Chapitre 5

1. Introduction

Jusqu'à présent, nous avons uniquement utilisé des variables simples et élémentaires (de types: entier, réel, caractère,…) qui correspondent à un unique emplacement en mémoire d'un type particulier. Ces variables ne peuvent contenir à un instant donné qu'une seule valeur, et l'affectation d'une nouvelle valeur détruit l'ancienne. Cependant, dans les problèmes réels on doit manipuler un grand nombre de données du même type auxquelles on applique les mêmes traitements.

Par exemple, on peut considérer les notes d'une promotion de 200 étudiants. La déclaration de 200 variables est un peu difficile.

Alors, lorsque les données sont nombreuses et de même type, au lieu de manipuler un grand nombre de variables, il est plus pratique de ranger ces variables dans une même structure de donnée appelé "TABLEAU".

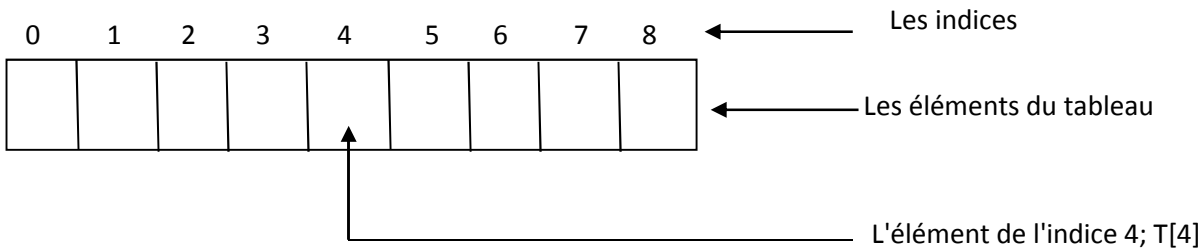
Pour l'exemple précédent le tableau permet de regrouper sous un même nom l'ensemble des notes des étudiants.

2. Le type tableau

Un tableau est une structure de données composé de plusieurs éléments (comme une variable) contiguë, le tableau peut stocker plusieurs valeurs de même type à la fois, chacune dans un élément du tableau.

Représentation graphique

Un tableau à une dimension est souvent représenté comme une suite de cases mémoires contigüe.



Taille d'un tableau

La taille ou longueur d'un tableau est le nombre des éléments qu'il contient

Remarques

- Tous les éléments d'un tableau portent le même nom (celui du tableau) et chacun à son indice (leur rang dans le tableau).
- Tous les éléments d'un tableau ont le même type.
- Un tableau possède une taille fixe qui ne pourra plus changer au cours de l'exécution du programme**.**

Déclaration d'un tableau

Avant de déclarer une variable de type tableau, il est préférable de déclarer d'abord un type tableau en indiquant le type de ses futures valeurs et en suite déclarer une ou plusieurs variables de type tableau puisque le type TABLEAU n'est pas un type prédéfinie comme entier, réel,...

Syntaxe

La déclaration d'un tableau suit la syntaxe suivante :

TABLEAU Nom _Tableau (borne_sup) **:** Type_éléments

Exemple : On déclare un tableau T de notes de 200 étudiants telle que chaque note est un nombre réel de la façon suivante

TABLEAU T (199) : **REEL**

Traduction en C :

Float t[199] ;

En mémoire, on va avoir une variable T comportant 200 éléments (cases mémoires) et dans chacune des cases, on peut placer une note d'un étudiant qui est de type réel.

Accès aux éléments d'un tableau et indice

Les éléments d'un tableau sont manipulés individuellement comme n'importe quelle variable: On peut modifier sa valeur dans le tableau, l'utiliser pour un calcul, l'afficher, dans laquelle lire une donnée, etc.

Indices des éléments d'un tableau

Chaque élément est repéré par un indice avec lequel on pourra accéder à sa valeur. L'indice du tableau peut avoir des valeurs de l'intervalle [borne inférieure, borne supérieure].

Syntaxe : L'accès à un élément d'un tableau se fait par le nom du tableau et l'indice de l'élément comme suit:

Nom_Tableau[indice]

Où:

Nom_Tableau: Est le nom du tableau contenant l'élément concerné par l'accès.

Indice: est l'indice de l'élément et il peut être:

- Une valeur; **Exemple** T[3].
- Une variable entière; **Exemple** T[i].
- Une expression entière positive; **Exemple** T[i+1].

