INPT
LA GRANDE ÉCOLE DES TÉLÉCOMMUNICATIONS ET DES TECHNOLOGIES DE L'INFORMATION

Filière Smart-ICT

Algorithmique et Programmation C

Mr N.EL FADDOULI
elfaddouli@emi.ac.ma
nfaddouli@gmail.com

Année Universitaire:2024/2025

1

Plan

3

Le langage C: Les tableaux - Définition

- Un **tableau** est une structure de données qui permet de stocker **plusieurs éléments** de **même type** (*entiers, flottants, caractères, etc.*)
- Ces éléments sont stockés dans une zone mémoire **contiguë**.
- Chaque **élément** est stocké dans **une case** identifiée par un **indice**. Ces indices commencent toujours à **0**.

12	45	-1	17	.....	56	19	3
0	1	2	3	.....			

- Le nombre de cases disponibles dans un tableau est **limité** et appelé la taille **maximale**.
- Un tableau est donc une **variable complexe** constituée de **plusieurs cases** dont chacune contient une seule **valeur** et identifiée par un **indice**.

Le langage C: Les tableaux - Déclaration (1/3)

- Pour déclarer un tableau, on doit spécifier, dans l'ordre, le **type** des éléments qu'il contiendra, le **nom** du tableau et sa **taille** entre crochets. La syntaxe générale est la suivante : ***type nom_du_tableau [taille_maximale]*** ;
- La **taille maximale** peut être une **constante** (*numérique ou symbolique*) ou une **variable** (*Faire attention à sa valeur avant la déclaration du tableau*).
- La taille maximale d'un tableau ne peut pas être modifiée après déclaration.
- **Exemple**:

```
#define max   100
main(){ int T1[50], T2[max], N;
        printf("Donnez le nombre de valeurs à saisir:"); scanf("%d",&N);
        float  T3[N];
        ….
        }
```


Le langage C: Les tableaux – Déclaration (2/3)

- On peut initialiser partiellement ou totalement un tableau:

type nom_du_tableau[*taille_maximale*] = { val_1, val_2, ..., val_n};

*Initialisation des **n** premières cases du tableau*

Exemple:
```
#define  max   100
main(){ int T1[3] = {9, 5, 8};          /* initialisation totale */
        int T2[max] = {2, -4, 10, 8};    /* initialisation partielle */
        ….
        }
```

- Si on n'indique pas la taille maximale lors de l'initialisation, le compilateur la déterminera automatiquement en fonction du nombre des valeurs fournies.

Exemple:
```
main(){ int  T[ ] = {9, 5, 8};   /* Taille = 3 */
        ….
        }
```

- Si on n'indique ni la taille maximale ni l'initialisation, on aura une erreur de compilation.

Le langage C: Les tableaux – Déclaration (3/3)

- La formule ci-après permet d'avoir la taille maximale d'un tableau:

sizeof(*nom_du_tableau*) / sizeof(*type_tableau*)

Taille globale en octet du tableau

Taille d'un élément du tableau

Exemple:
```
#define  max   100
main(){ int T1[max] ,T2[ ] = {2, -4, 5},  N;
        N = sizeof(T1)/sizeof(int);   /* N= 100 */
        N = sizeof(T2)/sizeof(int);   /* N= 3 */
        ….
        }
```

Le langage C: Les tableaux – Accès aux éléments (1/2)

- Pour accéder à un élément pour avoir sa valeur ou pour le modifier, on doit préciser le **nom** du tableau et l'**indice** de l'élément entre crochets: ***nom_du_tableau[indice]***
- L'indice peut être une constante, une variable ou une expression de type entier.

Exemple:
```
main() {   int T[10] , i=2, j, M[20];
           T[0] = 44;  M[1]= T[0]*2;
           scanf("%d",&T[i] ;
           prinft("Le quatrième élément : %d\n", T[i-1]);
           .....
           }
```

- On ne peut pas affecter un tableau à un autre, par exemple l'affectation ***T = M*** est fausse.
- S'il faut copier un tableau dans un autre, il faut faire la copie élément par élément.

Le langage C: Les tableaux – Accès aux éléments (1/2)

- Le compilateur C **ne contrôle pas le débordement d'indices** dans un tableau.

