Les tableaux En C

Qu'est-ce qu'un tableau ?

- Tableau = ensemble
  - de taille fixe
  - de variables du même type
  - adressées par un indice (ou numéro) : leur position dans le tableau

- Exemples :
  - numéros de carte des étudiants d'un groupe de TD → tableau d'entiers
  - notes d'un étudiant → tableau de décimaux
  - Noms des étudiants d'un groupe de TD → tableau de chaines de caractères.

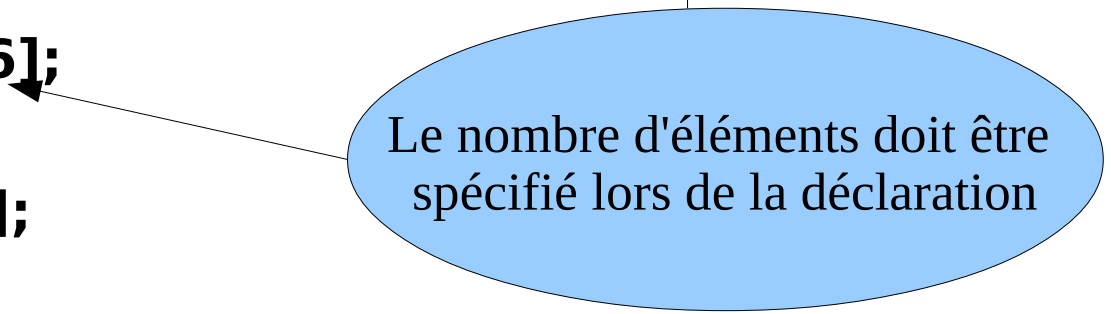
Déclaration d'un tableau

Syntaxe :

<type_tableau> <nom_tableau[<nombre_elements>]>;

Exemple :

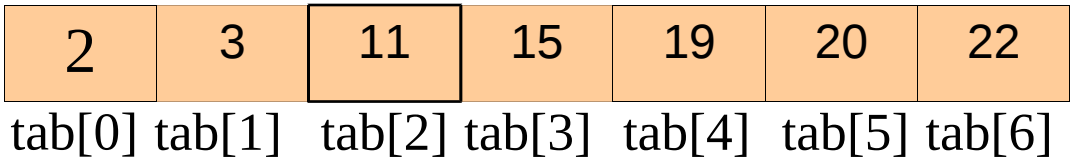
- **char tableau[26];**
- **int table[10];**
- **float vecteur[3];**



Les éléments d'un tableau

Les éléments du tableau sont indicés à partir de **ZERO**

Chacun de ces éléments est désigné par le nom du tableau et son indice



Lors de la déclaration d'une variable de type tableau, la variable contient l'adresse de la première case du tableau (case d'indice 0).

Opérations sur les éléments des tableaux

- Exemples de lecture:
  - Char tab [10] ; tab[0] = 'A';
  - int i = 3; tab[i] = 'D'; tab[i+1] = 'E';

- Exemples d'écriture:
  - int tab[5] ; x = tab[1];
  - tab[3] = tab[2] ; tab[4] = tab[3] + 1 ;

- Initialisation d'un tableau :
  - int t1[] = {1,2,3,4} => tableau de 4 entiers
  - int t2[8] = {11,12,13} => [11,12,13, ?, ?, ?, ?,?]

Initialiser un tableau à la déclaration

- type_elem nom_tab[taille_tab] = {val1, val2, … };
  - La liste contient au maximum taille_tab valeurs ;
  - int t1[3] = {11,12,13}
  - Si elle en contient plus → erreur de compilation

    warning: excess elements in array initializer
  - Si la liste est incomplète, le contenu des cases non initialisées est arbitraire
  - int t1[8] = {11,12,13} => [11,12,13, ?, ?, ?, ?,?]

