5. Tableaux 2D

déclaration

exemple :

```
#define MAX_L 2
#define MAX_C 3

int tab[MAX_L][MAX_C];
```

accès

tab[i][j] : accès à la case (i,j) du tableau

L'action favorite pour parcourir un tableau 2D est 2 boucles for imbriquées

l'initialisation

- à la définition :

exemple :

```
#define MAX_L 2
#define MAX_C 3
double M[MAX_L][MAX_C] = {{37.2,37.5, -2.45},  // 1ere ligne
                          {38.4, 40.5, 43.2}}; //  2e  ligne
```

Ou

```
double M[MAX_L][MAX_C] = {37.2,37.5, -2.45,  // 1ere ligne
                          {38.4, 40.5, 43.2}; //  2e  ligne
```

L'initialisation s'effectue ligne par ligne avec les listes de valeurs d'initialisation.

- par lecture

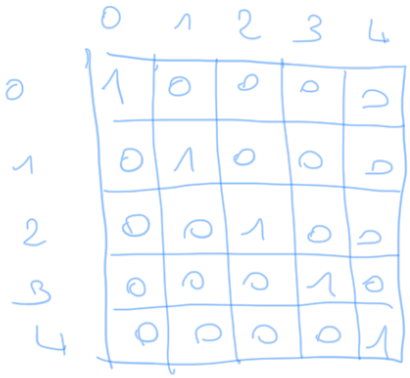
exemple :

```
for (i=0;i< MAX_L;i++)
  for (j=0;j<MAX_C;j++){
     printf("donner la valeur de la case (%d,%d) : " , i,j) ;
     scanf("%f",&M[i][j]);
     }
```

- par calcul

exemple :

```
#define MAX_L 5
#define MAX_C 5
for (i=0;i< MAX_L;i++)
  for (j=0;j<MAX_C;j++)
     if (j==i) M[i][j]=1;
     else M[i][j]=0;
```



Passage d'un tableau 2D en paramètre :

```
void affiche(double T[][MAX_C])
{int i,j;
   for (i=0;i< MAX_L;i++)
    for (j=0;j<MAX_C;j++)
      printf("\n tableau[%d][%d] = %4.3f",i,j,T[i][j]);

}
```

Commentaires:

Il ne faut pas mentionner la 1ere dimension (MAX_L) mais la 2d dimension du tableau est obligatoire (MAX_C).

exemple appel de la fonction précédente :

```
affiche(M);
```

Relation tableau-pointeur pour les tableaux 2D:

Soit par exemple

```
#define MAX_L 2
#define MAX_C 3
int M[MAX_L][MAX_C]= {7,8,9,15,16,17};
```

On le représente généralement de la manière suivante

7	8	9
15	16	17

Or en mémoire il est stocké de la manière suivante :

7	8	9	15	16	17

*M *M+1 *M+5

&M[1][2]

*M + (1*3 +2)

*M + 5 = adresse du dernier element

Un tableau à 2 dimensions est en fait un tableau de tableau c'est-à-dire un tableau de pointeurs où chaque pointeur pointe sur le premier élément d'une ligne

M ->	M[0]	->	7	8	9
	M[1]	->	15	16	17

M est de type int **x

M est un pointeur de pointeur d'entier

Quand on écrit dans le programme : `&M[i][j]`
la machine calcule en fait l'expression mathématique suivante :
`*M + (i x MAX_C + j)`

la relation avec les pointeurs

*M <=> M[0]
int *
pointe sur
M[0][0]

`tab[1][2]`

est la même chose que
`*(tab[1]+2)`

qui est la même chose que
`*(*tab + 1*MAX_C+2)` => `*(*tab + 5)`

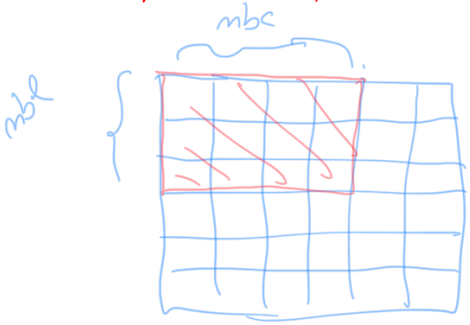
Conseil pour LO01 : utilisez les notations `tab[i][j]`.

Occupation partielle d'un tableau à plusieurs dimensions :

- identique à une dimension : on déclare un tableau avec des dimensions constantes correspondant au nombre de ligne max et nom de colonne max.
- -il faut gérer deux variables nbl pour le nombre de lignes occupées et nbc pour le nombre de colonnes occupées.
- Les boucles de parcours de tableau sont conditionnées sur ces variables

Exemple

```
void afficheMatrice(int M[][MAX_C], int nbl, int nbc)
{int i,j;
    for (i=0;i<nbl;i++){
     for (j=0;j<nbc;j++)
       printf("\t%d",T[i][j]);
     putchar('\n') ;
     }
}
```



6. Tableaux et chaînes de caractères

Attention : il ne faut pas confondre tableau de caractères et chaîne de caractères, ce n'est pas tout à fait la même chose.