Exemple:
```
main() {   int tableau[3] = {1, 2, 3};
           /* Accès correct à un élément du tableau */
              printf("%d\n", tableau[0]);  /* Affiche 1 */
              printf("%d\n", tableau[2]);  /* Affiche 3 */
           /* Accès hors des limites du tableau */
              printf("%d\n", tableau[5]);  /* Comportement indéfini */
           .......  }
```

- Le programmeur doit s'assurer que les indices utilisés pour accéder aux éléments d'un tableau sont valides et dans les limites spécifiées dans la déclaration du tableau.
- Conséquences d'un débordement d'indices
  - **Accès à des données incorrectes** : on peut lire ou écrire dans une zone de mémoire qui n'appartient pas au tableau, ce qui peut conduire à des résultats imprévisibles.
  - **Crash du programme** : l'accès à une zone de mémoire protégée peut entraîner un plantage du programme (*comme l'erreur de segmentation, souvent appelée "segmentation fault"*).

Le langage C: Les tableaux – Parcours

- Pour lire ou afficher les éléments d'un tableau, il faut les traiter un par un en faisant un parcours du tableau depuis le début jusqu'à la fin.
- On peut utiliser juste une partie du tableau depuis son début en ignorant les autres éléments.
- Le nombre d'éléments utilisés est appelé la **taille effective** càd réellement utilisée.
- **Exemple**: lecture et affichage d'un tableau de N entiers.

```
main() { int T[100], i, N;
         do { printf("Nombre d'entiers:"); scanf("%d",&N); } while (N<=0 || N>100);
    for (i=0; i<N; i++)    /* Lecture du tableau*/
        { printf("Elément %d:",i);   scanf("%d",&T[ i ]) ; }
     /* Affichage du tableau*/
    for (i=0; i<N; i++)  printf("Elément %d: %d \n ", i, T[i]) ;
    }
```

```
int  i, N;
do { printf("Nombre d'entiers:"); scanf("%d",&N); } while (N<=0);
int T[N] ;
```


Le langage C: Les tableaux – Exercices

N.B: On doit d'abord remplir (*lecture*) le tableau à traiter avec le nombre d'éléments indiqué.

1. Déterminer si deux tableaux T1 et T2 de N entiers sont identiques ou pas. Ils sont identiques s'ils contiennent les mêmes éléments dans le même ordre.
2. Inverser l'ordre des éléments d'un tableau de N entiers (***sans tableau intermédiaire***)
3. Soit E un tableau de N entiers.
   Créer et afficher un autre tableau P contenant les entiers pairs de E.

Le langage C: Les chaînes de caractères – Définition

- Une **chaîne de caractères** est un tableau de type **char** qui contient une suite de caractères.
- Contrairement à certains autres langages, il n'existe pas de type natif "***chaîne de caractères***" en C. Les chaînes sont représentées par des tableaux de caractères.
- Le contenu d'une chaîne de caractère est **terminée** par un caractère **nul** (**'\0'**), qui marque sa fin.
- **Exemple**: Une chaîne de caractère dont le contenu est "Rabat"

R	a	b	a	t	\0	....	
0	1	2	3	4	5	.....	

- Il existe plusieurs fonctions dans la bibliothèque standard **<string.h>** pour manipuler les chaînes de caractères.

Le langage C: Les chaînes de caractères – Déclaration & Initialisation

- Une **chaîne de caractères** est déclarée en tant que tableau de type **char**:

char ***nom_de_chaîne*[*taille_max*] = "***texte***";**

Où l'initialisation est optionnelle

- **Exemple**: char nom[20], adresse[30]="Rue Casa N 20";

char ch[ ] ="Bonjour"; */* La taille de **ch** est **automatiquement** définie à **8** (7 lettres + '\0')*/*

⇨ adresse

R	u	e		C	a	s	a		N		2	0	\0	...	
0	1	2	3	4	5	6	7	8	9	10	11	12	14	...	29

N.B: Le caractère nul (**\0**) sera **ajouté automatiquement** à la fin de chaîne lors de l'initialisation.

- Le contenu d'une chaîne déclarée sans initialisation sera indéterminé.
- La taille effective (***longueur***) d'une chaîne est le nombre de caractères qui précèdent **\0**.
- Une chaîne de caractères **vide** commence par **\0**.