Exemples : On peut accéder aux éléments du tableau T comme suit:

T [3] ← 4.5 ;
x ← (T[i] + T[i+1])/2;
ECRIRE (T[1])
Traduction en C :

```
T[3]=4.5 ;
X=(T[i] + T[i+1])/2;
Printf("%d", T[1]);
```

Exercice : donner la valeur de x après l'exécution de ce programme.

```
#include <stdio.h>
int main()
{
int z=1,x,t[]={1,5,6,8,9};
x=t[z+2]+t[4-z];
printf("%d",x);
   return 0;
}
```

Tableaux dynamiques

Il arrive fréquemment que l'on ne connaisse pas à l'avance le nombre d'éléments que devra comporter un tableau. On a la possibilité de déclarer le tableau sans préciser au départ son nombre d'éléments. Au cours du programme, que l'on va fixer ce nombre via une instruction de redimensionnement : Redim.

Exemple : on veut faire saisir des notes pour un calcul de moyenne, mais on ne sait pas combien il y aura de notes à saisir. Le début de l'algorithme sera quelque chose du genre :

```
Tableau Notes() : réel
Variable nb : entier
```

```
Début
Ecrire "Combien y a-t-il de notes à saisir ?"
Lire nb
Redim Notes(nb-1) …
```

Initialiser un Tableau

Il s'agit d'initialiser les valeurs des éléments d'un tableau par des valeurs nulles, on initialise le tableau puisque juste après sa déclaration, le tableau ne contient aucune valeur.

L'algorithme d'initialisation

ALGORITHME Initialisation_Tableau
TABLEAU T (199) : **REEL**
VAR ;
I : **ENTIER**
DEBUT
POUR I ←0 **à** 199
T[I] ← 0
FINPOUR
FIN

Remplir un Tableau

Cet algorithme permet de lire les notes des étudiants saisies par l'utilisateur à partir du clavier note par note et les mémoriser dans les éléments du tableau **T.**

ALGORITHME Remplir_Tabeau
TABLEAU T : (199) : REEL
VAR ;
I : ENTIER
DEBUT
POUR I ←0 à 199
LIRE T[I]
FINPOUR
FIN

Affichage des éléments d'un tableau

Cet algorithme va afficher tous les éléments du tableau T
ALGORITHME Afficher_Tableau
TABLEAU T : (199) : REEL
VAR ;
I : ENTIER
DEBUT
POUR I ←0 à 199
ECRIRE T[I]
FINPOUR
FIN

Recherche dans un Tableau ou parcours

On cherche dans un tableau sur une valeur soit saisi à partir du clavier ou une autre valeur dans une variable. Dès qu'on trouve la valeur cherchée on arrête la recherche.
Généralement on distingue deux méthodes de recherche:

- Une recherche séquentielle,
- Une recherche dichotomique (plus optimisée)

Recherche Séquentielle

```
ALGORITHME Recherche_Sequentielle
TABLEAU T : (199) : REEL
VAR ;
        I : ENTIER
        X : REEL            { X est la valeur à chercher dans le tableau}
        Trouve : BOOLEEN {cette variable va nous permettre de sortir dès qu'on
                          trouve la valeur cherché}
DEBUT
LIRE(X)
I ←0
Trouve :=  FAUX
TANTQUE (I<= 199) ET (Trouve = FAUX)
    SI (T[I] < > X) ALORS
        I ← I + 1
    SINON
        Trouve :=  VRAI
    FINSI
FINTANTQUE
SI (Trouve = VRAI) ALORS
    ECRIREX, 'existe dans le tableau T'
SINON
    ECRIRE X, 'n" existe pas dans le tableau T'
FINSI
FIN
```

Recherche Dichotomique : Pour pouvoir appliquer la recherche dichotomique, il faut que le tableau soit trié. La recherche dichotomique consiste à accélérer le temps de recherche d'un élément dans un tableau.

Technique : On commence par comparer X avec le contenu de l'élément du milieu du tableau : si X est inférieur à cette valeur alors on va continuer la recherche dans la partie gauche du tableau T avec modification de la borne supérieure, sinon on continuera la recherche dans la partie droite du tableau avec modification de la valeur de la borne inférieure. On s'arrêtera quand X serait égale à la valeur du milieu ou quand on dépasse les bornes du tableau.

```
ALGORITHME Recherche_Dichotomique
TABLEAU T : (199) : REEL
VAR
    Borne_Inf, Borne_sup, Milieu: ENTIER
    X : REEL            { X est la valeur à chercher dans le tableau}
DEBUT
LIRE X
Borne_Inf ←0
Borne_sup  ← 199
REPETER
    Milieu  ←  (Borne_Inf  + Borne_sup ) div 2
    SI X < T(Milieu) ALORS
        Borne_sup  ← Milieu – 1
    SINON
        Borne_Inf  Milieu +1
FINSI
JUSQU'A (X=T(Milieu)) OU (Borne_Inf  > Borne_sup  )
FIN
```

