1. INTRODUCTION

Supposons qu'on ait une dizaine de variables de même type (entier par exemple) qu'on appellera A_1, A_2, A_3, A_4, … , A_{10}. Ces dix variables simples occupent chacune un emplacement physique en mémoire, après leur déclaration. Il y a une autre possibilité : Les rassembler toutes dans une seule structure appelée « tableau » dans dix emplacements contigus mais avec un seul nom (tableau). Chaque élément de ce tableau est repéré par un indice (son rang dans le tableau).

2. DECLARATION

En algorithmique, la déclaration d'un tableau se fait à la clause variable par la syntaxe suivante :

identificateur-tableau : tableau [1..N] de type ;

Comme vous pouvez le remarquer, pour définir un tableau on doit donner son nom, son type et sa taille (nombre de cases). ***En langage C***, La définition d'un tableau à une dimension (vecteur) suit la syntaxe suivante :

Syntaxe :

Type *Identificateur* **[***Taille constante***] ;**

- La ***Taille*** du tableau est le ***nombre de ses éléments***. Elle ne peut être une variable. ***Elle doit être une constante*** définie avant ou au moment de la déclaration.

C'est-à-dire Type-case nom-tableau [nombre-case] ;

REMARQUE :

- Un tableau est un ensemble de données qui sont toutes de même type.
- Le tableau a un nom.
- Les données possèdent un identificateur unique : le nom du tableau.
- La taille d'un tableau correspond au nombre de cases.
- Les données se différentient par leur numéro d'indice.
- le nombre de cases est entre crochets et non entre parenthèses.

Exemples :

Déclaration de tableaux

```
#include <stdio.h>
#define taille1  5      /*taille1: constante de valeur 5*/
#define taille2  3      /*taille2: constante de valeur 3*/
main( )
{
int a [taille1];        /*a: tableau de 5 entiers*/
a[0]=15;                /*la première case du tableau a reçoit 15*/
char b [taille2];       /*b: tableau de 3 caractères*/
b[0]='x';               /*la première case du tableau b reçoit la lettre x*/
b[1]='y';               /*la deuxième case du tableau b reçoit la lettre y*/
float c [10];           /*c: tableau de 10 réels*/
scanf("%f", &c[0]);     /*lit un réel et l'affecte à la première case de c*/
}
```

ATTENTION:

- En C, Les indices d'un tableau sont des entiers commençant **à 0.**
- L'affectation élément par élément d'un tableau B à un autre tableau A (A[i]=B[i]) réalise une copie de B dans A. Une affectation "brutale" de B à A (A=B) n'est pas possible .

3. INITIALISATION

Syntaxe :
Type *Identificateur* **[***Taille constante***] = {***Valeur1***,** *Valeur2*,…,*Valeurn***};**

Initialisation d'un tableau à plusieurs dimensions

```
#define taille1 3  /*taille1: constante de valeur 3*/
#define taille2 2  /*taille2: constante de valeur 2*/
main( )
{
int M [taille1][taille2]={{0,1},{2,3},{4,5}};          /*initialisation ligne par ligne*/
}                                                      /*M[0]={0,1}, M[1]={2,3} et M[2]={4,5}*/
```

4. LECTURE ET AFFICHAGE

Lecture et affichage de tableaux

```
#include <stdio.h>
#define taille 20                  /*taille: constante de valeur 20*/
main( )
{
int i, t [taille];                 /*t: tableau de 20 entiers*/
for(i=0;i<taille;i++)              /*lit les 20 entiers élément par élément*/
      scanf ("%d",&t[i]);
for(i=0;i<taille;i++)              /*affiche les 20 entiers élément par élément*/
      printf ("%d\n",t[i]);
}
```

5. L'AFFECTATION

L'affectation de valeurs aux éléments d'un tableau se fait également individuellement (comme pour la lecture et l'affichage).

Affectation de valeurs à un tableau

```
#include <stdio.h>
#define taille 20              /*taille: constante de valeur 20*/
main( )
{
int i, t [taille];             /*t: tableau de 20 entiers*/
for(i=0;i<taille;i++)          /*affecte i à chaque élément d'indice i*/
      t[i]=i;
}
```

6. DECLARATION D'UN TABLEAU A DEUX DIMENSIONS

Syntaxe :

Type *Identificateur* **[***Taille1***]** **[***Taille2***]** … **[***Taille n***] ;**

- Taillei est la taille de la dimension i. Elle doit être une constante définie avant ou au moment de la déclaration.
- ***Un élément*** d'un tableau t à n dimensions est repéré par ses indices, sous forme ***t[i1][i2]…[in]***.

A deux dimensions :

<NomTableau>[N (nbr de lignes),M (nbr de colonnes)] : type

Déclaration et lecture d'un tableau à deux dimensions

```
/*Ce programme lit le nombre de buts marqués par chacun des 11 joueurs de 8 équipes*/

      #include <stdio.h>
      #define   taille1   8          /*taille1: constante de valeur 8*/
      #define   taille2   11         /*taille2: constante de valeur 11*/

      main( )

      {
      int t [taille1][taille2];      /*t : matrice de 8 lignes, 11 colonnes*/
      int i, j;

      for(i=0;i<taille1;i++)         /*lit les éléments de t ligne par ligne*/
            for (j=0; j<taille2;j++)
                  scanf ("%d",&t[i][j]);
      }
```