MAI

LICENCE 2

# MATHÉMATIQUES APPLIQUÉES À L'INFORMATIQUE

## Programmation en langage C

Enseignant : M. Khalifa SYLLA

# LES TABLEAUX

UVS
UNIVERSITÉ VIRTUELLE DU SÉNÉGAL

***Définition***

Un tableau est une structure de données qui permet de stocker de manière contiguë(les uns à la suite des autres) en mémoire un ensemble de valeurs de même type.
Les éléments du tableau sont de n'importe quel type du langage, à savoir:

- Types élémentaires des données de type simple : int, char, float, long, double...
- Type tableau
- Type structure
- Type pointeur

Lorsque le tableau est composé de données de type élémentaire, on parle de tableau unidimensionnel (ou vecteur).
Lorsque celui-ci contient lui-même d'autres tableaux on parle alors de tableaux multidimensionnels (ou matrice).

**I – Tableaux à une dimension : vecteur**

Un tableau (unidimensionnel) est une variable structurée formée d'un nombre entier N de variables du même type, qui sont appelées les *composantes* du tableau. Le nombre de composantes N est alors la *dimension* du tableau.

**1. Déclaration**

Pour déclarer un tableau à une dimension, il faut donner:

- son ***nom*** ou identificateur de la variable de type tableau, par exemple TAB,
- sa ***dimension*** : nombre d'éléments;
- et le ***type de données*** des éléments qui composent ce tableau.

**type** nom_du_tableau [nombre_éléments];
Exemple:

```
#define N 6
int t [N];          // définit le tableau t qui peut contenir 6 éléments de types entiers.
```

***Mémorisation***

Si un tableau est formé de N composantes et si une composante a besoin de M octets en mémoire, alors le tableau occupera de N*M octets.

## 2. Initialisation et réservation automatique

### 2.1 – Initialisation

Lors de la définition on utilise les signes { } pour lister les futurs éléments du tableau.

```
#define N 7
char t[N]={'B','o','n','j','o','u','r'};      // Ou encore char t[N]= ''Bonjour'';
int t[N]={1,2,3,4,5,6,7} ;
int t[N]={1,2,3};              // initialise les trois premiers éléments et fixe les autres à 0.
```

### 2.2 – Réservation automatique

Il est possible de ne pas spécifier la taille du tableau, le compilateur prend alors le nombre d'éléments spécifiés dans l'initialisation;

```
int t[]={1,2,3};     char t[]={'B','o','n','j','o','u','r'};
```

Cas particulier

char c[]=''bonjour'', le compilateur ajoute le caractère null !!!

## 3. Indexation

Pour accéder à un élément du tableau, on le fait via son indice (position). l'accès au premier élément du tableau se fait par **T[0]**

Pour référencer un élément du tableau, on utilise la syntaxe suivante:

**nom_du_tableau[exp]**

Où **exp** est une expression qui renvoie une valeur entière positive ou nulle qui est l'indice de l'élément dans le tableau.

Cette instruction renvoie alors l'élément recherché qui est de type le type des éléments du tableau.

Exemple:

int tab[4],i=2;

tab[0]; : désigne le premier élément du tableau

tab[i+1] ; : désigne le dernier élément du tableau

tableau commence à l'indice 0 Notre tableau de 4 int a donc les indices 0, 1, 2 et 3. Il n'y a pas d'indice 4 dans un tableau de 4 éléments.

Maintenant que l'on peut retrouver un élément il est possible de travailler avec comme une variable d'un type donné.

## 4. Opérations de saisie et d'affichage

### 4.1 – Saisie d'un tableau A de cinq éléments

```
int main()
{
    int A[5];
    int I;  /* Compteur */
    for (I=0; I<5; I++)
       scanf("%d", &A[I]);
    return 0;
}
```

### 4.2 – Affichage d'un tableau A de cinq éléments

```
main()
{
    int A[5];
    int I;  /* Compteur */
    for (I=0; I<5; I++)
       printf("%d ", A[I]);
    return 0;
    printf("\n");
}
```

***Remarque :*** Avant de pouvoir afficher les composantes d'un tableau, il faut évidemment leur affecter des valeurs.

## II – Tableau à deux dimensions : matrice

Un tableau à deux dimensions **A** est à interpréter comme un tableau unidimensionnel de taille **L** dont chaque élément est un tableau à une dimension de taille **C**.