3. Les tableaux multidimensionnels

Motivation

Prenons toujours l'exemple des notes des étudiants, mais dans ce cas chaque étudiant parmi les 200 étudiant de la promotion à une dans chacune des 6 matières étudié en première année

Pour gérer les notes de tous les étudiants on doit déclarer un tableau de 200 éléments pour chaque matière. Mais cette solution peut poser des problèmes d'accès, de manipulation et de parcours avec un nombre important de tableaux.

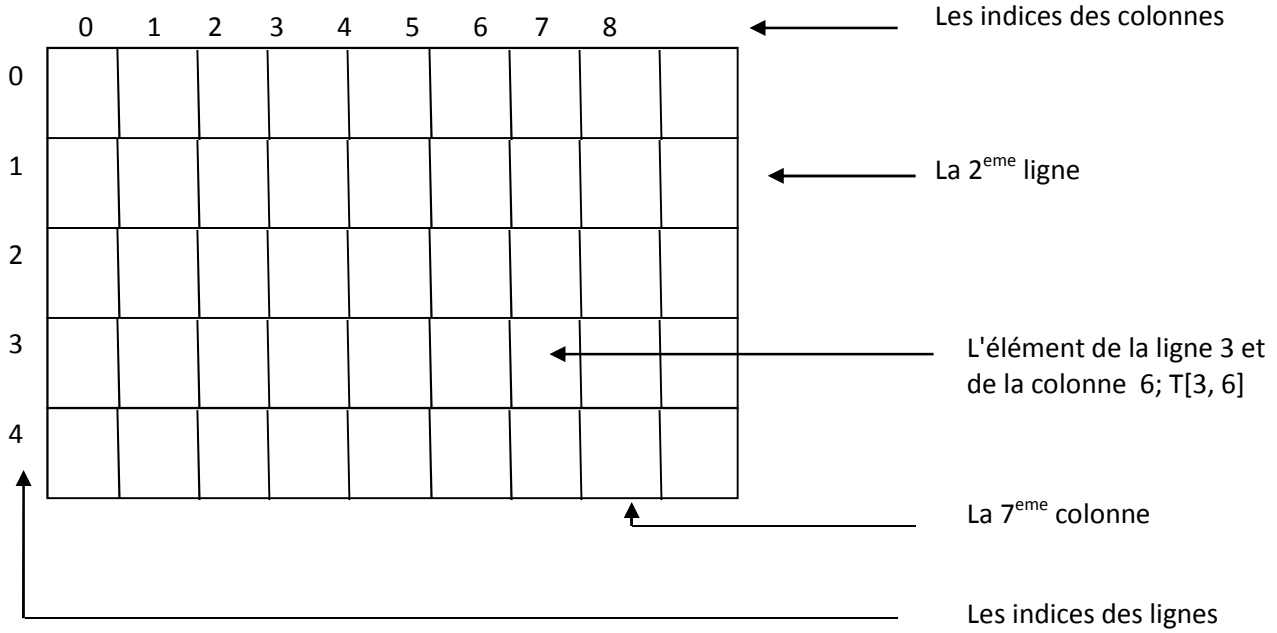
Alors la solution est d'utiliser une seul structure de données qui regroupe les notes de tous les étudiants dans les toutes les matières étudiées qui sont de même type. Une telle structure est le tableau à deux dimensions ou "Matrice".

Définition

Un tableau à deux dimensions appelé également matrice est une structure de données composé d'un **ensemble de lignes** et un **ensemble de colonne**, et chacun des lignes et de colonnes composé à son tour d'un ensemble d'éléments permettant de stocker des données de même type.
Il est caractérisé donc par son nombre de lignes et son nombre de colonnes.
Un tableau est dit ***vecteur*** si le nombre de dimension est égal à 1, ***matrice de dimension n*** si le nombre de dimension est égale à n >1.

Représentation graphique

Un tableau à deux dimensions est souvent représenté par un certain nombre de lignes et de colonnes. Un élément est repéré par son numéro de ligne et son numéro de colonne.



