UNIVERSITÉ ABOU BAKR BEL-KAID–TLEMCEN
FACULÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE
Première année licence



Algorithmique et structure de données 1

Chapitre 5:Les tableaux et les chaines de caractères

Mme HABRI née BENMAHDI Meryem Bochra

Année universitaire: 2024-2025

Plan du cours

1. **Introduction : nécessité de données tabulaires (exemples)**
2. **Tableau**
3. **Tableau en mémoire**
4. **Tableau à une dimension (1D)**
5. **Le tableau à deux dimensions (2D), ou matrice**
6. **Quelques applications des tableaux 1D et 2D**
7. **Le type chaîne de caractères (cas particulier de tableau 1D)**
8. **Quelques applications des chaînes de caractères**

Introduction

Exercice 1: Écrire un programme permettant de saisir les 7 moyennes des matières du premier semestre d'un étudiant inscrit en première année informatique puis afficher avec sa moyenne.

Exercice 1

```
#include <stdio.h>
int main() {
    float note, somme = 0, moyenne;
    int i;
    // Saisie et accumulation des 7 moyennes
    for (i = 1; i <= 7; i++) {
        printf("Entrez la moyenne de la matiere %d : ", i);
        scanf("%f", &note);
        somme += note;
    }
    // Calcul de la moyenne générale
    moyenne = somme / 7;
    // Affichage du résultat
    printf("\nLa moyenne generale de l'etudiant est : %.2f\n",
moyenne);
    return 0;
}
```

Introduction

Exercice 2: Écrire un programme permettant de calculer la moyenne générale de 600 étudiants. Pour cela, le programme doit :

1. **Saisir la moyenne de chaque étudiant (une moyenne comprise entre 0 et 20).**
2. **Vérifier que chaque moyenne saisie est valide. Si la valeur saisie est inférieure à 0 ou supérieure à 20, afficher un message d'erreur et demander à l'utilisateur de saisir à nouveau la moyenne.**
3. **Calculer la somme des moyennes des 600 étudiants.**
4. **Calculer et afficher la moyenne générale des 600 étudiants avec deux chiffres après la virgule.**

Travail demandé : Écrire un programme en langage C utilisant les structures répétitives for et while pour résoudre ce problème.

Exercice 2

```
#include <stdio.h>
int main() {
   float note, somme = 0, moyenne;
   int i;
   printf("=== Calcul de la moyenne generale des 600 etudiants ===\n");
   // Boucle de saisie pour les 600 étudiants
   for (i = 1; i <= 600; i++) {
      printf("Entrez la moyenne de l'etudiant %d : ", i);
      scanf("%f", &note);
      // Contrôle de saisie (note entre 0 et 20)
      while (note < 0 || note > 20) {
         printf("Erreur ! La note doit etre entre 0 et 20.\n");
         printf("Reentrez la moyenne de l'etudiant %d : ", i);
         scanf("%f", &note); }
      somme += note;  // ajout de la note à la somme totale}
// Calcul de la moyenne générale
   moyenne = somme / 600;
   // Affichage du résultat
   printf("\nLa moyenne generale des 600 etudiants est : %.2f\n", moyenne);
   return 0;}
```

Introduction

Exercice 3: Écrire un programme qui permet de saisir les moyennes de tous les 600 étudiants inscrit en première année licence informatique puis de déterminer combien d'entre elles sont supérieures à la moyenne de la promotion.

Solution

```
void main() { int M1,
M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,..... ...
MoyenneGenerale =
(M1+M2+M3+M4+M5+M6+M7+M8+M9+M10+M11+M12+
M13+...)/600;
 printf("La moyenne est : %d", Moyenne); Euuuuuuuuuuuuuhhhhh
!!!! ... }
```

N'y a t-il pas un moyen plus simple et plus élégant pour solutionner ça?

