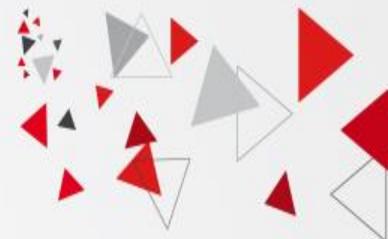


# ► Les tableaux bidimensionnels

-  Exemple introductif
-  Définition
-  Déclaration
-  Tableau et mémoire
-  Manipulation d'un tableau



## Exemple introductif (1/2)

1



```
int NOTE[20] = {10, 14, ... , 13, 16};
```

10

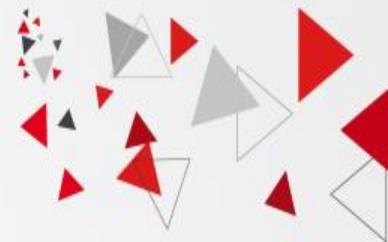


?

```
int NOTE[10][20] = {{10, 14, ... , 13, 16}, {09, 10, ... , 14, 12}, ... ... ...  
{16, 14, ... , 17, 10}};
```



## Exemple introductif (2/2)



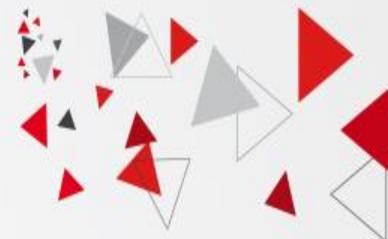
```
int NOTE[10][20];
```

NOTE	0	1	2	...	19
------	---	---	---	-----	----

0	10	14	...	13	16
1	09	10	...	14	12
...	...	...	...	...	...
9	16	14	...	17	10

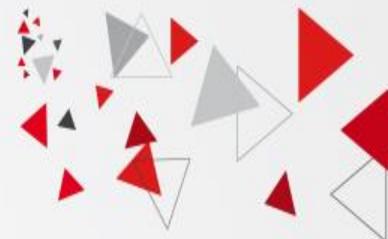
10 lignes

20 colonnes



## Définition

- Un tableau A à deux dimensions est à interpréter comme un tableau (à une dimension) de dimension L dont chaque composante est un tableau (unidimensionnel) de dimension C.
- On appelle L le nombre de lignes, C le nombre de colonnes. L et C sont les dimensions du tableau.



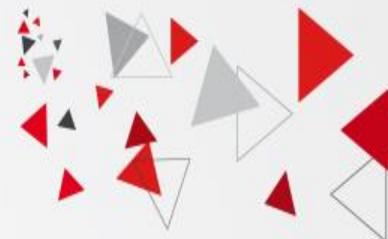
# Déclaration

## Syntaxe:

```
type Nom_du_tableau [ligne][colonne] ;
```

## Exemple :

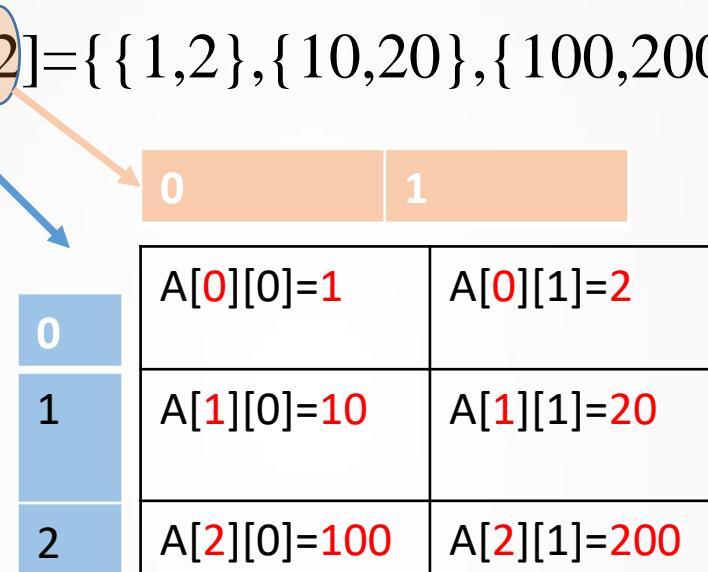
```
int A[10][10]; // tableau de 10×10 entiers de type int  
float B[5][4]; // tableau de 5×4 décimaux de type float  
char C[2][25]; // tableau de 2×25 caractères
```



