Chapitre 7 : Les tableaux

7.1 Définition

Les tableaux sont des structures de données qui permettent de mémoriser et gérer un ensemble de données du même type. Lorsque que l'on défini un tableau, on réserve en mémoire une zone constituée de cases contiguës de même taille. Le nombre de case est fixe et correspond à la longueur déclarée lors de la définition du tableau. Comme toutes variables, les tableaux doivent être utilisé avant utilisation

7.2 Tableau unidimensionnel

7.2.1 Définition

Pour définir un tableau à une dimension, on utilise la syntaxe suivante :

```
type nomTableau[N];
```

Le type est évidement le type de donnée qui sera stocké dans le tableau (int, char, etc.)

Le nomTableau est le nom de la variable

Le N est la longueur du tableau. En C, elle ne peut être une variable. Il s'agit donc d'une constante déclarée directement (un entier) soit définie en début de fichier

Exemple :

```
int tab[3];
```

On a déclaré ici un tableau qui contiendra 3 entiers. En mémoire, il sera présenté comme ceci :

Numéro de case	N°1	N°2	N°3
Contenu de la case	?	?	?
Indice d'accès à la case	0	1	2

Pour accéder à la case 1, on utilisera l'indice 0. Par conséquent, le dernier élément d'un tableau se trouve toujours à l'indice (n° de case – 1).

Lors de la définition d'un tableau, le contenu des cases n'est pas initialisé.

7.2.2 Initialisation à la déclaration

```
int tab[3] = {5, 2, 17};
```

7.2.3 Initialisation par affectation

```
int tab[3];
int tab[0] = 4;
int tab[1] = 5;
int tab[2] = tab[1]*tab[0];
```

À la suite de ce code, le tableau tab à toutes ses valeurs initialisées

7.2.4 Initialisation par lecture

```
int i;
int tab[10];
for(i=0; i<10; i++)
   scanf("%d", &tab[i]); // alternative: scanf("%d", *(tab+i));
```

7.2.5 Débordement par excès ou par défaut.

Lorsque l'on essaye d'accéder à une valeur en dehors du tableaux en C, il se peut que l'on obtienne une valeur. Il s'agit du contenu de case mémoire hors du tableau, et par conséquent, le contenu obtenu n'est jamais exploitable. De plus, si l'on modifie ce contenu, on risque d'écraser des données importantes du système !

Remarque :

Le débordement par défaut correspond à ce qui se trouve avant le début du tableau.

7.3 Les tableaux à deux dimensions

7.3.1 Déclaration d'un tableau à deux dimensions

```
int tab[2][3];
```

Le tableau tab est un tableau de 2 lignes et 3 colonnes.

Pour accéder au contenu, il suffit d'indiquer les indices comme ceci :

```
tab[0][1]; //accède au deuxième élément de la première ligne
```

7.3.2 Initialisation à la déclaration.

Pour initialiser un tableau au moment de sa déclaration, la syntaxe est la suivante :

```
float truc[2][3] = {{5.1, 17},{12, 1.4, 7}};
```

On constate dans cette exemple, que la 3ème valeur de la première ligne n'est pas initialisée. Ça n'est pas un soucis, mais attention si l'on tente d'exploiter cette donnée.

7.3.3 Initialisation par affectation.

```
Tab[0][1] = 5;
```

7.3.4 Initialisation par lecture

```
int i, j;
float machin[2][3];
printf("Entrer six valeurs");
for(i=0; i<2; i++) {
   for(j=0; j<3; j++) {
      scanf("%f", &(machin[i][j]));
   }
}
```