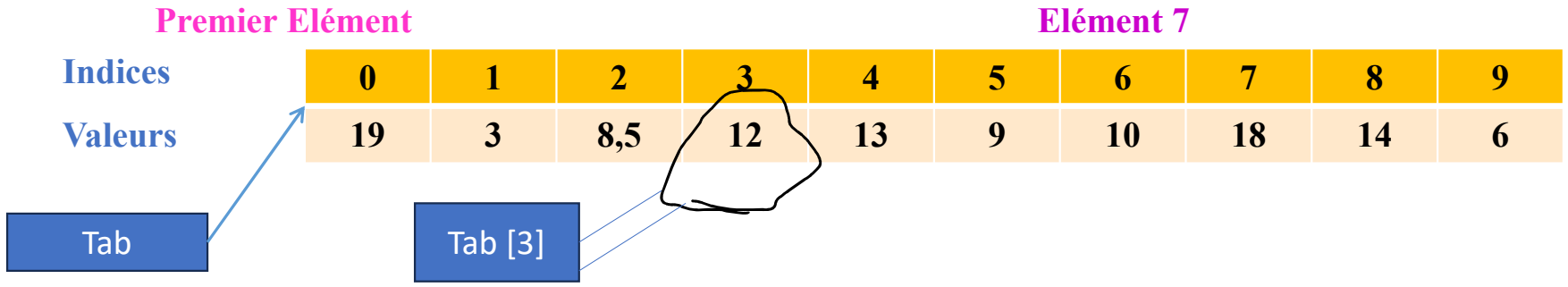
Nécessité des tableaux

- **Les tableaux, c'est la solution idéale.**
- **Lorsqu'un programme doit traiter plusieurs valeurs similaires (notes, températures, etc.)**
- **Il devient difficile d'utiliser uniquement des variables simples.**
- **Les tableaux permettent de regrouper des données du même type sous un seul nom.**

Exemple : stocker les notes de 600 étudiants dans un tableau.

Les tableaux

- **Un ensemble de valeurs portant le même nom de variable et repérées par un nombre, s'appelle un tableau, vecteur, ou encore une variable indicée.**
- **Le nombre qui, au sein d'un tableau, sert à repérer chaque valeur s'appelle l'indice, L'indice représente la position d'un élément dans le tableau**
- **Chaque élément du tableau est désigné par le nom du tableau, suivi de l'indice de l'élément, entre crochets.**



Le tableau en mémoire

- **Un schéma d'illustration d'un tableau de 10 cases en mémoire qui commence à l'adresse 2001.**
- **Lorsqu'un tableau est créé, il prend un espace contigu en mémoire : les cases sont liées les unes à la suite des autres.**
- **Toutes les cases d'un tableau sont du même type. Ainsi, un tableau de int contiendra uniquement des int, et pas autre chose.**

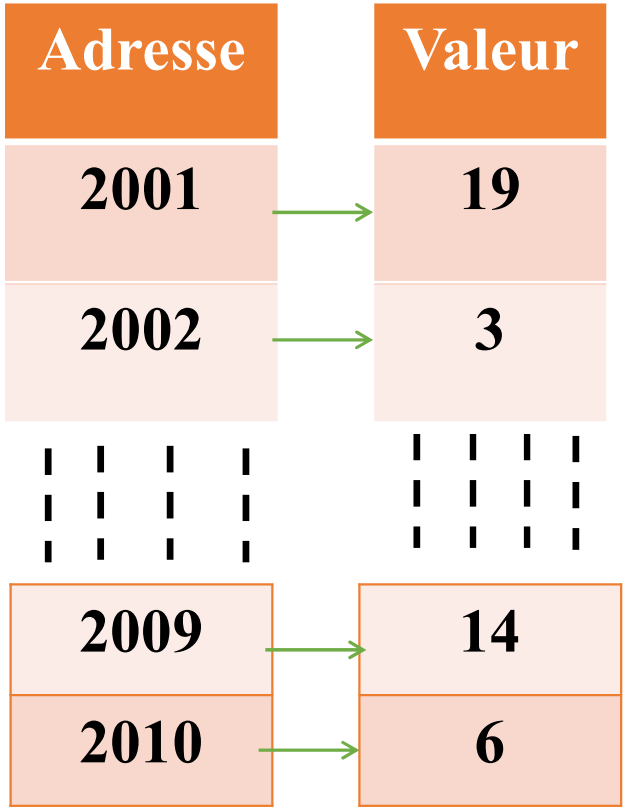


Tableau á une seule dimension (1D)
Déclaration

Définition: **Un tableau 1D est une collection d'éléments de même type.**

Déclaration :

Exemple:

En algorithmique:

```
Var  taille: entier
     varNom: tableau [1…taille] de type
```

En langage C

```
int taille;
Type varNom[taille];
```

En algorithmique:

```
note : tableau [1..10] de réel
note[1]←12,5
```

En langage C

```
float note[10 ];
note[0]=12,5;
```

Il suffit donc de rajouter entre crochets le nombre de cases que vous voulez mettre dans votre tableau. pas de limite.