1.Tableaux de caractères

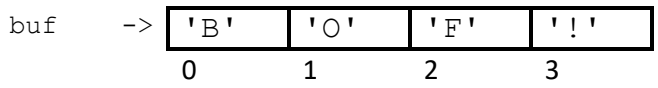
Définition d'un tableau de caractères

```
char buf[4] ;
```

Accès aux ~~champs~~ composantes

```
buf[0] = 'B';
buf[1] = 'O';
buf[2] = 'F';
buf[3] = '!';
```

Voici ce que contient le tableau :



Initialisation :

- Initialisation à la définition

```
char buf[6] = {'b', 'o', 'f', '!'} ;
```

2.Chaînes de caractères

Pour distinguer un tableau de caractère d'une chaîne de caractères, le langage C utilise un caractère spécial de la table des codes ASCII, c'est le '\0' (backslash zéro). Il sert de marqueur pour indiquer la fin d'une chaîne de caractère.
Une chaîne de caractères est stockée dans un tableau de caractère. Pour qu'on puisse le considérer comme une chaîne de caractère il faut que le marqueur de fin de chaîne '\0' soit dans le tableau.
Tous les caractères qui précèdent font partie de la chaîne de caractères et on ne se préoccupe pas de ceux qui suivent le marqueur.

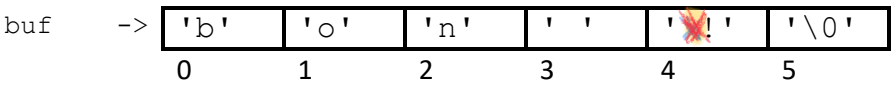
les constantes chaines de caracteres sont entre " "

Initialisation d'une chaîne :

- à la déclaration :

```
char buf[6] = {'b', 'o', 'n', ' ', '!', '\0'} ;
```

on obtient le tableau :



Le calcul de la taille peut être fait à l'initialisation.

de la même manière, les trois instructions suivantes sont identiques. Les tableaux seront de taille 8.

```
char toto[ ] = "bonjour";
char *titi = "bonjour";
char tutu[ ]={'b','o','n','j','o','u','r','\0'};
```

- par saisie :

```
char chaine1[6] = {'a', 'u', 'x', '\0'} ; // chaine "aux"
char chaine2[12] ;

scanf("%s",chaine2); //remplit la chaine2 et met le marqueur de fin '\0'
scanf("%s",chaine1); //efface la chaine1 "aux" avec la chaine lue au
                     // clavier
```

Attention, la fonction `scanf("%s",…)` s'arrête au 1er espace rencontré. Pour aller jusqu'au « enter », il faut utiliser `gets(char *S)`

La fonction `puts(char *S)` permet aussi d'afficher une chaine passée en paramètre.
On peut évidemment remplir case par case le tableau sans oublier d'ajouter le '\0' à la fin.

On peut utiliser les fonctions prédéfinies dans la bibliothèque
`<string.h>`

Quelques fonctions (il en existe beaucoup d'autres):

`int strlen(char *S);` retourne la longueur de la chaine S
`char* strcpy(char *S1, char *S2);` copie S2 dans S1 (retourne pointeur S1, pas utile)
`char* strncpy(char *S1, char *S2, int n);` copie n premiers caractères de S2 dans S1 (retourne pointeur S1, pas utile)
`char* strcat(char *S1, char *S2);` concatène S2 à S1 (retourne pointeur S1, pas utile)
`char* strncat(char *S1, char *S2, int n);` concatène n premiers caractères de S2 à S1 (retourne pointeur S1, pas utile)
`int strcmp(char *S1, char *S2);` renvoie 0 si les chaînes sont identiques ; une valeur positive si S1 est supérieure à S2 et négative si S2 est supérieure à S1

Exemple :

```
#include <string.h>
#define MAX_LEN 30
char mot[MAX_LEN];
char phrase[MAX_LEN] ;

int nbc ; // nombre de caractères

scanf("%s",mot);
nbc = strlen(mot);
strcpy(mot,"bonjour");
strncpy(mot,"aurevoir",3)
scanf("%s",phrase);
strcat(phrase,", ");
strncat(phrase,"bonsoir a tous",6);
strncat(phrase,mot);
```

remarque :

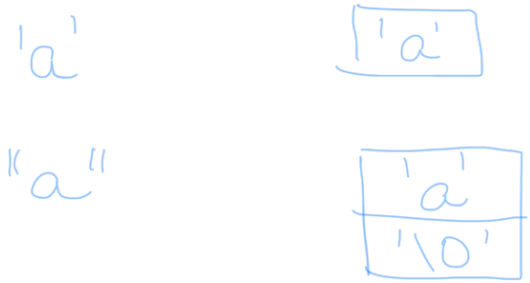
Une chaîne c'est un tableau de caractères et l'utilisation d'une convention. On peut donc la manipuler comme un tableau.

```
int longueur_chaine(char * ch1)
{
  int nb_carac = 0;
  while (ch1[nb_carac]!='\0')
    nb_carac++;
  return nb_carac;
}
```

Pour appeler cette fonction :

```
char mot[MAX_LEN]= "unmotarallonge"
longueur_chaine(mot);
longueur_chaine("table");
```

Rappel sur les littéraux constants :
Nous rappelons aussi que 'a' désigne un unique caractère et "a" désigne une chaîne de caractères. Dans le premier cas nous avons un unique octet dans le second la chaîne de caractère occupe deux octets dans un zone mémoire qui est un tableau de caractères.



7. Allocation dynamique