Le langage C: Les chaînes de caractères – Accès aux éléments

- On peut accéder à un élément d'une chaîne en utilisant son indice.

nom_de_chaîne[*indice*]

- **Exemple**:

```
char nom[20], adresse[30]="Rue Casa N 20";
nom[0] = 'A';
nom[1] = nom[0] + 1; /* nom[1]= 'B' */
printf("%c \n", nom[1]); /* affichage de B*/
```

Caractère + entier ⇔ Code Ascii + entier

- On ne peut pas faire d'affectation entre chaînes de caractères, par exemple l'affectation *nom=adresse* n'est pas autorisée.
- Eliminer le dernier caractère d'une chaîne consiste à le remplacer par **\0**.

Le langage C: Les chaînes de caractères – Affichage

- On peut afficher une chaîne de caractères en utilisant la fonction **printf** avec le qualificateur de format **%s** comme suit: **printf(".... %s ... ", nom_de_chaîne);**

Exemple:

```
char ch[20] ="Ceci est un exemple";
printf("Votre chaîne est: %s (par défaut) \n", ch);
```

```
Votre cha¯ne est: Ceci est un exemple (par dÚfaut)
```

- On peut aussi utiliser la fonction **puts** de la bibliothèque standard **<stdio.h>** pour afficher une chaîne et faire **automatiquement** un retour à la ligne.

puts(*nom_de_chaîne*);

Exemple:

```
char ch[20] ="Ceci est un exemple";
puts(ch);
```

Le langage C: Les chaînes de caractères – Lecture (1/2)

- On peut lire une chaîne de caractères en utilisant la fonction **scanf** avec le qualificateur de format **%s** comme suit: **scanf("%s", nom_de_chaîne);**

  ⚠ *On n'utilise pas l'opérateur d'adresse* **&**

- **Exemple**:

```
char  ch[20];
printf("Donnez une chaîne:");
scanf("%s", ch);
printf("Votre chaîne est : %s \n", ch);
```

```
Donnez une chaine:mon programme C
Votre chaine est : mon
```

- ⚠ La lecture se termine dès la rencontre d'un caractère **espace**, **tabulation** ou **retour** chariot (entrée).

Le langage C: Les chaînes de caractères – Lecture (2/2)

- Si la chaîne à lire contient des caractères espace et tabulation, on utilise la fonction **gets** de la bibliothèque standard **<stdio.h>**: **gets(nom_de_chaîne);**

Exemple:

```
char  ch[20];
printf("Donnez une chaîne:");
gets( ch);
printf("Votre chaîne est : %s \n", ch);
```

```
Donnez une chaine:mon programme C
Votre chaine est : mon programme C
```

- ⚠ La lecture se termine dès la rencontre d'un **retour** chariot

Le langage C: Les chaînes de caractères – La bibliothèque <string.h> (1/5)

☞ Le langage C propose plusieurs fonctions dans la bibliothèque standard **<string.h>** pour manipuler les chaînes de caractères.

☞ **strlen** : retourne la longueur d'une chaîne (⚠ *sans compter le caractère nul* ***\0***).

strlen(*chaîne_de_caractères*)

Exemple: char ch[20]="Bonjour";
int l = **strlen**(ch); /* l = 7 */

☞ **strcpy** : copie une chaîne source dans une chaîne destination (⚠ *copie aussi* ***\0***) et retourne la chaîne destination *(**pointeur** sur le premier caractère de la chaîne destination)*

strcpy (*chaîne_destination* , *chaîne_source*);

Exemple: char ch[20]="Bonjour", s[30];
strcpy(s, ch) ; /* s = "Bonjour" */
strcpy(s, "Bonsoir") ; /* s = "Bonsoir" */
printf("%s \n", **strcpy**(s, "Bonsoir") ; /* affichage de "Bonsoir" */

Destination: *Une variable*
Source: *variable, constante ou une expression de type chaîne de caractères*

Le langage C: Les chaînes de caractères – La bibliothèque <string.h> (2/5)

☞ **strncpy** : copie la **première partie** de la chaîne source dans une chaîne destination et retourne la chaîne destination *(**pointeur** sur le premier caractère de la chaîne destination)*

strcnpy (*chaîne_destination* , *chaîne_source* , *nbre_de_caractère_à_copier*);

⚠ *Cette fonction* ***n'ajoute pas*** *automatiquement* ***\0*** *dans la chaîne destination. Il faut l'ajouter manuellement par code dans le programme*

Exemple: char ch1[20]="Bonjour", ch2[30]="MMMMMM", ch3[30];
strncpy(ch2, ch1, **3**) ; /* ch2 = "BonMMM" */
ch[**3**] = '**\0**'; /* ch2 = "Bon" */
printf("%s \n", ch2) ;
strncpy(ch3, ch1, **20**) ; /* taille de ch1=7<20 ⇨ ch3= "Bonjour" */

Attention aux débordements : Il faut s'assurer que la destination ait suffisamment d'espace pour contenir les caractères copiés **et** le caractère nul (\0), sinon on risque un bug du programme.