Tableau á une seule dimension (1D)

Accès aux éléments du tableau

Pour accéder à chaque élément du tableau, il faut écrire le nom du tableau suivi de l'indice de l'élément concerné entre crochet.

En algorithmique:

note[1]←12,5

En langage C

Note[0]=12;

Remarques

- **En programmation C, un tableau commence à l'indice numéro 0! Notre tableau Note de 10 réels a donc les indices 0, 1, 2,... et 9. Il n'y a pas d'indice 10 dans un tableau de 10 cases! C'est une source d'erreurs très courantes pour les débutants.**
- **En algorithmique, le tableau commence à l'indice numéro 1 ! Notre tableau Note de 10 réels a donc les indices 1, 2,... et 10.**

Tableau á une seule dimension (1D)
initialisation d'un tableau

Pour initialiser un tableau dans un algorithme

```
Var  note : tableau [1..5] de réel
        Début
note[1]←12,5
note[2]←10,5
note[3]←19
note[4]←10
note[5]←15,5
        Fin
```

Tableau á une seule dimension (1D)
initialisation d'un tableau

Pour initialiser un tableau en programmation C : il existe deux méthodes pour initialiser un tableau

```
void main()
 { int T[4];
T[0]=2;
T[1]=8;
T[2]=3;
T[3]=1;}
```

```
void main()
 { int T[4] = {2, 8, 3, 1};
 double note[3] = {10, 16, 14.892};
 int tab[6] = {10, 23}; // 10, 23, 0, 0, 0, 0
}
```

- **On peut initialiser tous les éléments d'un tableau par un zéro.**
- **On peut également initialiser que les premiers éléments d'un tableau, tous ceux qui ne sont pas renseignées, seront automatiquement mis à 0.**

Tableau á une seule dimension (1D)

parcourir un tableau

- **Supposons qu'on veuille maintenant afficher les valeurs de chaque case du tableau.**
- **On aurait pu faire autant de printf qu'il y a de cases. Mais ce serait répétitif et lourd, et imaginez un peu la taille de notre code si on devait afficher le contenu de chaque case du tableau une à une!**
- **Le mieux est de se servir d'une boucle qui est très pratique pour parcourir un tableau :**

Exemple

```
#include<stdio.h>
int main()
{double notes[6]={15.5, 13, 4, 19, 17.5, 16};
for (int i = 0 ; i <6; i++)
printf("%f\n", notes[i]);
return 0;}
```

Tableau á une seule dimension (1D)