On appelle **L** le *nombre de lignes* du tableau et **C** le *nombre de colonnes* du tableau. L et C sont alors les deux *dimensions* du tableau. Un tableau à deux dimensions contient donc *L*C éléments*.

En faisant le rapprochement avec les mathématiques, on peut dire *"**A** est une **matrice** de dimensions **L** et **C**"*.

### 1. Déclaration

Pour déclarer un tableau à deux dimensions, il faut donner:

- son ***nom*** ou identificateur de la variable de type tableau,
- sa ***dimension de lignes*** : nombre de lignes;
- sa ***dimension de colonnes*** : nombre de colonnes;
- et le ***type de données*** des éléments qui composent ce tableau.

**type nom_du_tableau [nbr_élts_1] [nbr_élts_2]------ [nbr_élts_n];**

**Exemple**

int t[3][5];

Crée un tableau de 15 éléments de type int.

Le tableau suivant nous montre les indexations.

| | | | | |
|---|---|---|---|---|
| t[0][0] | t[0][1] | t[0][2] | t[0][3] | t[0][4] |
| t[1][0] | t[1][1] | t[1][2] | t[1][3] | t[1][4] |
| t[2][0] | t[2][1] | t[2][2] | t[2][3] | t[2][4] |

***Mémorisation***

Comme pour les tableaux à une dimension, le nom d'un tableau est le représentant de ***l'adresse du premier élément*** du tableau (c.-à-d. l'adresse de la première ***ligne*** du tableau). Les composantes d'un tableau à deux dimensions sont stockées ligne par ligne dans la mémoire.

Un tableau de dimensions L et C, formé de composantes dont chacune a besoin de M octets, occupera L*C*M octets en mémoire.

### 2. Initialisation et réservation automatique

#### 2.1 – Initialisation

Lors de la déclaration d'un tableau, on peut initialiser les composantes du tableau, en indiquant la liste des valeurs respectives entre accolades. A l'intérieur de la liste, les composantes de chaque ligne du tableau sont encore une fois comprises entre accolades. Pour améliorer la lisibilité des programmes, on peut indiquer les composantes dans plusieurs lignes.

**Exemples**

```
int A[3][10] ={{ 0,10,20,30,40,50,60,70,80,90},
               {10,11,12,13,14,15,16,17,18,19},
               {1,12,23,34,45,56,67,78,89,90}};

float B[3][2] = {{-1.05,   -1.10  },
                 {86e-5,    87e-5 },
                 {-12.5E4, -12.3E4}};
```

Lors de l'initialisation, les valeurs sont affectées ligne par ligne en passant de gauche à droite. Nous ne devons pas nécessairement indiquer toutes les valeurs: Les valeurs manquantes seront initialisées par zéro. Il est cependant défendu d'indiquer trop de valeurs pour un tableau.

### 2.2 – Réservation automatique

Si le nombre de **lignes L** n'est pas indiqué explicitement lors de l'initialisation, l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

```
int A[][10] = {{ 0,10,20,30,40,50,60,70,80,90},
               {10,11,12,13,14,15,16,17,18,19},
               { 1,12,23,34,45,56,67,78,89,90}};
```

réservation de 3*10*2 = 60 octets

```
float B[][2] = {{-1.05,   -1.10  },
                {86e-5,    87e-5 },
                {-12.5E4, -12.3E4}};
```

réservation de 3*2*4 = 24 octets

## 3. Indexation

Considérons un tableau A de dimensions **L** et **C**. En langage C :

- les indices du tableau varient de **0** à **L-1**, respectivement de **0** à **C-1**.
- la composante de la N$^{ième}$ ligne et M$^{ième}$ colonne est notée: ***A[N-1][M-1]***

**4. Opérations de saisie et d'affichage**

**4.1 – Saisie d'un tableau A de cinq ligne et dix colonnes**

```
main()
{
        int A[5][10];
        int I,J;
        /* Pour chaque ligne ... */
        for (I=0; I<5; I++)
          /* ... considérer chaque composante */
             for (J=0; J<10; J++)
                    scanf("%d", &A[I][J]);
        return 0;
}
```

**4.2 – Affichage d'un tableau A de cinq ligne et dix colonnes**

```
main()
{
     int A[5][10];
     int I,J;
     /* Pour chaque ligne ... */
     for (I=0; I<5; I++)
       {
            /* ... considérer chaque composante */
            for (J=0; J<10; J++)
                  printf("%7d", A[I][J]);
            /* Retour à la ligne */
            printf("\n");
       }
     return 0;
}
```

**III– Chaine de caractères**

Il n'existe pas de type spécial chaîne ou *string* en C. Une chaîne de caractères est traitée comme un *tableau à une dimension de caractères* (vecteur de caractères). Il existe quand même des notations particulières et une bonne quantité de fonctions spéciales pour le traitement de tableaux de caractères.

