

# Algorithmique et Structures de Données

Année universitaire  
2020-2021

Dr. Marwa CHAIEB

# Chapitre 5

## Les tableaux

# Plan

3

- Partie 1: Introduction
- Partie 2: Les tableaux monodimensionnels
- Partie 3: Les tableaux à deux dimensions
- Partie 4: De l'algorithmique au langage C

# Plan

4

- Partie 1: Introduction
- Partie 2: Les tableaux monodimensionnels
- Partie 3: Les tableaux à deux dimensions
- Partie 4: De l'algorithmique au langage C

# Introduction

5

## Problème

Ecrire un algorithme qui permet de lire les notes de 100 étudiants et d'afficher les notes des 10 premiers d'entre eux.

# Introduction

6

**ALGORITHME** étudiant

**Var** n1, n2, ....., n100 : réel;

**Début**

    Lire (n1);

    Lire (n2) ;

    .....

    Lire (n100);

    Ecrire ('Voici les notes des dix premiers étudiants') ;

    Ecrire (n1);

    Ecrire (n2);

    .....

    Ecrire (n10);

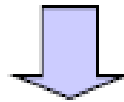
**Fin**

# Introduction

7

Nécessité d'utiliser 100 variables de même type pour saisir les notes de tous les étudiants,

➡ ce qui augmente la taille de l'algorithme



Une nouvelle structure permettant de ranger les notes des étudiants.

# Plan

8

- Partie 1: Introduction
- Partie 2: Les tableaux monodimensionnels
- Partie 3: Les tableaux à deux dimensions
- Partie 4: De l'algorithmique au langage C



# Les tableaux monodimensionnels

9

Lorsque les données sont nombreuses et de même type, afin d'éviter de multiplier le nombre des variables, on les regroupe dans un tableau

## Définition

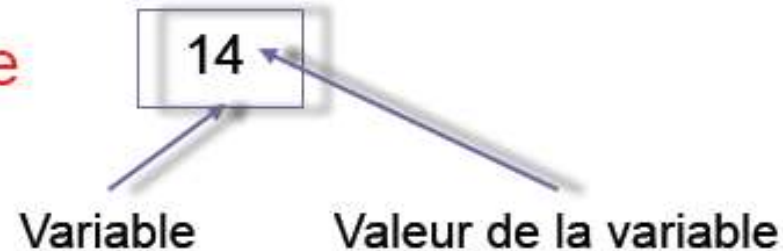
Un tableau mono-dimensionnel ou vecteur est une manière de ranger des éléments ou des valeurs de même type, il regroupe ces éléments dans une structure fixe et permet d'accéder à chaque élément par l'intermédiaire de son rang ou indice.