⚠ **strncpy** va au-delà de **\0** dans la chaîne source.

Le langage C: Les chaînes de caractères – La bibliothèque <string.h> (3/5)

☞ **strcmp** : permet de comparer deux chaînes de caractères.

strcmp (*chaîne1*, *chaîne2*)

⚠ *La valeur de retour est:*

- **0** si *chaîne1*=*chaîne2*
- **>0** si *chaîne1*>*chaîne2*
- **<0** si *chaîne1*<*chaîne2*

⚠ *La comparaison n'est pas selon la taille mais selon l'ordre lexicographique des caractères*
(**0,1, 2, 3, 4, 5, 6, 7, 8, 9**, . . . ,**A,B,C, ... ,Z**, . . . ,**a, b, c, ... , z**,…)

⚠ *Elle utilise les codes ascii des caractères pour la comparaison. Il y a donc distinction entre minuscule et majuscule*

Exemple:
```
char  ch1[20]="Un", ch2[30]="Information", ch3[30]="un"; int c;
c= strcmp(ch1, ch2) ;   /* c>0 */
c= strcmp(ch2, ch1) ;   /* c<0*/
c= strcmp(ch1, ch3) ;   /* c<0 */
```

⚠ *Pour ignorer la distinction minuscule/majuscule, on utilise* **strcmpi** *de la même façon*

Le langage C: Les chaînes de caractères – La bibliothèque <string.h> (4/5)

☞ **strcat** : Concatène (ajoute) une chaîne à la fin d'une autre.

strcat(*chaîne1*, *chaîne2*)

⚠ *Elle ajoute automatiquement* **\0** *à la fin de chaîne1*

⚠ *Elle retourne le résultat de la concaténation (pointeur sur le premier caractère de chaîne1)*

Exemple:
```
char  ch1[20]="Un", ch2[30]="jour", ch3[30]="née";
strcat(ch1, ch2) ;   /* ch1="Unjour" */
printf("%s  \n", strcat(ch2, ch3) ) ;   /* ch2= "journée"  et affichage de ch2*/
```

⚠ *Il faut prévoir l'espace pour les caractères ajoutés dans chaîne2, sinon on risque un bug du programme.*

Le langage C: Les chaînes de caractères – La bibliothèque <string.h> (5/5)

- **strlwr** : retourne la conversion d'une chaîne en minuscule.

strlwr(*chaîne*)

- **strupr** : retourne la conversion d'une chaîne en majuscule.

strupr(*chaîne*)

Exemple:

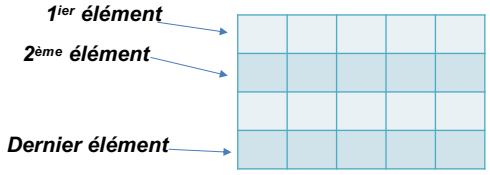
```
char  ch1[30]="Bonjour",  ch2[30];
printf("En majuscule %s\n", strlwr(ch1) ); /* Affichage de "bonjour" */
strcpy(ch2, strupr(ch1) ); /* ch2="BONJOUR" */
```

⚠ *Ces deux fonctions n'existent pas dans toutes les versions système.*

Le langage C: Les tableaux à plusieurs dimensions - Définition

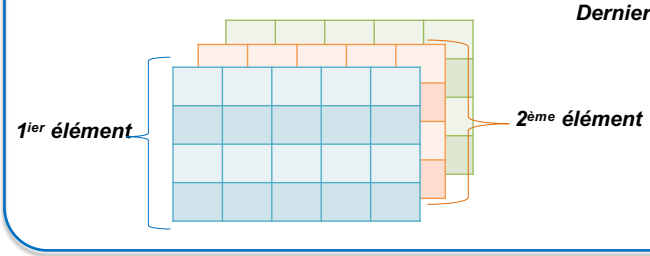
- Un **tableau à plusieurs dimensions** est un tableau qui possède plus d'une dimension. Il peut être vu comme un **tableau de tableaux**.
- Chaque élément d'un tel tableau est lui-même un tableau.
- Il permet permet de représenter des données sous forme de matrices, cubes, ou des structures plus complexes.