**1. Déclaration**

***char nomch[taille]***

Exemples

***char NOM [20];***
***char PRENOM [30];***
***char PHRASE [200];***

Comment affecter la valeur d'une chaîne de caractère avec scanf();

char c[50];

printf(''Donner une chaîne'');

scanf(''%s'', c)

Pour afficher:

printf(''%s'',c)

Deux fonctions qui permettent un traitement d'entrée sortie simple avec les string.

**gets, puts**.

**gets**

Lit une chaîne de caractères dans l'entrée standard et la stocke dans le tableau de caractères associé.

*char c[45];*

*printf(''Donner la chaîne''); gets(c);*

**puts**

Affiche la chaîne associée sur la sortie standard et va à la ligne suivante.

puts(s);

## 2. Opérations sur les strings

### Longueur d'une chaine

**strlen (nom_du tableau);**

Donne la longueur du tableau en octets, donc le nombre d'éléments du tableau

### Concaténation de deux chaines

**strcat (nom_premier_tableau, nom_deuxième_tableau);**

concatène la seconde chaîne à la fin de la première.

### Comparaison

**strcmp (nom_premier_tableau, nom_deuxième_tableau);**

Compare les strings caractère par caractère dans l'ordre lexicographique, jusqu'à la rencontre d'une différence ou du caractère nul.

strcmp renvoie une valeur nulle si les deux chaînes sont semblables, sinon une valeur négative si la première chaîne est inférieure à la seconde et une valeur positive dans le cas contraire.

### Les fonctions de <stdlib>

La bibliothèque <*stdlib*> contient des déclarations de fonctions pour la conversion de nombres en chaînes de caractères et vice-versa.

Les trois fonctions définies ci-dessous correspondent au standard ANSI-C et sont portables. Le symbole <s> peut être remplacé par :

- une chaîne de caractères constante
- le nom d'une variable déclarée comme tableau de **char**
- un pointeur sur **char**

**Conversion de chaînes de caractères en nombres**

| | |
|---|---|
| **atoi(<s>)** | retourne la valeur numérique représentée par <s> comme **int** |
| **atol(<s>)** | retourne la valeur numérique représentée par <s> comme **long** |
| **atof(<s>)** | retourne la valeur numérique représentée par <s> comme **double** |

**Règles générales pour la conversion:**

- Les espaces au début d'une chaîne sont ignorés
- Il n'y a pas de contrôle du domaine de la cible
- La conversion s'arrête au premier caractère non convertible
- Pour une chaîne non convertible, les fonctions retournent zéro

## Les fonctions de <ctype>

| ***La fonction:*** | ***retourne une valeur différente de zéro,*** |
|---|---|
| **isupper(<c>)** | si <c> est une majuscule **('A'...'Z')** |
| **islower(<c** | si <c> est une |

| | |
|---|---|
| **>)** | minuscule **('a'...'z')** |
| **isdigit(<c>)** | si <c> est un chiffre décimal **('0'...'9')** |
| **isalpha(<c>)** | si **islower(<c>)** ou **isupper(<c>)** |
| **isalnum(<c>)** | si **isalpha(<c>)** ou **isdigit(<c>)** |
| **isxdigit(<c>)** | si <c> est un chiffre hexadécimal (**'0'...'9'** ou **'A'...'F'** ou **'a'...'f'**) |
| **isspace(<c>)** | si <c> est un signe d'espacement (' ', **'\t'**, **'\n'**, **'\r'**, **'\f'**) |

Les fonctions de ***conversion*** suivantes fournissent une valeur du type **int** qui peut être représentée comme caractère; la valeur originale de <c> reste inchangée:

| | |
|---|---|
| **tolower(<c>)** | retourne <c> converti en minuscule si <c> est une majuscule |
| **toupper(<c>)** | retourne <c> converti en majuscule si <c> est une minuscule |