# Les tableaux monodimensionnels

10

Variable simple

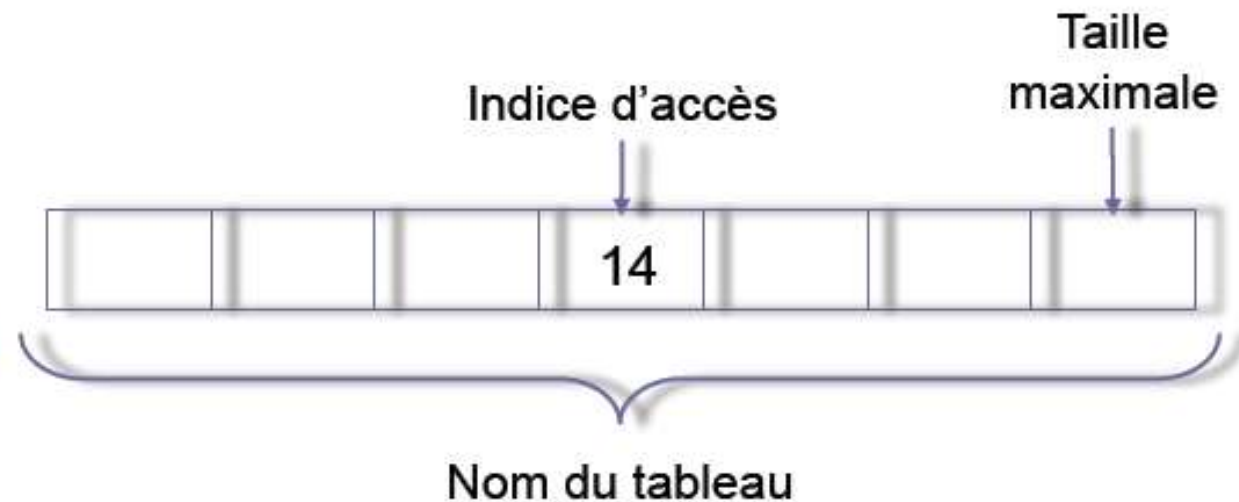
Nom de la variable



Tableau

Nom du tableau

Indice



# Les tableaux monodimensionnels

11

## Déclaration d'un tableau mono-dimensionnel

### Syntaxe

Nom\_Tableau: tableau [1..N] de **Type**

Avec **Type** est le type des éléments du tableau

### Exemple

Tab: Tableau [1..100] de réel

# Les tableaux monodimensionnels

12

- ▶ Exemple :

45	54	1	-56	22	134	49	12	90	-26
----	----	---	-----	----	-----	----	----	----	-----

- ▶ Chacun des dix nombres du tableau est repéré par son rang, appelé indice
- ▶ Déclaration : **Tab : tableau [1..10] de entier**
- ▶ Pour accéder à un élément du tableau, il suffit de préciser entre crochets l'indice de la case contenant cet élément.
- ▶ Exemple :  
Pour accéder au 5ème élément (22), on écrit : **Tab [5]**

# Les tableaux monodimensionnels

13

Les instructions de lecture, écriture et affectation s'appliquent aux tableaux comme aux variables.

$$x \leftarrow \text{Tab}[1]$$

La variable  $x$  prend la valeur du premier élément du tableau, c'est à dire : 45

$$\text{Tab}[6] \leftarrow 43$$

Cette instruction a modifiée le contenu du tableau

# Lecture et affichage

14

```
Procédure Lecture(S: tab: tableau [1..N]: Entier)  
Var i : Entier  
Début  
    Pour i de 1 à N Faire  
        écrire("Entrez la valeur" , i, "du tableau")  
        lire(tab[i]) ;  
    Fin Pour  
FinProcédure
```

```
Procédure Affichage (E: tab: tableau [1..N]: Entier)  
Var i : Entier  
Début  
    Pour i de 1 à N Faire  
        écrire("tab[,i,]=", tab[i]) ;  
    Fin Pour  
FinProcédure
```

# Recherche d'un élément dans un tableau

15

On veut déterminer s'il existe un indice  $i$  allant de 1 à  $N$  tel que  $val = T[i]$

## ■ Tableau non Ordonné

**Algorithme** *Recherche1*

**Var**  $i, val$  : **Entier**,  $T$ : **tableau**  $[1, N]$ : **Entier**,  $Trouve$ : **Booléen**

**Début**

$Trouve \leftarrow \text{Faux}$

**Pour**  $i$  **de** 1 **à**  $N$  **Faire**

**Si** (  $T[i] = val$  ) **alors**

$Trouve \leftarrow \text{Vrai}$

**Fin Si**

**Fin Pour**

**Fin**

# Recherche dichotomique dans un tableau trié

16

## **ALGORITHME RechercheDicho**

### **Var**

T : tableau [1..N] de entier  
elem ,Binf, bsup : entier  
Trouve : booléen

### **Début**

binf ← 1; bsup ← N; Trouve ← faux

### **Répéter**

mil ← (binf + bsup) div 2

si (elem = T[mil]) alors

trouve ← vrai

sinon Si (elem < T[mil]) alors

bsup ← mil - 1

sinon binf ← mil + 1

finsi

finsi

Jusqu'à ((Trouve = vrai) ou (binf > bsup))

### **Fin**



# Insertion d'un élément dans un tableau trié

17

Un tableau  $T$  de dimension  $N+1$  contient  $N$  valeurs entières triées par ordre croissant; La  $(N+1)$ ième valeur est indéfinie.

Insérer une valeur  $VAL$  donnée au clavier dans le tableau  $T$  de manière à obtenir un tableau de  $N+1$  valeurs triées.