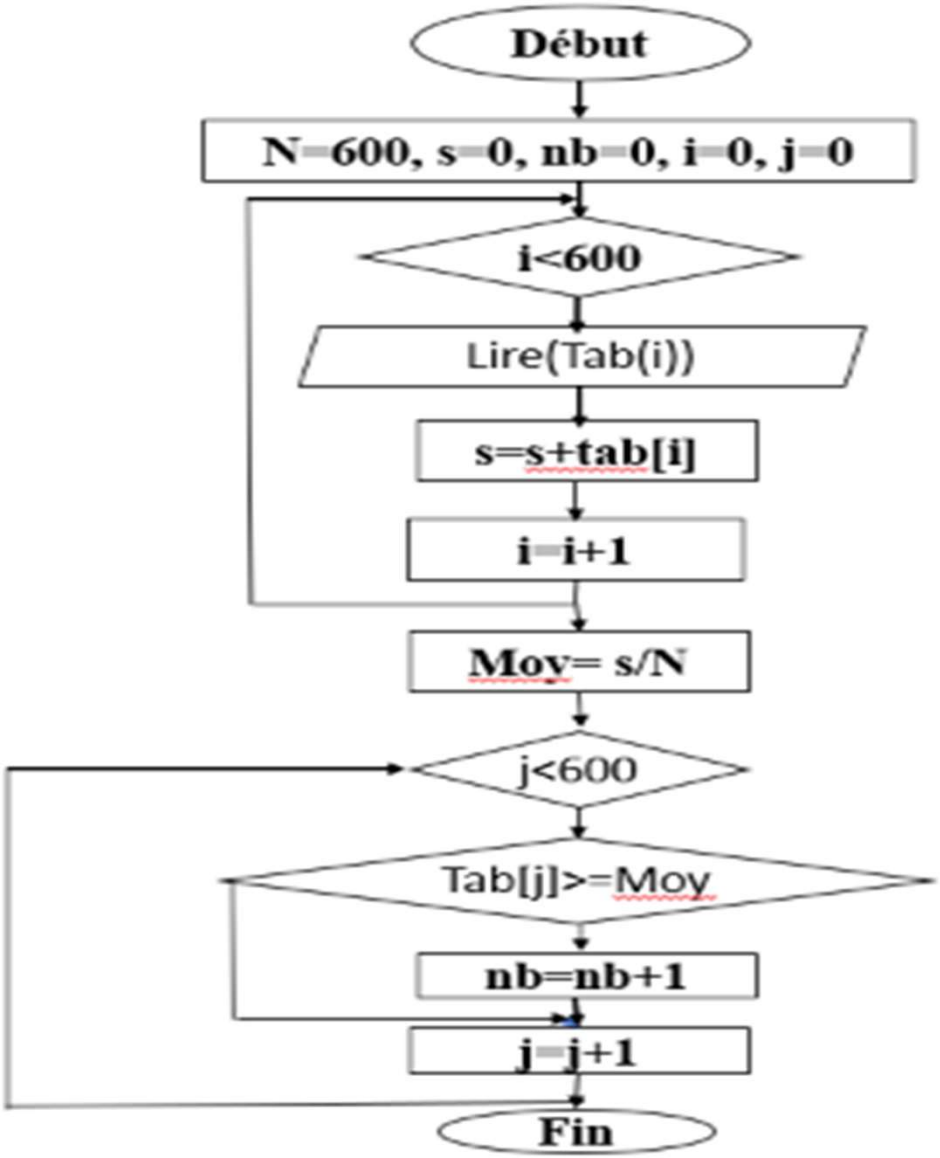
parcourir un tableau

```
#include<stdio.h>
 void main()
 {double notes[6]={15.5, 13, 4, 19, 17.5, 16};
 for (int i = 0 ; i < 8 ; i++)
 printf("%f\n", notes[i]);}
```

- **La boucle parcourt le tableau à l'aide d'une variable appelée i (c'est le nom le plus souvent utilisé pour parcourir un tableau!).**
- **Notez qu'on peut mettre une variable entre crochets pour parcourir le tableau, c'est-à-dire accéder à ses valeurs.**
- **Attention à ne pas tenter d'afficher la valeur de notes[7]!Sinon vous aurez soit n'importe quoi, soit une belle erreur, l'OS interrompra votre programme car il aura tenté d'accéder à une adresse ne lui appartenant pas.**

si on revenait à l'exercice 3?

Exercice 3: Trouver l'organigramme et écrire l'algorithme et le programme qui permet de saisir les moyennes de tous les 600 étudiants inscrit en première année licence informatique puis de déterminer combien d'entre elles sont supérieures ou égale à la moyenne de la promotion.


