Chapitre 3 : Les tableaux

Notion de tableau

Soit un entier n positif.

Supposons qu'on veuille saisir n valeurs réelles afin de calculer leur moyenne, ou de trouver leur minimum, ou de les afficher par ordre croissant.

Pour des valeurs petites de n, on peut déclarer n variables réelles pour résoudre le problème.

Mais si n est assez grand, on se rend compte que cela devient impropre, fastidieux, voire impossible.

Il faudrait, dans ce cas, utiliser une variable permettant de représenter les n valeurs. Le type de données de cette variable serait le type tableau.

Notion de tableau

Exemple

Saisir la liste des 12 notes sur 30

16 23 8 19 28 20 18 14 10 9 15 24

Voici la liste de ces notes sur 20

10.67 15.33 5.33 12.67 18.67 13.33 12 9.33 6.67 6 10 16

- Problème :
  - Déclaration de 12 variables différentes.
  - Tâche fastidieuse avec 30 ou 40 notes
  - Variables possédant des noms différents
    - Impossible d'utiliser une boucle.
    - Code devient très répétitif et redondant
- Solution :
  - Utiliser une SD avec un nom commun pour toutes les variables et de les repérer par un numéro.
  - Les tableaux

Notion de tableau

Définition :

Un tableau est une collection séquentielle d'éléments de même type, où chaque élément peut être identifié par sa position dans la collection. Cette position est appelée indice et doit être de type scalaire.

Notion de tableau

Déclaration :

Pour déclarer un tableau, il faut donner :

- son nom (identificateur de la variable)
- sa taille : le nombre d'élément du tableau.
- le type des éléments le composant.

Syntaxe :

variable Tab : tableau [taille] d'entiers
ou
Variable nom : tableau[<taille >] de <type des composants>

Exemple :

variable tab : tableau[10] de
réels

Notion de tableau

En C, la déclaration se met sous la forme :

< type > < identificateur >[<taille >]

<type> <nomdutableau>[<taille >] ;

Exemple :

```
int tab [10];  // déclaration d'un tableau de 10 entiers

float tab [10];  // déclaration d'un tableau de 10 réels
```

Notion de tableau

Schématiquement, on va représenter la variable tab comme suit:

0	1	2	3	4	5	6	7	8	9
8.4	3.5	12	20	10	13.34	50	100	30.1	60.9

Dans la mémoire centrale, les éléments d'un tableau sont stockés de façon linéaire, dans des zones contiguës.

Le tableau ci-dessus est de dimension 1, nous verrons un peu plus loin que l'on peut représenter des tableaux à 2 dimensions, voire même plus. L'élément n° 1 sera représenté par l'expression tab [1].

Dans notre exemple, tab[1] peut être traité comme une variable réelle. On dit que le tableau t est de taille 10.

Les éléments d'un tableau à n éléments sont indicés de 0 à n -1.

Création d'un tableau

La création d'un tableau consiste au remplissage des cases le composant. Cela peut se faire par saisie, ou par affectation.

Par exemple, pour remplir le tableau t précédent, on peut faire :

t[1] = 8.4 ; **t[2] = 3.5** ; … **tab[10] = 60.9 ;**

Si on devait saisir les valeurs, il faudrait écrire :

```
Pour i ← 0 à 9  faire
      lire(t[i] );
```

```
for( i = 0 ; i<10 ; i++)
      scanf("%d" , &t[i] );
```

Affichage d’un tableau

Afficher un tableau revient à afficher les différents éléments qui le composent. Pour cela, on le parcourt (généralement à l’aide d’une boucle avec compteur) et on affiche les éléments un à un.

Exemple :

```
for( i = 0 ; i<10 ; i++)
      printf("%d" , t[i] );
```

Exercice d’application

Ecrire un programme qui permet de créer un tableau d’entiers t1 de taille 20 par saisie, et un tableau t2 de même taille en mettant dans t2[i] le double de t1[i], i appartenant à {0, .., 19 }.