- type_elem nom_tab[] = {val1, val2, val3};
  - La taille du tableau est donnée par la taille de la liste. Elle n'est pas modifiable
  - int t1[] = {1,2,3,4} => tableau de 4 entiers

Déclaration et initialisation

```
#include <stdio.h>
int main() {
float t_f[3] = {3.1, 2.2};
int t_int[4];
t_int[0] = 1;
printf("t_int[0]= %d\n",t_int[0]);
printf("t_int[1]= %d\n",t_int[1]);
printf("t_f[2]= %d\n", t_f[2]);
return 0
}
```

A la compilation

warning: ‘t_int[1]’ is used uninitialized in this function

Le compilateur ne détecte pas que t_f[2] n'est pas initialisé non plus

Fixer la taille d'un tableau

- Il est interdit d'utiliser une variable pour dimensionner un tableau:
  - `int taille = 3;`
  - `int tab[taille]= {1, 2, 3};`
  - error: variable-sized object may not be initialized

- #define permet d'associer une valeur à un identificateur

On ne peut pas effectuer d'opérations sur l'identificateur

→ la valeur correspondante est constante:

```
#define TAILLE 3
```

…

```
int tab[TAILLE];  → int tab[3] ;
```

Erreurs d'accès aux éléments d'un tableau

- `#define TAILLE 3`
- `int tab[TAILLE];`

- Le compilateur ne fait pas de vérification sur les indices

  `tab[-1] = 1;` → pas d'erreur à la compilation

- Pas de marqueur de fin de tableau

  `tab[3] = 3;` → pas d'erreur à la compilation

- Dans les deux cas, résultat imprévisible à l'exécution

  toujours vérifier que **`0 ? indice ? TAILLE -1`**

Exemple : Saisir les valeurs d'un tableau 1D

```
#include<stdio.h>
#define MAX 10
int main(void) {
    int i, tab[MAX] ;
    For (i=0 ; i < MAX ; i++) {
        printf("donnez la %d ème élément : ",i);
        scanf("%d", &tab[i]);
}
Return 0 ;
}
```

Exemple : copie de tableaux

```
#include<stdio.h>
#define MAX 10
int main(void) {
    int i, T1[MAX], T2[MAX] ;
    for (i=0 ; i < MAX ; i++) {
        printf("donnez la %d ème élément : ",i);
        scanf("%d", &T1[i]);
    }
    for (i=0 ; i < MAX ; i++) {
    T2[i] = T1[i]
    }

return 0 ;
}
```

Recherche du minimum dans un tableau

```
#include<stdio.h>
#define MAX 10
int main(void) {
    int i, pos_min = 0, T1[MAX] ;
    for (i=0 ; i < MAX ; i++) {
        If (T1[i] < T1[pos_min] {
          pos_min = i;
        }
    }
   printf("Le minimum est dans la case %d et
          sa valeur est : %d \n", pos_min,T1[pos_min]);
    }
return 0 ;
}
```

Parcours incomplets de tableaux

- Recherche d'une valeur particulière dans un tableau
  - On continue la recherche tant que :
    - on n'a pas atteint la fin du tableau
  - ET
    - on n'a pas trouvé l'élément cherché
- La condition de continuation porte dans l'ordre
  - sur la taille du tableau
  - sur la valeur de l'élément recherché
- Boucle while

Recherche d'une valeur dans un tableau

```
#include<stdio.h>
#define MAX 3
int main(void) {
    int i, T1[MAX]  = {1,4,8};
     int val = 3
    While ( (i < MAX) && (T1[i] != val) {
          i++ ;
          }
   if (i == MAX)
     printf("élément non trouvé\n");
   else
     printf ("élément trouvé\n") ;
    }
return 0 ;
}
```

Déclaration d'un tableau 2D

Syntaxe :

```
<type> <nom_tab>[<DimLigne>][<DimCol>];
```

Exemple :

- **int tab[3][5]; ==> tableau de 3x5 éléments**

Utile pour représenter les plateaux de jeux :