Début
N=600, s=0, nb=0, i=0, j=0
i<600
Lire(Tab(i))
s=s+tab[i]
i=i+1
Moy= s/N
j<600
Tab[j]>=Moy
nb=nb+1
j=j+1
Fin

```
Algorithme de calcul de moyenne
Var i, s, nb, N: entier
        moy: réel
        Tab : tableau [1..N] de réel
Début
s=0
nb=0
N=600
Pour i←1 jusqu'à N faire
Afficher ("entrez la moyenne du letudiant ", i) ;
Lire(Tab[i])
Finpour
 Pour i←1 jusqu'à N faire
s←s+Tab[i];
Finpour
moy←s/N;
Pour i←1 jusqu'à N faire
    si(Tab[i]>=moy) alors nb←nb+1 finsi
Finpour
 afficher (nb, "etudiants ont plus que la moyenne de la promotion", moy) ;
Fin
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int i, s=0, nb=0, N=600;
double moy ;
double Tab[N] ;
 for (i=0 ; i<N; i++ )
{printf ("entrez la moyenne du letudiant n %d \n", i+1) ;
scanf("%lf", &Tab[i]);}
 for (i=0 ; i<N ; i++) s=s+Tab[i];
moy=s/N;
for(i=0 ; i<N; i++ )
    if(Tab[i]>=moy) nb=nb+1;
 printf ("%d etudiants ont plus que la moyenne de la promotion %lf", nb,moy) ;
return 0; }
```

Exercice d'application

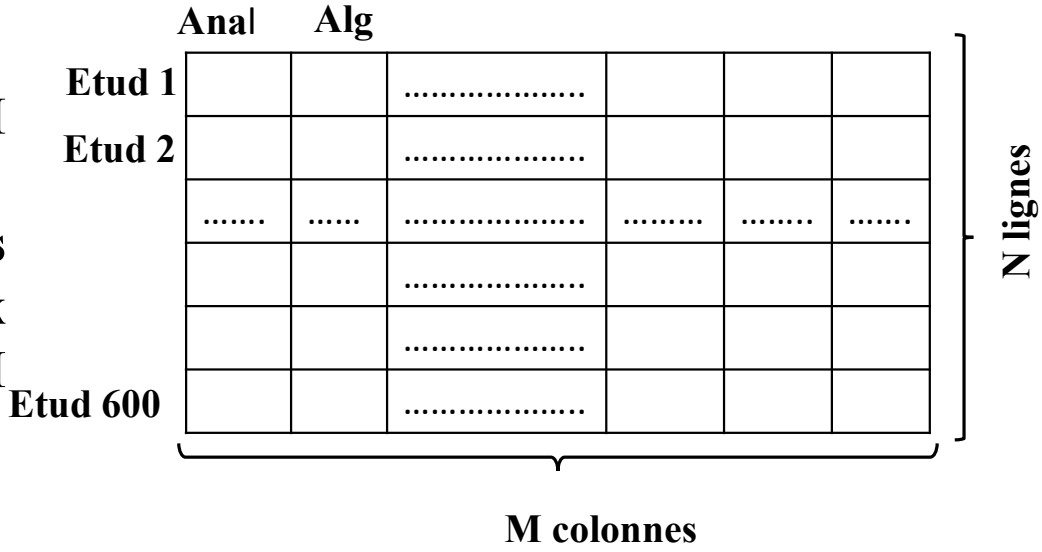
Écrire un programme qui lit un tableau (1D)de nombres réels, puis détermine et affiche le nombre de fois qu'une valeur donnée apparaît dans ce tableau.

```
#include <stdio.h>
int main() {
    int n, i, compteur = 0;
    float valeur, tableau[100];
    printf("Entrez le nombre d'elements du tableau : ");
    scanf("%d", &n);
    printf("Entrez les elements du tableau :\n");
    for (i = 0; i < n; i++) {
        printf("Element %d : ", i + 1);
        scanf("%f", &tableau[i]);}
    printf("Entrez la valeur a rechercher : ");
    scanf("%f", &valeur);
for (i = 0; i < n; i++) {
        if (tableau[i] == valeur)  compteur++;}
printf("La valeur %.2f apparait %d fois dans le tableau.\n", valeur, compteur);
     return 0;
}
```

Tableau á deux dimensions (2D)

- **Un tableau à deux dimensions (ou matrice) est une structure de données qui permet de stocker des éléments sous forme de lignes et de colonnes.**
- **Un tableau à deux dimensions est un tableau de dimension N dont chaque élément est un tableau de dimension M.**

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



En mémoire, les éléments d'un tableau à deux dimensions sont stockés ligne par ligne en continu.