Taille d'une matrice

La taille d'un tableau à deux dimensions ou matrice est le nombre de lignes multiplié par le nombre de colonnes.

Remarques

Mêmes remarques que le tableau à une dimension

- Tous les éléments d'une matrice portent le même nom (celui de la matrice) et chacun à deux indices, celui de ligne et de colonne.
- Tous les éléments d'une matrice ont le même type.
- Une matrice aussi, possède une taille fixe qui ne pourra plus changer au cours de l'exécution du programme**.**

Déclaration d'une matrice

Syntaxe

La déclaration d'un tableau à deux dimensions suit la syntaxe suivante :
TABLEAU Nom_Matrice (born_Ligne, born_Colonne) **:** Type_éléments

Exemple

On déclare une matrice M de notes de 200 étudiants dans six matières de la façon suivante:

TABLEAU M [199, 5) : **REEL**
Traduction en C :
Float m[199][5] ;

En mémoire, on va avoir une variable M comportant 200 lignes et 6 colonnes. Alors la matrice est composée de 1200 cases mémoires élémentaires.

Accès aux éléments d'une matrice

L'accès à un élément de la matrice ne peut se faire qu'avec deux indices : un indice pour identifier la ligne et un autre pour identifier la colonne de l'élément concerné.

Syntaxe

L'accès à un élément d'une matrice se fait par le nom de la matrice, l'indice de la ligne de l'élément et l'indice de la colonne de l'élément comme suit:

Nom_Matrice [Indice_Lignes, Indice_Colonnes]

Où:

Nom_Matrice: Est le nom de la matrice contenant l'élément concerné par l'accès.
Indice_Lignes et **Indice_Colonnes :** Sont les indices de ligne et de colonne respectivement, chacun des deux indice peuvent être:

- Une valeur; **Exemple** M [3,5].
- Une variable entière; **Exemple** M [5, i].
- Une expression entière positive; **Exemple** M [i, j+1].

Exemples : On peut accéder aux éléments de la matrice M de type Mat_Notes déclarée précédemment comme suit:

M [3, 6] ← 15
x ← (M[I,j] + M [I, j+1])+ M [I, j+2])/3
ECRIRE M[1, 5]
Traduction en C:
M[3][6]=15;
x = (M[I][j] + M [I][j+1])+ M [I][j+2])/3
printf("%f", M[1][5])

Opérations de base sur une matrice : Pour représenter les différentes opérations, on va utiliser la matrice des notes de type Mat_Notes déclarée précédemment.

Remarque: Les parcours des matrices se font avec des boucles imbriquées : une boucle externe pour parcourir les lignes et une boucle interne pour chaque ligne, cette dernière fait le parcours des éléments dans toutes les colonnes de cette ligne.

Initialiser une matrice

ALGORITHME Initialisation_Matrice

TABLEAU M (199, 5) : **REEL**
VAR

```
        i, j : ENTIER {les indices de lignes et de colonnes respectivement}
DEBUT
    POUR i ← 0 à 199
        POUR j de 0 à 5
            M[i, j] ←  0
        FINPOUR
    FINPOUR
FIN
```

Remplir une matrice

Cet algorithme permet de lire les notes des étudiants saisies par l'utilisateur à partir du clavier note par note et les mémoriser dans les éléments de la matrice M de type Mat_Notes**.**

```
ALGORITHME Remplir_Matrice
TABLEAU M (199, 5) : REEL
VAR
        i, j : ENTIER {les indices de lignes et de colonnes respectivement}
DEBUT
    POUR i ← 0 à 199
        POUR j de 0 à 5
            LIRE M[i, j]
        FINPOUR
    FINPOUR
FIN
```

Affichage les éléments d'une matrice

Cet algorithme va afficher tous les éléments de la matrice M de type Mat_Notes

```
ALGORITHME Afficher_Matrice
TABLEAU M (199, 5) : REEL
VAR
        i, j : ENTIER {les indices de lignes et de colonnes respectivement}
DEBUT
    POUR i ← 0 à 199
        POUR j de 0 à 5
        ECRIRE M[i, j]
    FINPOUR
FIN
```