Exercice d'application

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

main()
          //{ création de t1 }
 {
     int Tab[10],Tab2[10],i;
           for(i=0;i<10;i++)
           {
                printf("donnée Tab[%d] :\n",i);
                scanf("%d",&Tab[i]);
           }
                // création de t2
        for(i=0;i<10;i++)
            {
                Tab2[i]=2*Tab[i];
                printf("Tab2[%d] = %d\n",i,Tab2[i]);
             }
          }
```

Traitement d’un tableau

Après avoir créé un tableau, on peut y effectuer plusieurs opérations comme le calcul de la somme ou de la moyenne des éléments, la recherche du plus petit ou du plus grand élément du tableau, le test d’appartenance d’un objet au tableau, …

Pour la suite, on considère un tableau d’entiers t déclaré comme suit :

Int tab[n]

Somme des éléments d'un tableau

On effectue la somme des éléments du tableau tab, le résultat est dans la variable S :

```
S=0 ;
for(i=0;i<10;i++)
    S = S + tab[i];
    printf("la somme des elements de tab est %d: ", S);
```

Minimum d’un tableau

On cherche le plus petit élément du tableau tab, le résultat est dans la variable min :

```
min = tab[0] ;

for(i=0;i<10;i++)
     if (tab[i] < min)
        min = tab[i];
        printf(« Le minimum des éléments de
tab est: \n», min) ;
```

Test d’appartenance

On cherche si l’entier x appartient à tab, le résultat est mis dans la variable booléenne appartient :

```
appartient = 0;

for (i=0; i < n; i++)
   if (tab[i]==x)
   appartient = 1;
   if (appartient==1)
      Printf("x appartient à tab")
  else
      printf("x n’’appartient pas à tab");
```

Test d'appartenance

On remarque que l'on peut arrêter les itérations (la recherche) si l'on rencontre l'élément x dans tab. Pour cela, il faut utiliser une boucle *while* ou *do while*:

```
i = 0;
while ((tab[i]!=x) && (i<n))
    i=i+1;
    if (i==n)
        printf("x n''appartient pas à tab");
    else
        printf("x appartient à tab");
```

```
i = 0 ;
do
    i=i+1;
while (tab[i]!=x) && (i<n));
    if (i>n)
        printf("x n''appartient pas à tab");
    else
        printf("x appartient à tab");
```

Dans les deux cas, si x appartient à tab, la valeur de i est l'indice de la case qui le contient.

Ajout d'un élément

On suppose que le tableau tab est « rempli » et qu'il reste une case non occupée à la fin (la $n^{ième}$ case).

Si on veut alors ajouter un entier x à la fin du tableau, il suffira d'écrire

tab[n] = x ;

Mais, si on veut ajouter x dans une case dont l'indice k est différent de n, il faudra décaler les éléments tab[k], tab[k+1], …, tab[n-1] vers la droite pour libérer la case d'indice k :

```
for( i = n ;i>=k; i--)
    tab[i] = tab[i-1];
tab[k] = x;
```

Suppression d’un élément

Pour supprimer l’élément se trouvant à la position k de tab, on l’écrase en faisant décaler les éléments placés après lui vers la gauche :

```
for (i = k; i< n; i++)
   tab[i] = tab[i+1];
```

Il faut noter qu’après une telle opération, l’élément se trouvant à la dernière position n’est plus significatif.

Tableaux de caractères

Une chaîne de caractères est soit une chaîne vide, soit un caractère suivi d'une chaîne de caractères ; en un mot c'est une collection de caractères.

Exemples :

"Bonjour", "L'ESTM se situe à Castor", "A", "2017 "

Les chaînes de caractères sont stockées sous forme de tableaux de caractères. Le compilateur complète la chaîne avec un caractère NULL ('\0') qui est ajouté à la suite du dernier caractère utile constituant la chaîne de caractères. Il faut donc que le tableau ait au moins un élément de plus que le nombre de caractères de la chaîne littérale.

- Par exemple, la phrase "Toto" sera codée de la sorte :

'T'	'o'	't'	'o'	0	…

Tableaux de caractères

Déclaration d'une chaîne de caractères

Une chaîne de caractères peut donc être déclarée de la manière suivante :

```
char ch[10]; // chaîne de 9 caractères + caractère NULL
```

Exemple :

```
char c [ 5 0 ] = "Toto" ;
```

Cette instruction déclare une chaine de caractères c initialisée à "Toto". Les 5 premiers éléments du tableau seront occupés par les 4 caractères de la chaîne ainsi que par le caractère null, les autres contiendront des valeurs non significatives.

Tableaux de caractères

Déclaration d'une chaîne de caractères

Observez bien l'exemple suivant :

```
char c [ 4 ] = "Toto" ;
```

Cette déclaration engendrera un warning à la compilation et probablement une erreur à l'exécution car l'affectation du caractère nuln à la 5-eme position du tableau donnera lieu à un débordement d'indice.

Tableaux de caractères

Accès aux éléments

Le code suivant permet d'afficher une chaîne caractère par caractère :

```
int  i = 0 ;
while(c[i]!=0)
     printf("%c" , c[i++]);
```

Notez que le corps de la boucle while est itéré jusqu'à ce que le caractère null soit rencontré. Il est donc impératif que votre chaîne se termine par le caractère nul et que le caractère nul se trouve dans la plage d'indices du tableau .

Tableaux de caractères

Initialisation à la saisie au clavier

On peut utiliser la fonction scanf et le format %s mais on utilisera de préférence la fonction gets non formatée.

