



Bases de la programmation

## Séance 9

# Tableaux à deux dimensions

*Sébastien Combéfis*

*mercredi 26 novembre 2014*



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

# Rappels du cours précédent

- Lecture au clavier à l'aide de scanf
- Arithmétique des pointeurs
  - Addition et soustraction (+ et -)
  - Incrémenter et décrémenter (++ et --)
  - Comparaison (==, <, >, <= et >=)
- Mémoire dynamique et tableau
  - Allocation de mémoire : malloc, calloc, realloc
  - Équivalence tableau/bloc de mémoire
  - Pointeur constant

# Pointeur de pointeur

- Un pointeur peut référencer un **bloc de mémoire de pointeurs**
- Un pointeur occupe **8 octets** en mémoire

```
int **data = (int**) malloc (3 * sizeof (int*));
```

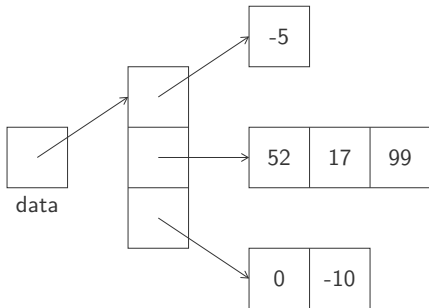
2000	
1992	
1984	
data : 1000	1984

# Tableau de tableaux I

- Une variable `int **data;`
- `data[i]` pointe un bloc mémoire
- `data[i][j]` est un `int`

9004	-10
9000	0
	...
7008	99
7004	17
7000	52
	...
5000	-5
	...
4016	9000
4008	7000
4000	5000
	...
data : 2000	4000

# Tableau de tableaux II



9004	-10
9000	0
	...
7008	99
7004	17
7000	52
	...
5000	-5
	...
4016	9000
4008	7000
4000	5000
	...
data : 2000	4000

# Tableau de tableaux III

```
// Les trois tableaux d'entiers
int *a = (int*) malloc (sizeof (int));
a[0] = -5;

int *b = (int*) malloc (3 * sizeof (int));
b[0] = 52;
b[1] = 17;
b[2] = 99;

int *c = (int*) malloc (2 * sizeof (int));
c[0] = 0;
c[1] = -10;

// Le tableau de tableaux
int **data = (int**) malloc (3 * sizeof (int*));
data[0] = a;
data[1] = b;
data[2] = c;
```

# Libération de mémoire

```
// Les trois tableaux d'entiers
// ...

// Le tableau de tableaux
int **data = (int**) malloc (3 * sizeof (int*));
data[0] = a;
data[1] = b;
data[2] = c;

// Libération de la mémoire
free (data[0]);
free (data[1]);
free (data[2]);
free (data);
```



# Équivalence de notations

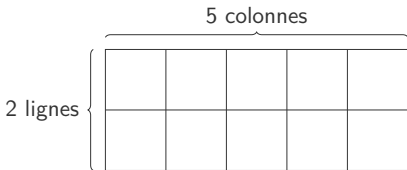
- Soit la déclaration de variable `int **data`,  
les notations suivantes sont équivalentes :

`tab[i][j]`                      `*(* (tab + i) + j)`

`&tab[i][j]`                      `* (tab + i) + j`

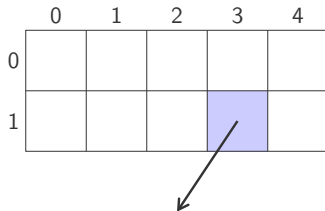
# Tableau à deux dimensions

- Tous les tableaux de deuxième niveau ont la même taille
- Le tableau à deux dimensions `int data[M][N]`
  - $M$  lignes
  - et  $N$  colonnes
- Par exemple, tableau à deux lignes et cinq colonnes



# Double indiçage

	0	1	2	3	4
0					
1					



Élément à la deuxième ligne et quatrième colonne



Élément à la ligne d'indice 1 et la colonne d'indice 3



`data[1][3]`

# Paramètre de type tableau

- Pour un tableau à **une dimension**

*tab est un tableau d'entiers **int***

```
int getMaxTab (int tab [], int N)
{
    // ...
}
```

- Pour un tableau à **deux dimensions**

*mat est un tableau de tableaux de 5 **int***

```
int getMaxMatrix (int mat[][5], int M, int N)
{
    // ...
}
```

# Déclaration de tableau

- Pour un tableau à **une dimension**

```
int tab[5];  
int tab[] = {12, 9, -5, 17, 3};  
int tab[5] = {12, 9, -5, 17, 3};
```

- Pour un tableau à **deux dimensions**

```
int mat[3][2];  
int mat[][2] = {{7, 4}, {9, -9}, {1, -4}};  
int mat[3][2] = {{7, 4}, {9, -9}, {1, -4}};
```

# Maximum d'un tableau à une dimension

- Pour un tableau contenant au moins un élément

```
int getMaxTab (int tab [], int N)
{
    int max = tab[0];

    int i;
    for (i = 1; i < N; i++)
    {
        if (tab[i] > max)
        {
            max = tab[i];
        }
    }

    return max;
}
```

# Maximum d'un tableau à deux dimensions I

- Pour une matrice contenant au moins un élément

```
int getMaxMatrix (int mat[][5] , int M, int N)
{
    int max = mat[0][0];

    int i , j;
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)
        {
            if (mat[i][j] > max)
            {
                max = mat[i][j];
            }
        }
    }

    return max;
}
```

# Maximum d'un tableau à deux dimensions II

- Simplification en se basant sur getMaxTab

```
int getMaxMatrix (int mat[][5], int M, int N)
{
    int max = mat[0][0];

    int i;
    for (i = 0; i < M; i++)
    {
        int maxline = getMaxTab (mat[i], N);
        if (maxline > max)
        {
            max = maxline;
        }
    }

    return max;
}
```



# Équivalence avec les pointeurs

- Pour un tableau à **une dimension**

*tab est un tableau d'entiers **int***

```
int getMaxTab (int *tab , int N);
```

- Pour un tableau à **deux dimensions**

*mat est un tableau de tableaux de 5 **int***

```
int getMaxMatrix (int (*mat)[5] , int M, int N)
```