Déclaration d'un tableau (2D) et accès aux éléments d'un tableau (2D)

- **Un tableau 2D (par exemple TAB[L][C]) est une collection de L × C éléments de même type, mais en mémoire, il est stocké d'une seule pièce, comme un grand tableau 1D.**

Déclaration

En algorithmique:

```
Var  tailleL, tailleC: entier
     varNom: tableau [1..tailleL,
1..tailleC] de type
```

En langage C

```
int tailleL,tailleC;
Type varNom[tailleL][tailleC];
```

Exemple

En algorithmique:

```
Note : tableau [1..10, 1..20] de
réel
note[1,1]←12,5
```

En langage C

```
float note[10][20];
Note[0][0]=12,5;
```

Initialisation Tableau (2D)

Pour initialiser un tableau en algorithmique

```
Var  Note : tableau [1..3,1..2] de réel
        Début
note[1,1]←12,5
note[1,2]←10,5
note[2,1]←19
note[2,2]←10
note[3,1]←15,5
note[3,2]←12,5
        Fin
```

Initialisation d'un Tableau (2D)

Pour initialiser un tableau en programmation C : il existe deux méthodes pour initialiser un tableau:

```
void main()
 { int T[2][3];
T[0][0]=2;
T[0][1]=18;
T[0][2]=3;
T[1][0]=1;
T[1][1]=24;
T[1][2]=61;}
```

```
void main()
 { int T[2] [3] ={ {2, 18, 3}, {1,24,61]}
}
```

Nous ne devons pas nécessairement indiquer toutes les valeurs : Les valeurs manquantes seront initialisées par zéro.

Parcourir un tableau (2D)

Donnez le rôle de ce programme?

```
#include <stdio.h>
int main() {
    int A[3][3];
for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("Entrez l'élément A[%d][%d] : ", i, j);
            scanf("%d", &A[i][j]);
     }}
printf("\nContenu du tableau :\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
   return 0;}
```

Ce programme permet de saisir les éléments d'un tableau 2D d'entiers (3 lignes × 3 colonnes), puis d'afficher ces éléments.

Exercice d'application

Écrire un programme en C qui lit les éléments d'un tableau d'entiers 2D

2×3, puis calcule et affiche la somme totale des éléments.

```
#include <stdio.h>
int main() {
    int A[2][3];
    int somme = 0;
for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
            somme += A[i][j];
        }
    }
printf("\nSomme des éléments = %d\n", somme);
 return 0;}
```

Le type caractère

- **Le type caractère en pseudocode et le type char.**
- **Le type char permet de stocker des nombres allant de-128 à 127.**
- **Un caractère est une variable de type Char qui prend 1 octet.**

Exemple pour lire et afficher un caractère

```
Algorithme d'affichage d'un
caractère
var car : caractère
Début
Écrire ("Donner une lettre : ")
   Lire (car)
   Écrire ("Vous avez saisi : ", car)
Fin
```

```
#include <stdio.h>
int main() {
   char car;
   printf("Donner une lettre : ");
   scanf(" %c", &car);
   printf("Vous avez saisi : %c\n", car);

   return 0;
}
```

Le type caractère

- **Comme la mémoire ne peut stocker que des nombres, on a inventé une table qui fait la conversion entre les nombres et les lettres. Ainsi par exemple que le nombre 65 équivaut à la lettre A.**
- **La table ASCII est un tableau de 256 caractères, numérotés de 0 à 255.**
- **La plupart des caractères de base sont codés entre les nombres 0 et 127.**
- **Voila un lien qui englobe les codes ASCII: Table des codes ASCII et leur correspondance**
- **En C, on peut travailler avec un caractère à partir de son numéro dans la table ASCII, il suffit d'écrire cette lettre entre apostrophes, comme ceci : 'A'. 'A' sera donc remplacé par la valeur correspondante : 65.**

Exemple:

```
Int main()
 {char lettre = 'A';
 printf("le code ASCII de %c est de %d\n", lettre, lettre);
 return 0; }
```

Chaine de caractères

Une chaine de caractères n'est rien d'autre qu'un tableau de caractères.

Adresse	Valeur
1000	‘h’
1001	‘e’
1002	‘l’
1003	‘l’
1004	‘o’

Chaine de caractères

Une chaine de caractères doit impérativement contenir un caractère spécial à la fin de la chaîne, appelé ‘caractère de fin de chaîne’. Ce caractère s'écrit '\0'.