# Insertion d'un élément dans un tableau trié

18

Insérer une valeur dans un tableau trié

1. Lecture des éléments du tableau dans un ordre croissant
2. Lecture de la valeur à insérer VAL
3. Déplacer les éléments plus grands que VAL d'une position vers l'arrière.
4. VAL est copiée a la position du dernier élément déplacé
5. Afficher le nouveau tableau

# Insertion d'un élément dans un tableau trié

19

## Algorithme *Insertion*

**Var** *i*, *val* : **Entier**, *T*: **tableau** [1,*N*+1]: **Entier**,

**Début**

**Pour** *i* **de** 1 **à** *N* **Faire**

**écrire**("Entrez la valeur" , *i*, "du tableau ")

**lire**(*tab*[*i*]) ;

**FinPour**

**écrire**("Entrez la valeur à insérer")

**lire**(*val*) ;

*i* ← *N*

**Tantque** (*i*>0) **ET** (*T*[*i*]>*val*) **Faire**

*T*[*i*+1] ← *T*[*i*]

*i* ← *i*-1

**FinTantQue**

*T*[*i*+1] ← *val*

**Fin**

# Plan

20

- Partie 1: Introduction
- Partie 2: Les tableaux monodimensionnels
- Partie 3: Les tableaux à deux dimensions
- Partie 4: De l'algorithmique au langage C