Recherche dans une matrice

```
ALGORITHME Recherche_Matrice
TABLEAU M (199, 5) : REEL
VAR
        i, j : ENTIER {les indices de lignes et de colonnes respectivement}
        X : REEL            { X est la valeur à chercher dans la matrice}
        Trouve : BOOLEEN {cette variable va nous permettre de sortir dès qu'on
                          trouve la valeur cherché}
DEBUT
LIRE(X)
i ←0
Trouve :=  FAUX
TANTQUE (i<= 199) ET (Trouve = FAUX) FAIRE
    j ←0
```

```
        TANTQUE (j<= 5) ET (Trouve = FAUX) FAIRE
                SI (M[i, j] < > X) ALORS
                        j ← j + 1
                SINON
                        Trouve := VRAI
                FINSI
        FINTANTQUE
        i ← i + 1
FINTANTQUE
SI (Trouve = VRAI) ALORS
        ECRIRE(X, 'existe dans la matrice M')
SINON
        ECRIRE (X, 'n" existe pas dans la matrice M')
FINSI
FIN
```

4. Les chaînes de caractères

Une catégorie privilégiée de fonctions est celle qui nous permet de manipuler des chaînes de caractères. Nous avons déjà vu qu'on pouvait facilement « coller » deux chaînes l'une à l'autre avec l'opérateur de concaténation &. Mais ce que nous ne pouvions pas faire, et qui va être maintenant possible, c'est pratiquer des extractions de chaînes (moins douloureuses, il faut le noter, que les extractions dentaires).

Presque tous les langages proposent les fonctions suivantes :

- Len(chaîne) : renvoie le nombre de caractères d'une chaîne
- Mid(chaîne,n1,n2) : renvoie un extrait de la chaîne, commençant au caractère n1 et faisant n2 caractères de long.

Ce sont les deux seules fonctions de chaînes réellement indispensables. Cependant, pour nous épargner des algorithmes fastidieux, les langages proposent également :

- Left(chaîne,n) : renvoie les n caractères les plus à gauche dans chaîne.
- Right(chaîne,n) : renvoie les n caractères les plus à droite dans chaîne
- Trouve(chaîne1,chaîne2) : renvoie un nombre correspondant à la position de chaîne2 dans chaîne1. Si chaîne2 n'est pas comprise dans chaîne1, la fonction renvoie zéro.

Exemples :

Len("Bonjour, ça va ?")	vaut	16
Len("")	vaut	0
Mid("Zorro is back", 4, 7)	vaut	"ro is b"
Mid("Zorro is back", 12, 1)	vaut	"c"
Left("Et pourtant…", 8)	vaut	"Et pourt"
Right("Et pourtant…", 4)	vaut	"t…"
Trouve("Un pur bonheur", "pur")	vaut	4
Trouve("Un pur bonheur", "techno")	vaut	0

Il existe aussi dans tous les langages une fonction qui renvoie le caractère correspondant à un code Ascii donné (fonction Asc), et Lycée de Versailles (fonction Chr) :

```
Asc("N")        vaut    78
Chr(63)         vaut    "?"
```

Traduction en C

Déclaration d'une chaîne

Une chaîne de caractères est un tableau de type char. La déclaration est identique à un tableau normal:

```
char <nom_chaine> [<dimension>]
```

Initialiser une chaîne de caractères

Nous pouvons indiquer explicitement le nombre d'octets à réserver, si celui-ci est supérieur ou égal à la longueur de la chaîne d'initialisation.

Exemples d'initialisation

char MACHAINE[ ] = "Hello"; char MACHAINE[6] = "Hello";
char MACHAINE[ ] = {'H','e','l','l','o','\0'}; char MACHAINE[8] = "Hello";

Pour accéder à ses éléments, on suit la logique d'un tableau.

Exemple:
char A[6] = "Hello";
A[0] contient 'H', A[1] contient 'e' … A[5] contient '\0'.

Une chaîne de caractères est une variable : on peut utiliser des opérations logiques et mathématiques.

Exemples

"ABC" précède "BCD" car 'A'<'B' "ABC" précède "B" car 'A'<'B' "Abc" précède "abc" car 'A'<'a'

```
if (C>='0' && C<='9') printf("Chiffre\n", C);
if (C>='A' && C<='Z') printf("Majuscule\n", C); if
```

Des fonctions de traitement des chaînes de caractères sont disponibles dans les bibliothèques standards:

<stdio.h> :

scanf, printf en utilisant %s dans le format

attention: scanf prend une adresse en argument (&x), une chaîne de caractères étant un tableau (ie, l'adresse du premier élément), il n'y a pas de &.

puts: puts(MACHAINE); est équivalent à printf("%s\n",TXT);

gets: gets(MACHAINE); lit une ligne jusqu'au retour chariot et remplace le '\n' par '\0' dans l'affectation de la chaîne.

<string>:

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>

Table des matières