CH5
Les Tableaux

1ère Année Licence GL

Kais ben Salah
&
Taoufik Sakka Rouis

Les tableaux

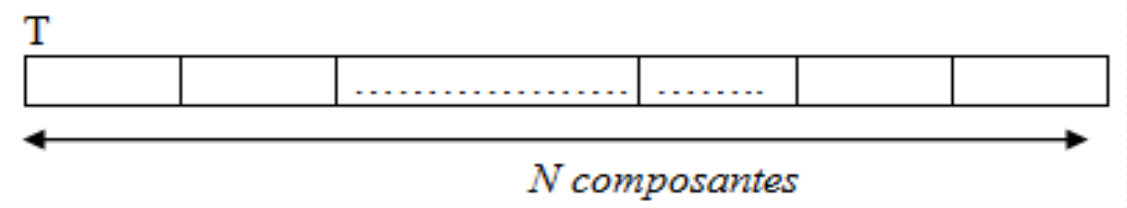
Introduction

Lorsque plusieurs données de même type, généralement destinées au même traitement doivent être accessibles le long d'un programme ,
→ Utilisation de la structure d'un tableau.

1- Les tableaux à une dimension

1.1. Définition

- Un tableau est une structure de données de base qui est un ensemble d'éléments contiguës en mémoire, auquel on a accès à travers un rang (ou indice).
- C permet de manipuler des tableaux. Le nombre d'éléments est alors la dimension du tableau. On dit encore que A est un vecteur de dimension N.



Les tableaux

1.2. Composantes

- **Nom** : identificateur d'un tableau.
- **Type_éléments** : Les éléments d'un tableau sont caractérisés par leur type (entier, réel, caractère,...).
- **Indice** : Tout type dont les éléments possèdent un successeur (les types scalaires), généralement de type entier.

1.3- Déclaration

Syntaxe :

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

Exemples :

1. int A[25] ;
2. float B[100] ;
3. char D[30] ;

Les tableaux

1.4.- Initialisation et réservation automatique

a) Initialisation

Exemples :

1. int A[5] = {10, 20, 30, 40, 50};
2. float B[4]= {-1.2, 33.33, 87e-5, -12.3E4};
3. int C[10] = {0, 1, 1, 0, 0, 0, 0, 1, 0, 1};

Remarque :

Si la liste ne contient pas assez de valeurs pour toutes les composantes, les composantes restantes sont initialisées par des valeurs quelconques.

Les tableaux

b) Réservation automatique

Si la dimension n'est pas indiquée explicitement lors de l'initialisation, alors l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Exemples :

1. Int T[] = {100, 200, 300, 400, 500};
→ réservation de 5*sizeof(int) octets (20 octets)

2. float B[] ={−1.05, 3.33, 87e − 5,−12.3E4};
→ réservation de 4*sizeof(float) octets (16 octets)

3. int T[5] = {100, 200, 300};

Les tableaux

1.5. Mémorisation

En C, le nom d'un tableau est le représentant de l'adresse du premier élément du tableau.

Exemple :

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

T

.....	100	200	300	400	500	.....
Adresse :	1E00	1E04	1E08	1E0C	1E10	

Pour cet exemple **T = 1E00**

Si un tableau est formé de N composantes et si une composante a besoin de M octets en mémoire alors le tableau occupera de N*M octets.

Les tableaux

1.6. Accès aux composantes d'un tableau

Considérons un tableau T de dimension N

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

Les tableaux

1.7. Chargement d'un tableau

```
void Remplir( int T[] , unsigned N)
{
    unsigned i ;
    for (i=0 ; i<N ; i++)
    {
        printf("Donner l'élément numéro %u : ",i) ;
        scanf("%d" , &T[i]) ;
    }
}
```

Les tableaux

1.8. Affichage du contenu d'un tableau

```
void Afficher(int T[],unsigned N)
{
   unsigned i;
   for(i=0 ; i<N ; i++)
        printf("%d\t", T[i]);
}
```

Les tableaux

2. Les tableaux à deux dimensions

2.1. Définition

- Un tableau à deux dimensions M et à interpréter comme un tableau (unidimensionnel) de dimension L dont chaque composante est un tableau (unidimensionnel) de dimension C.
- On appelle L le nombre de lignes du tableau et C le nombre de colonnes du tableau.
- Un tableau à deux dimensions contient L*C composantes.

Les tableaux

2.2. Déclaration et mémorisation

a) Déclaration

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

Exemples :

1. int A[10] [10] ;

2. int B[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}};

b) Mémorisation

- Le nom d'une matrice est le représentant de l'adresse de son premier élément.
- Les composantes d'un tableau à deux dimensions sont stockées ligne par lignes dans la mémoire.

Les tableaux

Exemple :

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

M

.....	1	2	10	20	100	200	.....
Adresse :	1E00	1E04	1E08	1E0C	1E10	1E14	

Pour cet exemple M = 1E00

Un tableau de dimensions L et C, formé de composantes dont chacune a besoin de M octets, occupera L*C*M octets en mémoire.

Exemple :

- En supposant qu'une variable du type double occupe 8 octets (c.-à-d. : sizeof(double)=8), pour le tableau A déclaré par : double **A[10][15] ;**
- **A** réservera **L*C*M** = 10*15*8 = 1200 octets en mémoire.

Les tableaux

2.3. Accès aux composantes d'une matrice

Considérons un tableau A de L lignes et C colonnes.

- Les indices du tableau varient de 0 à L -1, respectivement de 0 à C-1.
- La composante de la **N^{ième}** ligne et **M^{ième}** colonne est notée :A[N-1][M-1]

Syntaxe : **<Nom du tableau> [<ligne>] [<colonne>]**

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

A[0][0]	A[0][1]	A[0][2]	.....	A[0][C-1]
A[1][0]	A[1][1]	A[1][2]	.....	A[1][C-1]
A[2][0]	A[2][1]	A[2][2]	.....	A[2][C-1]
.....	.....	.....	.....	.....
A[L-1][0]	A[L-1][1]	A[L-1][2]	.....	A[L-1][C-1]

Les tableaux

2.4. Chargement d'une matrice

```
void remplir(int M [ ][ ], unsigned L , unsigned C)
{
    unsigned i , j;
    for (i=0 ; i<L ; i++)
        for (j=0 ; j<C ; j++)
        {
            printf(" donner [%u][%u] : ", i , j) ;
            scanf("%d", &M[i][j]) ;
        }
}
```

Les tableaux

2.5. Affichage du contenu d'une matrice

```
void Affiche(int M[ ][ ], unsigned L , unsigned C)
{
    unsigned i , j;
    for (i=0 ; i<L ; i++)
    {
        for (j=0 ; j<C ; j++)
            printf ("%7d \t" , M[i][j]);
        printf(" \n");
    }
}
```