Exemple:



Deux dimensions

Trois dimensions



Le langage C: Les tableaux à plusieurs dimensions - Déclaration

☞ On doit préciser le type et le nom du tableau ainsi que la taille maximale entre crochets de chaque dimension selon la syntaxe générale suivante:

Type Nom-Tableau **[*Taille$_1$*] [*Taille$_2$*]...[*Taille$_n$*] ;**

Où ***Taille$_i$*** est la taille maximale de la dimension i

☞ **Exemple:**

Initialisation: int M[3][2] = { {7,9} , {0,6} , {5,4} } ;

```
int  M[3][2];  /* Une matrice d'entiers de 3 lignes et 2 colonnes */
int  C[5][8][3];  /* Un cube de 5 matrices dont chacune est de 8 lignes et 3 colonnes */
```

☞ Chaque cellule élémentaire dans un tableau à **n** dimensions est identifiée par **n** indices.

Exemple:

- Une cellule d'une matrice est identifiée par **deux** indices: *n° de ligne* et *n° de colonne*.
- Une cellule d'un cube est identifiée par **trois** indices: *n° de matrice*, *n° de ligne* et *n° de colonne* dans cette matrice.

Le langage C: Les tableaux à plusieurs dimensions - Parcours

☞ Pour parcourir un tableau à plusieurs dimensions cellule par cellule, on utilise plusieurs boucles imbriquées.

☞ **Exemple:**

```
int  M[3][4], i, j;  /* Une matrice d'entiers de 3 lignes et 2 colonnes */
/* Lecture  de M ligne par ligne */
for (i=0;  i<3;  i++) {
    for (j=0;  j<4;  j++) {
        printf(" Valeur de M[%d][%d]:", i , j);
        scanf("%d",&M[ i ][ j ]);
    }
}
```

Le traitement de la cellule courante d'indices (i, j) peut être n'importe lequel (lecture, affichage, comparaison , ...)

⚠ Pour faire un parcours colonne par colonne, on permute les deux boucles.

Le langage C: Matrice de caractères

- On doit préciser le type et le nom du tableau ainsi que la taille maximale entre crochets de chaque dimension selon la syntaxe générale suivante:

char *Nom_matrice* [*Nb_lignes*] [*Nb_colonnes*];

- **Exemple:**

Initialisation: char M[3][20] = { "Amine" , "Saad' , "Amal" } ;

```
char   M[3][20]; /* Une matrice de 3 lignes (chaînes) et 20 colonnes (taille max=20) */
int  i=1, j, n;
gets(M[ i ]);   /*  lecture de la chaîne (ligne) d'indice i  */
n= strlen(M[ i ]);   /*  n = la longueur de la chaîne d'indice i  */
M[ i ][j]= 'F' ;  /*  Modification du caractère d'indice j dans la chaîne d'indice i  */
```

- **M [i]** : la ligne d'indice **i** est une chaîne de caractère
- **M [i][j]** : le caractère d'indice **j** dans la chaîne d'indice **i**.

Le langage C: Matrice de caractères

- On déclare cette matrice comme un tableau à deux dimensions de type **char**:

char *Nom_matrice* [*Nb_lignes*] [*Nb_colonnes*];

- **Exemple:**

```
char   M[3][20]; /* Une matrice de 3 lignes (chaînes) et 20 colonnes (taille max=20) */
int  i=1, j, n;
gets(M[ i ]);   /*  lecture de la chaîne (ligne) d'indice i  */
n= strlen(M[ i ]);   /*  n = la longueur de la chaîne d'indice i  */
M[ i ][j]= 'F' ;  /*  Modification du caractère d'indice j dans la chaîne d'indice i  */
```

Initialisation: char M[3][20] = { "Amine" , "Saad' , "Amal" } ;

- **M [i]** : la ligne d'indice **i** est une chaîne de caractère
- **M [i][j]** : le caractère d'indice **j** dans la chaîne d'indice **i**.