FinPour i
          Pour i ← 1 à 2 faire
                Pour j ← 1 à 3 faire
                      Ecrire (X[i, j]);
                FinPour j
          FinPour i
    Fin
```

Tableaux à deux dimensions

Exercice 3 :

Quel résultat produira cet algorithme ?

Cet algorithme remplit et affiche un tableau 4x2 de la manière suivante:

T[0, 0] = 0
T[0, 1] = 1
T[1, 0] = 1
T[1, 1] = 2
T[2, 0] = 2
T[2, 1] = 3
T[3, 0] = 3
T[3, 1] = 4

0	1
1	2
2	3
3	4

```
Algorithme tableau_Exo3;
   Var
      k, m: entier;
      T : tableau[4, 2] d'entiers;
    Début
          Pour k ← 0 à 3 faire
                Pour m ← 0 à 1 faire
                      T[k, m] ← k+m;
                FinPour m
          FinPour k
          Pour k ← 0 à 3 faire
                Pour m ← 0 à 1 faire
                      Ecrire (T[k, m]);
                FinPour m
          FinPour k
Fin
```

Exercice 4 :

Ecrire un algorithme permettant d'initialiser et d'afficher le tableau à deux dimension suivant:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

```
Algorithme exercice4;
    Var
        T : tableau[2, 8] d'entier;
        val, i, j : entier;
    Début
        // initialisation du tableau de 2*8 valeur
        val← 1;
        Pour i ← 1 à 2 faire
            Pour j ← 1 à 8 faire
                T[i,j] ← val;
                val ← val+1;
            FinPour j
        FinPour i
        Pour i ← 1 à 2 faire
            Pour j ← 1 à 8 faire
                Ecrire(T[i,j]);
            FinPour j
        FinPour i
    Fin
```

Tableaux à deux dimensions

Exercice 5 :

Ecrire un algorithme qui permet de :

- saisir les données d'un tableau T à deux dimensions de 10 lignes et 4 colonnes,
- calculer la somme, le produit et la moyenne des données saisi
- afficher les résultats du calcul sur l'écran.

```
Algorithme tableau_deuxD_Exo5;
        Var
                T : tableau[10, 4] de réels;
                i, j : entier;
                S, prod, M : réel;
        Début
        // Saisie des éléments du tableau
                Pour i ← 1 à 10 faire
                        Pour j ← 1 à 4 faire
                                Ecrire("entrer l'élément T " , i , j);
                                Lire(T[i, j]);
                        FinPour j
                FinPour i
        // affichage des éléments du tableau
                Pour i ← 1 à 10 faire
                        Pour j ← 1 à 4 faire
                                Ecrire(T[i, j]);
                        FinPour j
                FinPour i
```

```
// calculer la somme, le produit et la moyenne et afficher les résultats
S ← 0;
prod ← 1;
Pour i ← 1 à 10 faire
    Pour j ← 1 à 4 faire
        S ← S+T[i,j];
        prod ← prod*T[i,j];
    FinPour j
FinPour i
        M ← S/40;          //40 est le nombre d'éléments du tableau=10*4
        Ecrire("la somme des éléments du tableau est S= ", S);
        Ecrire("le produit des éléments du tableau est prod= ", prod);
        Ecrire("la moyenne des éléments du tableau est M= ", M);
Fin
```

Transposition dune matrice carrée

- **Une matrice** carrée est une matrice à **n lignes** et **n colonnes**.
- L'opération de transposition consiste à inverser les lignes et les colonnes en effectuant une symétrie par rapport à la diagonale de la matrice.

M

1	2	3
4	5	6
7	8	9

M^t

1	4	7
2	5	8
3	6	9

Algorithme Transposition;

Var

i, j, X : entier;

Début

Pour i de 1 à n faire
Pour j de (i+1) à n faire
X ← M[i,j];
M[i,j] ← M[j,i];
M[j,i] ← X;
FinPour j
FinPour i

Pour i ← 1 à n faire
Pour j ← 1 à n faire
Ecrire**(T[i,j]);**
FinPour j
FinPour i

Fin