



Chap. 3 Tableaux

I2011 Langage C : bases

Anthony Legrand
Jérôme Plumet

Les tableaux

- Taille d'un tableau (statique) fixée à la compilation
⇒ taille = expression entière constante



*Une variable constante, définie à l'aide du mot-clé **const**, ne constitue pas une expression constante!*

- C ne conserve pas la taille d'un tableau
⇒ programmer explicitement la gestion de la taille

Définition d'un tableau

- Spécification du **type** des éléments, du **nom** du tableau et de sa **taille**

```
#define N 50 // définition d'une constante N
...
int t1[4]; // définition d'une table de 4 int
           // d'indices compris entre 0 et 3
double t2[N]; // valable si N est connu
              // à la compilation
float t3[2*N-1];
```

- Les éléments d'un tableau sont stockés consécutivement en mémoire (i.e. un seul bloc)

Initialisation d'un tableau

4

En C, il est conseillé de toujours initialiser les tableaux.

► Taille explicite:

```
int tab[3] = {1,123,-15};  
int tab[3] = {1,123,-15,8}; // erreur de compil.  
int tab[4] = {1,6}; // <=> int tab[4] = {1,6,0,0}
```

→ les éléments non explicitement initialisés le sont à 0

► Taille implicite:

```
int tab[] = {1,5,-7}; // <=> int tab[3] = {1,5,-7}
```

Accès aux éléments (1)

► Accès à un élément d'un tableau

→ nom de la table et valeur de l'indice
entre crochets: ' [' et '] '

```
int tab[10] = {0};  
tab[0] = 30;      // assignation de la valeur 30  
                  // au 1er élément du tableau  
tab[2]++;         // incrémentation du 3ème  
                  // élément du tableau
```

Accès aux éléments (2)

6

► indice $\in [0, \text{TAILLE}-1]$



Pas de vérification – ni à la compilation, ni durant l'exécution – que l'indice reste dans les limites de l'index

⇒ risque de dépassement de capacité du tableau (*buffer overflow*)!

→ **Démo : `index_overflow.c`**

Tableaux multidimensionnels (1)

7

- En C, un tableau multidimensionnel est considéré comme un tableau dont les éléments sont eux-mêmes des tableaux

```
int mat[4][3]; // matrice de 4 lignes et  
              // 3 colonnes
```

→ **mat** est un tableau de 4 sous-tableaux, contenant chacun 3 entiers

Tableaux multidimensionnels (2)

8

- L'initialisation d'un tableau multidimensionnel se fait en initialisant chaque sous-tableau

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

```
int mat[3][5] = {{1, 3, 5}, {2, 4}};
```

```
int mat[3][5] = {0}; // initialisation de tous  
                    // les éléments à 0
```

- les éléments non explicitement initialisés le sont à 0

Tableaux multidimensionnels (3)

9

- L'accès à un élément d'un tableau multidimensionnel se fait en indiquant chaque indice entre crochets []

```
mat[2][4] = 5; // assignation de la valeur 5  
               // à l'élément en 3ème ligne  
               // et 5ème colonne
```