# Tableaux à deux dimensions

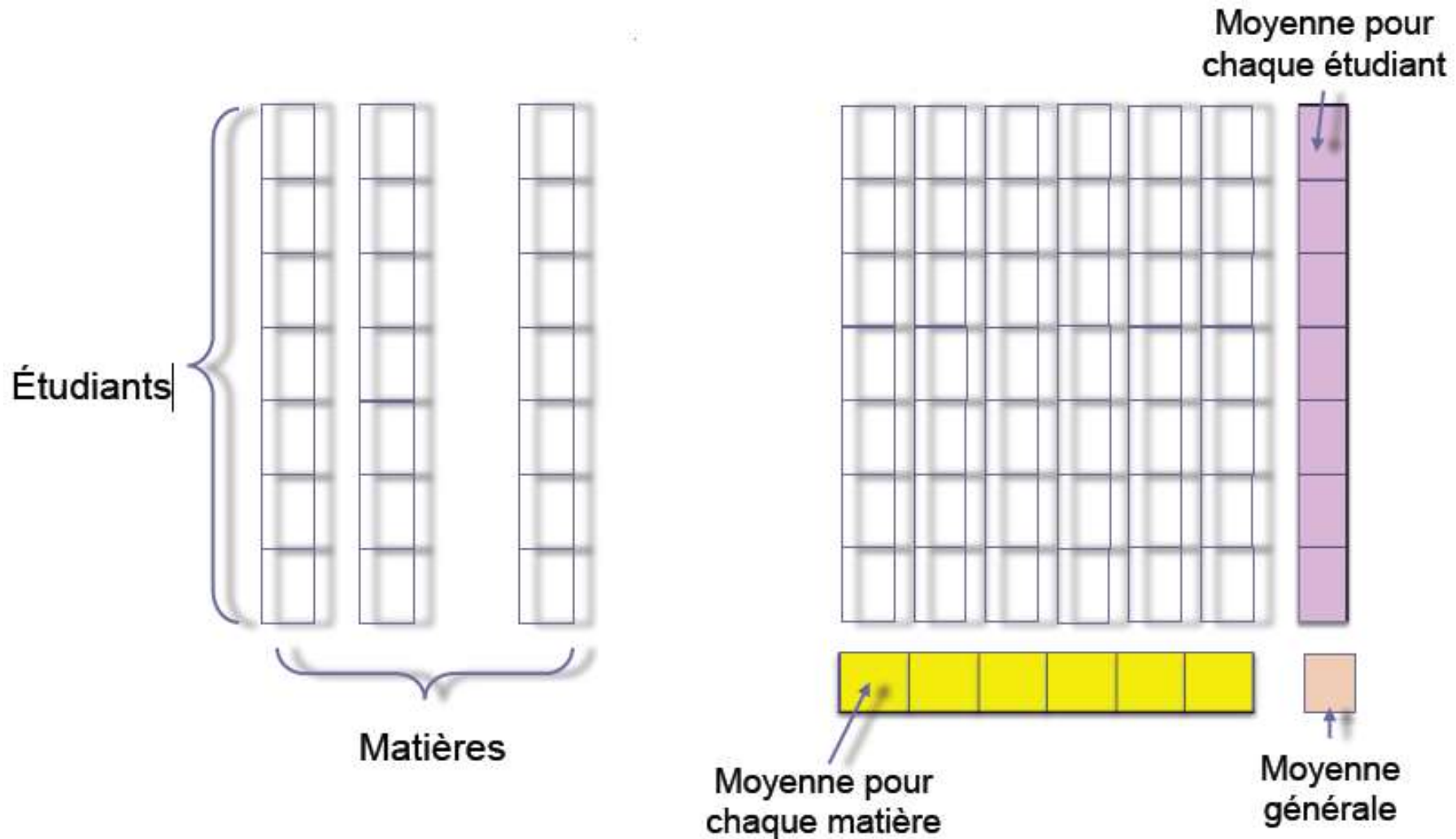
21

## Problème

Comment sauvegarder et manipuler les notes des étudiants d'une classe, dans 6 matières ?