```
char texte[10];
      printf("Entrer un texte : ");
      scanf("%s", texte);
      est équivalent à gets(texte);
```

Une chaîne étant un pointeur, on n'écrit pas le symbole &.

Remarque: scanf ne permet pas la saisie d'une chaîne comportant des espaces. Les caractères saisis à partir de l'espace ne sont pas pris en compte (l'espace est un délimiteur au même titre que (line feed) LF('\n')) mais ils sont rangés dans le buffer d'entrée. Pour saisir une chaîne avec des espaces, il faut utiliser l'instruction gets. A l'issue de la saisie d'une chaîne de caractères, le caractère de retour chariot est remplacé par le caractère de fin de chaîne '\0'.

La bibliothèque string.h

Cette bibliothèque propose des fonctions de maniement de chaînes de caractères, à savoir :

- strcmp : comparer deux chaînes.
- strlen : longueur d'une chaîne de caractère
- strsubs : rechercher une sous-chaîne
- strcat : concaténer deux chaînes
- strcpy : copier une chaîne

Il vous est conseillé d'examiner et de comprendre comment fonctionnent ces fonctions

La bibliothèque string.h

strlen(<s>)	fournit la longueur de la chaîne ***sans*** compter le '\0' final	
strcpy(<s>, <t>)	copie <t> vers <s>	
strcat(<s>, <t>)	ajoute <t> à la fin de <s>	
strcmp(<s>, <t>)	compare <s> et <t> lexicographiquement et fournit un résultat:	
	négatif	si <s> précède <t>
	zéro	si <s> est égal à <t>
	positif	si <s> suit <t>
strncpy(<s>, <t>, <n>)	copie au plus <n> caractères de <t> vers <s>	
strncat(<s>, <t>, <n>)	ajoute au plus <n> caractères de <t> à la fin de <s>	

Les tableaux à deux dimensions

Pour traiter les notes obtenues par un étudiant à 10 épreuves on peut utiliser un tableau de 10 réels. Pour traiter les notes obtenues par 5 étudiants, on pourrait utiliser 5 tableaux de 10 réels chacun.

Mais puisqu'on va effectuer très probablement les mêmes traitements sur ces tableaux, il est préférable de les regrouper dans une seule variable qui sera un tableau de 5 lignes et 10 colonnes.

Chaque élément de ce tableau multidimensionnel sera identifié par deux indices (i,j) : la position i indiquant la ligne et la position j indiquant la colonne.

Les tableaux à deux dimensions

Déclaration :

La déclaration se met sous la forme :
< type > < identificateur > < taille1 > < taille2 >

On accède à un élément du tableau en donnant un indice par dimension (entre crochets).

Exemple :

```
#define LIGNE  2 // matrice de 2 lignes et 3 colonnes
#define COLONNE  3

int i, j; // i est l'indice de ligne et j est l'indice de colonne
int tab[LIGNE] [COLONNE]; // déclaration d'un tableau de 6 entiers
```

Les tableaux à deux dimensions

Exemple :

// affichage des 6 éléments du tableau

```
for (i = 0; i < ligne; i++)
        for (j = 0; j < colonne; j++)
          printf("%d\n", tab[i][j]);
```

Les tableaux à deux dimensions

Exercice d’application

Ecrire un programme qui permet de créer (par saisie) et d’afficher un tableau tab à deux dimensions d’entiers de taille 3x5.

Les tableaux à deux dimensions

```
#include <stdio.h>

 main()  // creation de t1
    {
      int Tab[3][5],i,j;

      //saisie du tableau
      for(i=0;i<3;i++)
            for(j=0;j<5;j++)
                {
                     printf("donner Tab[%d][%d]",i,j);
                     scanf("%d",&Tab[i][j]);
                 }
           //(affichage de t )
           for(i=0;i<3;i++)
                  {
                     for(j=0;j<5;j++)
                             printf("%d",Tab[i][j]);
                             printf("\n");
                   }
 }
```

Chapitre 4: LES POINTEURS