# Tableau & mémoire

- Représentation graphique:

```
short A [3][2]={ { 1,2 },{ 10,20 },{ 100,200 } };
```

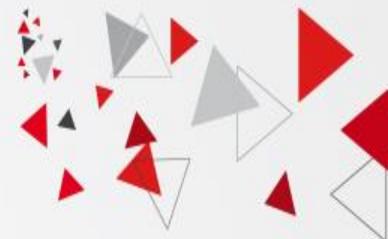


- Représentation en mémoire



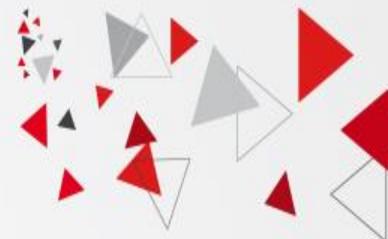
Adresse : 2F06 2F08 2F0A 2F0C 2F0E 2F10 2F12





# Manipulation d'un tableau bidimensionnel

- Initialisation
- Accès
- Remplissage
- Affichage

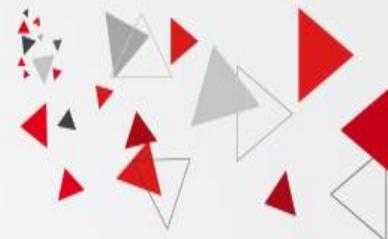


## ► Initialisation (1/2)

- Indiquer la liste des valeurs respectives entre accolades.
- A l'intérieur de la liste, les composantes de chaque ligne du tableau sont encore une fois comprises entre accolades.

### Exemple

```
int A [3][10] ={{ 0,10,20,30,40,50,60,70,80,90},  
{10,11,12,13,14,15,16,17,18,19},  
{ 1,12,23,34,45,56,67,78,89,90}};
```



## ► Initialisation (2/2)

int A [3][3]={ {1,0,0},{1,1,1},{1,0,1}};

1	0	0
1	1	1
1	0	1



int B [3][3]={ {1,1,1} };

1	1	1
!?	!?	!?
!?	!?	!?

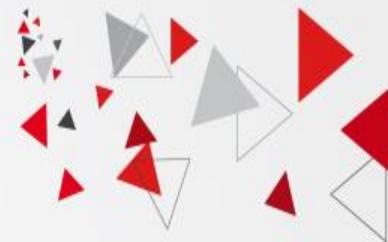


int C [3][3]={ {1,1,1,1} };





## Accès



Les éléments d'un tableau de dimensions L et C se présentent de la façon suivante:

`int A [3][2] = {{1, 2 }, {10, 20 }, {100, 200}};`

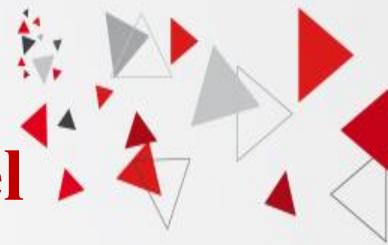
	0	1
0	<b>A[0][0]</b>	<b>A[0][1]</b>
1	<b>A[1][0]</b>	<b>A[1][1]</b>
2	<b>A[2][0]</b>	<b>A[2][1]</b>

Les indices du tableau varient de **0** à **L-1**, respectivement de **0** à **C-1**.

La composante de la  $N^{\text{ième}}$  ligne et  $M^{\text{ième}}$  colonne est notée:  
**A[N-1][M-1]**



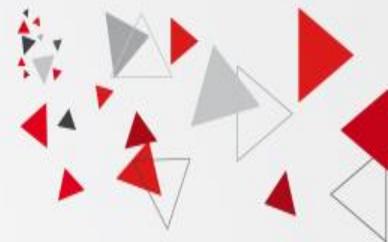
# Remplissage d'un tableau bidimensionnel



```
int main()
{
    int A[5][10];
    int I,J;
    for (I=0; I<5; I++)
    {
        for (J=0; J<10; J++)
        {
            printf( «donner la valeur à stocker »);
            scanf("%d", &A[I][J]);
        }
    }
    return 0;
}
```



## Affichage d'un tableau bidimensionnel



```
int main()
{
    int A[5][10];
    int I,J;
    for (I=0; I<5; I++)
    {
        for (J=0; J<10; J++)
        {
            printf("%7d", A[I][J]);          // décalage lors de l'affichage de contenu
        }
        printf("\n");                    // retour à la ligne après l'affichage de chaque ligne
    }
    return 0;
}
```