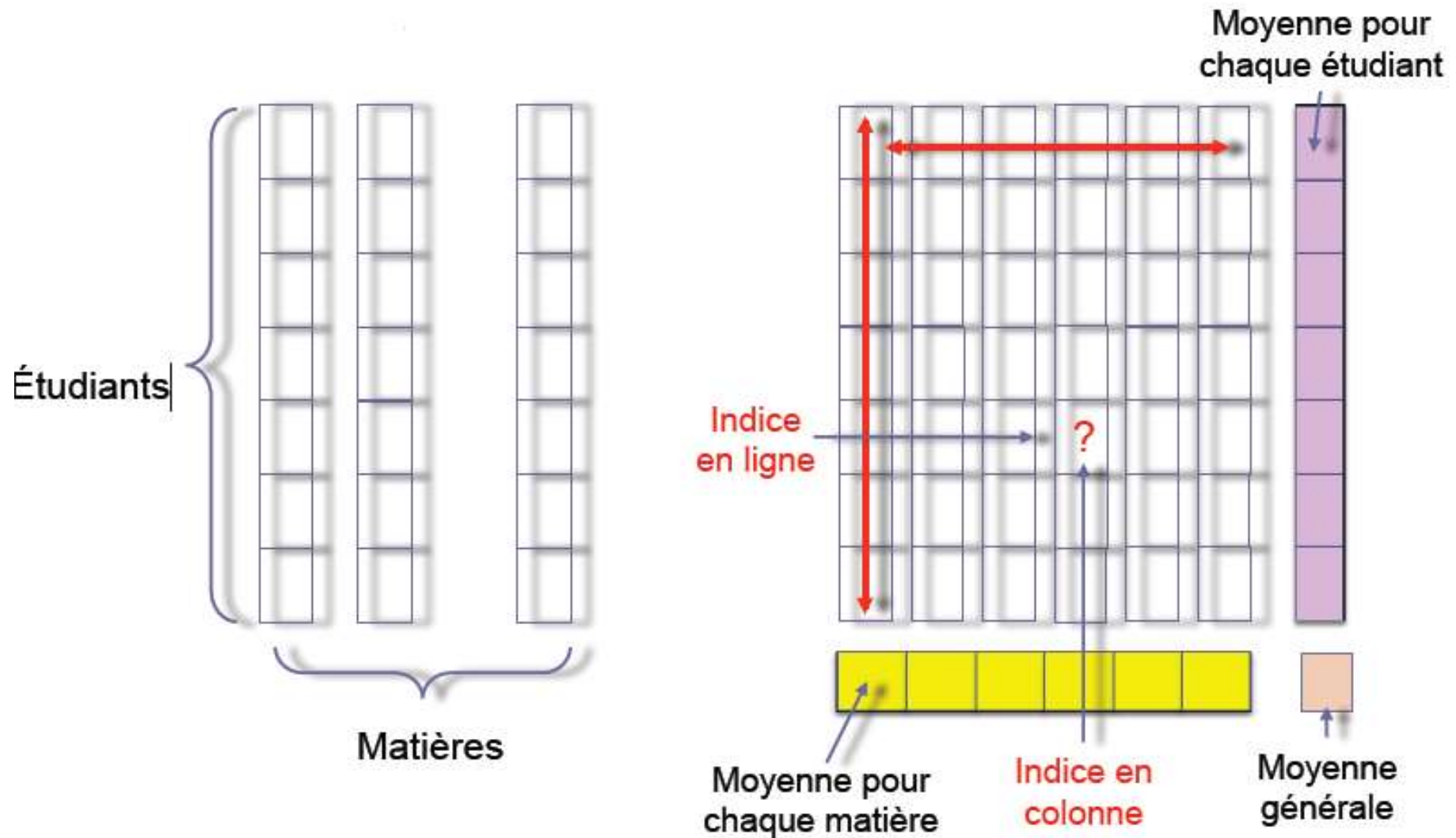
# Tableaux à deux dimensions

22



# Tableaux à deux dimensions

23



# Tableaux à deux dimensions

24

## Déclaration d'un tableau à deux dimensions

```
Var  
  ' Nom_Tableau: tableau [1..N, 1..M] de Type
```

► 19

## Appel :

$X \leftarrow \text{Tab}[2, 3]$

$\text{Tab}[2, 3]$  permet d'accéder à l'élément de la matrice qui se trouve à l'intersection de la ligne 2 et de la colonne 3



# Tableaux à deux dimensions

25

## Algorithme InitTableau

Var

i,j: entier

Tab: tableau [1..100,1..50] de entier

Début

Pour i de 1 à 100 faire

    pour jde 1 à 50 faire

        Tab[i,j] ← 0

    Fin pour

Fin pour

Fin

## Algorithme AfficheTableau

Var

i,j: entier

Tab: tableau [1..100,1..50] de entier

Début

Pour i de 1 à 100 faire

    pour j de 1 à 50 faire

        ecrire(Tab[i,j])

    Fin pour

Fin pour

Fin

# Plan

26

- Partie 1: Introduction
- Partie 2: Les tableaux monodimensionnels
- Partie 3: Les tableaux à deux dimensions
- Partie 4: De l'algorithmique au langage C

# Les tableaux à une dimension

- Un tableau est un ensemble fini d'éléments de même type, stockés en mémoire à des adresses contiguës
- Déclaration `type nom_tableau[nombre_elements]`
- *nombre\_elements*: est une constante entière positive
- Exemple: `float Moy[20] ;`
- L'indice du premier élément est toujours 0.
- L'indice du dernier élément est alors *nombre\_element-1*

# Les tableaux à une dimension

- Accès aux éléments :

- Syntaxe :

*nom\_tableau* [indice]

- Exemples :

- //affecter la moyenne 17 au 2ème élément

`Moy[1] = 17 ;`

- //Saisie de la moyenne du 5ème élément

`scanf("%f", &Moy[4]);`

- //Affichage de la moyenne du 5ème élément

`Printf("%f", Moy[4]);`

# Les tableaux à une dimension

- Le nom du tableau est une adresse constante du premier élément

`Tab=&tab[0]`

- Un tableau ne peut pas figurer à gauche d'un opérateur d'affectation

~~`Tab1=tab2;`~~

- Pour copier un tableau dans un autre:

```
void main() {  
    const int N=10;  
    int tab1[N], tab2[N];  
    int i;  
    ...  
    for (i = 0; i < N; i++)  
        tab1[i] = tab2[i];  
}
```

# Les tableaux à une dimension

- On peut initialiser un tableau lors de sa déclaration par une liste de constantes de la façon suivante:

```
type nom_tableau[] = {constante_1,constante_2,...,constante_N};
```

- Exemple:

```
main()
{
int tab[] = {1, 2, 3, 4};

}
```

# Les tableaux à une dimension

## □ Exercice :

### □ Écrire un programme C qui permet de :

- Lire et d'afficher un tableau T de 10 éléments.
- Calculer la somme des éléments de T.
- Chercher le maximum et le minimum de T.

```
#include<stdio.h>
void main()
{
int T[10];
int i, Som, Max, Min;
```

# Les tableaux à une dimension

```
//Saisie de T
for(i=0;i<10;i++)
{
    printf ("Donner l'élément d'indice %d", i)
    scanf ("%d", &T[i]);
}
```

```
// Affichage de T
for(i=0;i<10;i++)
    printf ("T[%d]=%d", i, T[i]);
```

```
// Somme de T
Som =0;
for(i=0;i<10;i++)
    Som=Som+T[i];
printf("La somme de T = %d", Som);
```