- **bataille navale, scrabble, sudoku, etc.**

Opérations sur les éléments des tableaux 2D

- Exemples d'écriture:
  - Char tab [10][5] ; tab[0][0] = 'A';
  - int i = 3; int j = 2 ; tab[i][j] = 'D'; tab[i+1][j+1] = 'E';

- Exemples de lecture:
  - int tab[5][5] ; int x = tab[1][2];
  - tab[3][2] = tab[3][1] ;

- Initialisation d'un tableau 2D:
  - int t[2][3] = {{1, 11, 111}, {2,22,222}};
  - C'est un tableau formé de 2 tableaux de 3 éléments.

Accès aux éléments d'un tableau

- Accès à un élément se fait par:

  <NomTableau>[<Ligne>][<Colonne>]

- A TTENTION:

Les indices du tableau varient de 0 à L-1 et de 0 à C-1.

tab[N-1][M-1] est l'élément de la $N^{ème}$ ligne et $M^{ème}$ colonne.

Les éléments du tableau **mat[2][3]** se présentent ainsi:

mat[0][0] mat[0][1] mat[0][2]

mat[1][0] mat[1][1] mat[1][2]

Lecture d'un tableau 2D

```
#include<stdio.h>
#define NB_L 10
#define NB_C 5
int main(void) {
    int i,j tab[NB_L][NB_C] ;
    For (i=0 ; i < NB_L ; i++) {
        for(j=0 ; j< NB_C ; j++)
        scanf("%d", &tab[i][j]);
}
Return 0 ;
}
```

Les chaînes de caractères en C

- **Une chaîne de caractères est une séquence finie de caractères**
  - Pas de type chaîne en C.
  - Convention de représentation

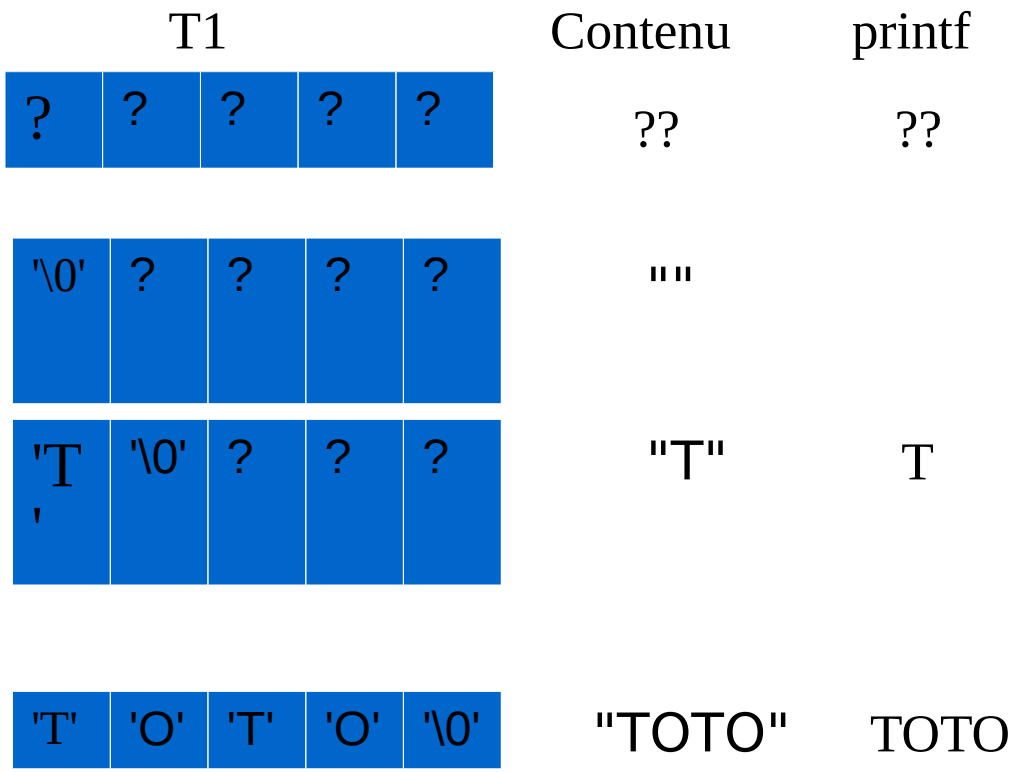
- **Représentation d'1 chaîne**:
  - tableau de caractères terminé par le caractère '\0'
  - char chaine[5]="toto" ==> il faut un tableau de 5 caractères pour contenir une chaîne de 4 caractères
  - char ch[5]={'t','o','t','o'} ;

