Université Paul Sabatier
L2 - S4 - Printemps 2005

Programmation impérative et langage C

feuille de **TD n° 2 :** Utilisation de tableaux.

---

1 Rappels

Un **tableau** est un groupement ordonné de données de même type appelées composantes :
Ce type peut-être :

.entier : int
.réel : float (simple précision) double (double précision)
.caractère : char

Les tableaux sont **contraints** : le nombre d'éléments d'un tableau est fixé à la compilation. Il faut donc dimensionner le tableau en veillant à ce que sa taille soit suffisante dans tous les cas d'éxécution (quitte à ne pas utiliser toutes les cases du tableau ce qui a pour conséquence une occupation quelquefois inefficace de la mémoire. [1])
On pourra éventuellement utiliser une constante pour dimensionner le tableau.

- Vecteurs :

$$identificateur\_de\_type \;\; identificateur\_de\_variable\; [taille]$$

- Tableaux à plusieurs dimensions :

$$identificateur\_de\_type \;\; identificateur\_de\_variable\; [taille\_1]\; [taille\_2]\; \cdots [taille\_n]$$

Exemples :

```
# define Nbre_Matieres 10
void main()
{
int Mois [6] ;
float Notes [6] [Nbre_Matieres] ;
```

[1] On verra dans un chapitre ultérieur la possibilité de créer une variable lors de l'éxécution , on parle alors de variable **dynamique**(par opposition aux variables **statiques**). On pourra ainsi dimensionner un tableau lors de l'éxécution ceci afin de mieux gérer la mémoire.

float $Moyennes\ [6] = \{10.2, 8.7, 12.6, 14.2, 15.0, 9\}$; * initialisation simultanée *\
⋮
}

L'accés à une case est direct par le nom du tableau et l'indice de sa case : $Mois[0]$, $Mois[1]$, $Notes[0][3]$ $\cdots$
Chaque case du tableau se comporte comme une variable du type considéré.

On peut créer un nouveau type grâce à la syntaxe :

typedef $Identificateur\_de\_type$ $Identificateur$ $[taille\_1]\ [taille\_2]\ \cdots [taille\_n]$

Exemples :

typedef int $Jours[12][31]$;
typedef char $Lettres\_min[26]$

Les cases d'un tableau occupent des emplacements mémoire contigus.

2 Exercices

◇ Exercice **1** :

1. Ecrire un programme qui, pour un ensemble de 100 naturels tapés au clavier, calcule le nombre d'éléments supérieurs à la moyenne de ces entiers.
2. Modifier le programme pourqu'il puisse traiter N entiers naturels, N étant un entier inférieur à 1000 lu au clavier.

◇ Exercice **2** : Ecrire un programme qui inverse l'ordre des éléments dans un tableau de 50 entiers lus au clavier. (On choisira une solution qui n'utilise qu'un seul tableau et qui ne le traverse qu'une fois (aprés la saisie)).

◇ Exercice **3** : N est un entier strictement positif inférieur ou égal à $max = 25$.
On considère un tableau carré comprenant N lignes et N colonnes et composé de N^2 nombres entiers strictement positifs. Ecrire un programme qui crée un tel tableau à partir de valeurs tapées au clavier, calcule les sommes des nombres de chaque ligne, chaque colonne et des deux diagonales et affiche la somme minimale.

Comment obliger l'utilisateur à taper un nombre N inférieur à max ?

◇ Exercice **4** :
On considère le programme suivant :

```
# include <stdio.h>
# include <conio.h>

typedef int Vec[3] ;

void main()
{
int i ;
Vec A,B ;

clrscr ;
for(i=0 ;i<=3 ;i=i+1) {A[i]=i+1 ;}
printf("\nA=") ;
for(i=0 ;i<=3 ;i=i+1){ printf(" %d ",A[i]) ; }
for(i=0 ;i<=3 ;i=i+1) {B[i]=i+1 ;}
printf("\nA=") ;
for(i=0 ;i<=3 ;i=i+1) { printf(" %d ",A[i]) ; }
}
```

Ce programme compile bien et à l'éxécution, on obtient :

A=1 2 3 3
A=4 2 3 3

Ce résultat n'est pas cohérent avec les instructions éxécutées, comment cela s'explique-t-il ?