# Les tableaux à une dimension

```
// Max et Min de T
Max = T[0];
Min = T[0];
for(i=1;i<10;i++)
    if(T[i]>Max)
        Max = T[i];
    else if (T[i]<Min)
        Min = T[i];
printf ("Le max de T = %d et min de T = %d", Max, Min);
}
```

# Les tableaux à deux dimension

□ Déclaration :

□ Syntaxe :

```
Type Nom [Dim1][Dim2];
```

Avec

- Type : Type des éléments du tableau.
- Nom : Nom du tableau
- Dim1 : Nombre de lignes
- Dim2 : Nombre de colonnes

□ Exemple :

```
float M[3][4];
```

M	0				3
0					
2					

# Les tableaux à deux dimension

- Accès aux éléments :

- Syntaxe :

```
Nom [Ind_Ligne][ind_Colonne]
```

- Exemples :

- //Affecter 22 à l'élément de la ligne 2 et de la colonne 3

- $M[1][2] = 22$

- //Saisie de l'élément de la ligne 3 et de la colonne 1

- `scanf("%f", &M[2][0]);`

- //Affichage de l'élément de la ligne 3 et de la colonne 1

- `printf("%f", M[2][0]);`

# Les tableaux à deux dimension

## □ Exercice :

Écrire un programme C qui permet de :

- Lire et d'afficher une Matrice  $M$  de 7 lignes et 5 colonnes.
- Calculer la somme des éléments de  $M$ .
- Chercher le maximum et le minimum de  $M$ .

```
#include<stdio.h>
void main()
{
    int M[7][5];
    int i, j, Som, Max, Min;
```

# Les tableaux à deux dimension

```
//Saisie de M
for (i=0;i<7; i++)
    for (j=0;j<5; j++)
    {
        printf ("Donner un élément ");
        scanf("%d", &M[i][j]);
    }

// Affichage de M
for (i=0;i<7; i++)
    for (j=0;j<5; j++)
        printf("%d", M[i][j]);

// Somme de M
Som =0;
for (i=0;i<7; i++)
    for (j=0;j<5; j++)
        Som =Som + M[i][j];
printf ("La somme de M = %d", Som);
```

# Les tableaux à deux dimension

```
// Max et Min de M
Max = M[0][0];
Min = M[0][0];

for (i=0;i<7; i++)
{
    for (j=0;j<5; j++)
    {
        if (M[i][j]>Max)
            Max = M[i][j];
        else if(M[i][j]< Min)
            Min = M[i][j];
    }
}

printf("Le max de M = %d et le min de M = %d", Max, Min);
```

# Allocation dynamique d'un tableau à une dimension

- Pour allouer dynamiquement un tableau à une dimension on utilise la fonction malloc.
- Rappel de la syntaxe de malloc
  - (type \*) malloc (taille à allouer)
- Nous voulons allouer un tableau d'entier de taille 10.

```
int* t = (int*) malloc (10 * sizeof(int));
```

# Allocation dynamique d'un tableau de deux dimensions

- Un tableau de deux dimensions est un tableau de tableaux
- Chaque case du premier tableau contient un **pointeur sur un tableau**.
- `t=(int**) malloc(n*sizeof(int*));`



# Remplissage et Affichage

```
#include<stdio.h>
#include<stdlib.h>
void remplirTab(int* t, int n){
    for(int i=0;i<n;i++){
        scanf("%d",t+i);
    }
}
int** remplirTab2D(int n){
    int i;
    int** t;
    t=(int**) malloc(n*sizeof(int*));
    for (i=0;i<n;i++){
        t[i]=(int*) malloc(n*sizeof(int));
    }
    for(i=0;i<n;i++){
        printf("Donner la ligne %d : ",i+1);
        remplirTab(t[i],n);
    }
    return t;
}
```

```
void showTab(int* t, int n){
    for(int i=0;i<n;i++){
        printf("%d ",*(t+i));
    }
}
void showTab2D(int** t, int n){
    for(int i=0;i<n;i++){
        printf("Ligne %d \n",i+1);
        showTab(t[i],n);
        printf("\n");
    }
}
int main() {
    int i,n;
    int** t= NULL;
    printf("donner la taille de la matrice carrée : ");
    scanf("%d",&n);
    t = remplirTab2D(n);
    showTab2D(t,n);
}
```