- Toutes les opérations sur les chaînes de caractères repèrent la fin de ces chaînes par le caractère spécial '\0' (Convention «C»).

Contenant/Contenu

```
#include<stdio.h>
int main(void) {
    int i, char T1[5];

    T1[0] = '\0' ;
     printf("%s",T1);
     T1[0] = 'T' ;T1[1] = '\0' ;
      printf("%s",T1);
     T1[0] = 'T' ;T1[1] = 'O' ;
     T1[2] = 'T' ;T1[3] = 'O' ;
      T1[4] = '\0' ;
      printf("%s",T1);
return 0 ;
}
```



Opérations sur les chaînes de caractères

- Toutes les opérations sur les chaînes de caractères repèrent la fin de ces chaînes par le caractère spécial '\0' (Convention «C»).

- Les prototypes de ces fonctions sont contenues dans les fichiers **<string.h>** et **<stdio.h>**

Lecture des chaînes de caractères

scanf : lecture d'une séquence de caractères ne contenant aucun caractère d'espacement (espace, tabulation, fin de ligne,...). Les espaces initiaux éventuels sont sautés.

gets : lecture d'une ligne complète y compris la fin de ligne

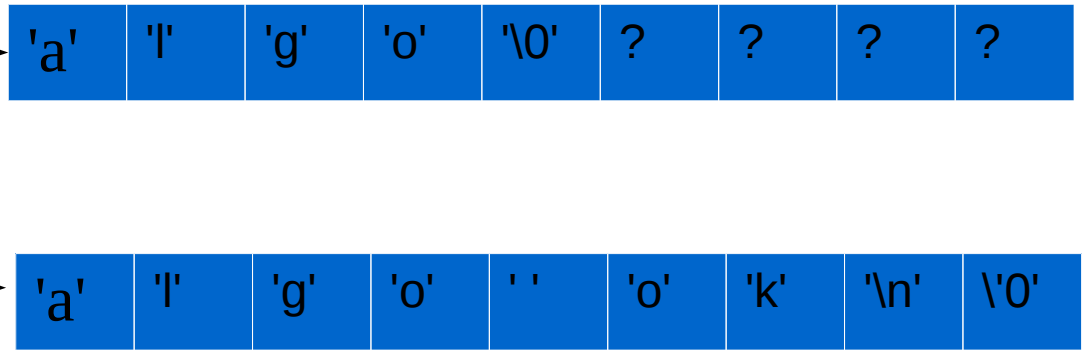
```
#include <stdio.h>
void main() {

char tab[9];

scanf("%s",tab);

gets(tab);

}
```



L'adresse d'un tableau, c'est son nom (*pas de &*)

Affichage d'une chaîne de caractères

```
#include<stdio.h>
int main(void) {
    int i, char T[5] = "toto";

     printf("%s",T);

     puts (T)

return 0 ;
}
```

Affiche toto à l'écran

printf("%s\n",T)

Bibliothèque < string.h >

- Définit des opérations sur les chaînes de caractères :
  - Copie : strcpy
    - ***char *strcpy(char *dest, char *source);***
  - Longueur : strlen
    - ***int strlen(char *S);***
  - Concaténation : strcat
  - Comparaison : strcmp
    - ***char *strcmp(char *S1, char *S2);***