

Cours: Algorithmes et Structures des Données

Chapitre 6: Les tableaux

Réalisé par:
Dr. Sakka Rouis Taoufik

1

Chapitre 6 : Les tableaux

I. Déclaration

La déclaration d'un tableau à une dimension réserve un espace de mémoire contiguë dans lequel les éléments du tableau peuvent être rangés.

Syntaxe :

<type simple> <Nom tableau> [<dimension>] ;

Exemples :

```
int T [20] ;  
float T1 [100] ;  
char T2 [30] ;
```

2

Chapitre 6 : Les tableaux

I. Déclaration

La déclaration d'un tableau à deux dimensions (appelé matrice) réserve un espace de mémoire contiguë dans lequel les éléments du tableau peuvent être rangés.

Syntaxe:

<type simple><Nom tableau>[<dimlig>][<dimcol>] ;

Exemples :

```
int   A[10][10] ;
float B[2][10] ;
char  C[10][20] ;
```

3

Chapitre 6 : Les tableaux

I. Déclaration

En C, le nom d'un tableau est le représentant de **l'adresse du premier élément** du tableau. Les adresses des autres composantes sont calculées (automatiquement) relativement à cette adresse.

Exemple : `int T[5] = {100,200,300,400,500} ;`

.....	100	200	300	400	500
Adresse :	1E06	1E08	1E0A	1E0C	1E0E	1E10

→ On peut conclure que `T == &T[0] == 1E06`

→ On peut conclure que **dans ce cas** `&T[i] == &T[0] + i * sizeof(int)`

4

Chapitre 6 : Les tableaux

I. Déclaration

Dans le cas d'un tableau unidimensionnel, si la dimension n'est pas indiquée explicitement lors de l'initialisation, alors l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Exemples:

```
int A [ ] = {10, 20, 30, 40, 50};
```

→ réservation de **5*sizeof(int)** octets

```
float B [ ] = {-1.05, 3.33, 87e-5, -12.3E4};
```

→ réservation de **4*sizeof (float)** octets

```
char C [ ] = {'a', 'b', 'c', 'd', 'e'};
```

→ réservation de **5*sizeof(char)** octets

5

Chapitre 6 : Les tableaux

I. Déclaration

Dans le cas d'une matrice, si les dimensions (nb des lignes ou nb des colonnes) ne sont pas indiquées explicitement lors de l'initialisation, l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Exemples:

```
int B [ ] [ ] = {{1,2,3,4},{10,20,30,40},{100,200,300,400}};
```

→ **Réservation** de **3*4* sizeof (int)** = 24 octets

```
int B[][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} };
```

→ **Réservation** de **3*10*2** = 60 octets

6

Chapitre 6 : Les tableaux

II. Accès aux composantes d'un tableau

Considérons un tableau T de dimension N:

En algorithmique,

- l'accès au premier élément du tableau se fait par **T[1]**
- l'accès au dernier élément du tableau se fait par **T[N]**

En C,

- l'accès au premier élément du tableau se fait par **T[0]**
- l'accès au dernier élément du tableau se fait par **T[N-1]**

Exemple : int T[5] = {100, 200, 300, 400, 500} ;

Nom : Tnote	100	200	300	400	500
Indice :	0	1	2	3	4
Contenu	T[0]	T[1]	T[2]	T[3]	T[4]

7

Chapitre 6 : Les tableaux

II. Chargement et affichage d'un tableau (Solution 1)

```
#include<stdio.h>
void main() {
    int l, N, T[20] ;
    do {
        printf("donner le nombre d'éléments");
        scanf("%d", &N);
    } while (N<=0 || N>20) ;
    /*chargement */
    for (i=0; i<N; i++) {
        printf("T[%d]:",i);  scanf("%d", &T[i]);
    }
    /*affichage*/
    for (i=0; i<N; i++)
        printf("T[%d] = %d\t", i, T[i]);
}
```

8

Chapitre 6 : Les tableaux

II. Chargement et affichage d'un tableau (Solution 2)

/*Cette fonction permet le chargement d'un tableau de n entiers*/

```
void remplirTab (int X [ ], int n) {
    int i ;
    for (i = 0 ; i < n ; i++) {
        printf (" donner l'élément numéro %d ", i);
        scanf("%d", &X[i]);
    }
}
```

Attention : noter bien que le passage des tableaux est toujours par **adresse**

9

Chapitre 6 : Les tableaux

II. Chargement et affichage d'un tableau (Solution 2)

/*Cette fonction permet l'affichage des éléments d'un tableau de n entiers*/

```
void affichageTab (int X[ ], int n) {
    int i ;
    for (i = 0 ; i < n ; i++)
        printf (" Contenu de la case %d = %d \n", i, X[i]) ;
}
```

10

Chapitre 6 : Les tableaux

II. Chargement et affichage d'un tableau (Solution 2)

```
void main ( )
{
    int T [20], N ;
    do {
        printf("donner le nombre d'éléments");
        scanf("%d", &N);
    } while (N<=0 || N>20) ;

    remplirTab (T, N) ;
    afficheTab (T, N) ;
}
```

11

Chapitre 6 : Les tableaux

III. Chargement et affichage d'une matrice

```
/*Cette fonction permet le chargement d'une tableau de n
lignes et m colonnes*/

void remplirMat (int X [ ] [ 50], int n, int m) {
    int i, j ;
    for (i = 0 ; i < n ; i++)
        for (j = 0 ; j < m ; j++) {
            printf (" donner l'élément d'indice %d, %d", i, j );
            scanf("%d", &X[i][j]);
        }
}
```

12

Chapitre 6 : Les tableaux

III. Chargement et affichage d'une matrice

```
/*Cette fonction permet l'affichage des éléments d'une
matrice de n lignes et m colonnes*/
void affichageMat (int X[ ][ 50], int n, int m) {
    int i, j ;
    for (i = 0 ; i < n ; i++) {
        for (j = 0 ; j < m ; j++)
            printf ("%7d", X[i][j]) ;
        /* Retour à la ligne */
        printf("\n");
    }
}
```

Attention : noter bien que le passage des matrices est toujours par **adresse**

13

Chapitre 6 : Les tableaux

IV. Exercices d'application

Exercice 1 : Somme, produit et moyenne des éléments

Écrire un programme C qui lit la taille N ($5 < N \leq 20$) d'un tableau T de type int. Remplit ce tableau par des valeurs entrées au clavier et affiche le tableau.

Calculer et afficher ensuite la somme, le produit et la moyenne des éléments du tableau.

Exercice 2 : Occurrence de 0

Écrire un programme C qui lit la taille N d'un tableau T de type int (taille maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.

Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

14

Chapitre 6 : Les tableaux**IV. Exercices d'application****Exercice 3:**

Écrire un programme C qui lit les dimensions L et C d'un tableau T à deux dimensions (matrice) du type **int** (tailles maximales: 50 lignes et 50 colonnes). Remplir ce tableau par des valeurs entrées au clavier et afficher le tableau ainsi que la somme de tous ses éléments.

Exercice 4:

Écrire un programme C qui transfère un tableau M à deux dimensions L et C (tailles maximales: 10 lignes et 10 colonnes) dans un tableau V à une dimension L*C.

Exemple:

```
a b c d
e f g h ==> a b c d e f g h i j k l
i j k l
```