Adresse	Valeur
1000	‘h’
1001	‘e’
1002	‘l’
1003	‘l’
1004	‘o’
1005	‘\0’

Chaine de caractères

- **Le caractère '\0' permet tout simplement d'indiquer la fin de la chaîne.**
- **Par conséquent, pour stocker le mot hello (qui comprend 5 lettres) en mémoire, il ne faut pas un tableau de 5 caractères, mais de 6!**
- **Chaque fois que vous créez une chaîne de caractères, vous allez donc devoir penser à prévoir de la place pour le caractère de fin de chaîne. Il faut toujours toujours ajouter un bloc de plus dans le tableau pour stocker ce caractère '\0', c'est impératif!**
- **Oublier le caractère de fin '\0' est une source d'erreurs impitoyable du langage C.**
- **Grâce au caractère '\0' :**
  - **Vous n'aurez pas à retenir la taille de votre tableau car il indique que le tableau s'arrête à cet endroit.**
  - **pour les autres types de tableaux, vous êtes toujours obligés de retenir la taille du tableau quelque part.**

Création et initialisation d'une chaine de caractères

Si on veut initialiser notre tableau chaine avec le texte hello , on peut utiliser la méthode manuelle mais peu efficace :

```
int main() {
char chaine[6]; //ou char chaine[]=" " mais non char chaine[]
chaine[0] = ‘h’;
chaine[1] = ‘e’;
chaine[2] = 'l’;
chaine[3] = ‘l’;
chaine[4] = ‘o’;
chaine[5] = '\0’;
 // Affichage de la chaine grâce au %s du printf
printf("%s", chaine);
return 0;}
```

Pour afficher une chaine de caractère, il faut utiliser le symbole %s (s comme string, qui signifie chaîne en anglais) dans la fonction printf.

Création et initialisation d'une chaine de caractères

- **En tapant entre guillemets la chaîne que vous voulez mettre dans votre tableau, le compilateur C calcule automatiquement la taille nécessaire. C'est-à-dire qu'il compte les lettres et ajoute 1 pour placer le caractère '\0'.**
- **Il écrit ensuite une à une les lettres du mot hello en mémoire et ajoute le '\0' comme on l'a fait nous-mêmes manuellement.**

```
int main() {
 char chaine[] = "hello"; // La taille du tableau chaine est automatiquement calculee
 printf("%s", chaine);
 return 0;}
```

Il y a toutefois un défaut : ça ne marche que pour l'initialisation! Vous ne pouvez pas écrire plus loin dans le code : chaine = "hello";.

Lire une chaine de caractères

- **On peut lire une chaîne entrée par l'utilisateur via un scanf, en utilisant le symbole %s.**
- **Seul problème : on ne sait pas combien de caractères l'utilisateur va entrer.**
- **Il va falloir créer un tableau de char très grand, suffisamment grand!**

```
int main()
{char prenom[100];
printf("Comment t'appelles-tu ? ");
scanf("%s", prenom);
printf("Salut %s, je suis heureux de te rencontrer !", prenom);
return 0;
}
```

- **Si on veut lire nom prénom, c'est-à-dire lire une chaine de caractères avec des espaces, est ce que c'est toujours possible avec cette méthode?**
- **Non, si on va lire HABRI Bochra, la chaine va contenir que HABRI.**

Lire une chaine de caractères avec des espaces

- **On peut lire une chaîne avec des espace entrée par l'utilisateur via un scanf, mais en utilisant le symbole %[^\n] .**

```
int main()
{char NomPrenom[100];
printf(« donnez votre nom prenom? ");
scanf(" %[^\n]", NomPrenom);
printf("Salut %s, je suis heureux de te rencontrer !", NomPrenom);
return 0;
}
```

Bibliothèque de manipulation une chaîne de caractères

- **Une bibliothèque est un ensemble de fonctions qu'on peut appeler dans le programme.**
- **Afin de nous aider un peu à manipuler les chaînes, on nous fournit dans la bibliothèque string.h.**
- **La bibliothèque string.h est un ensemble de fonctions dédiées aux calculs sur des chaînes.**
- **Pensez donc à inclure #include <string.h> en haut des fichiers .c où vous en avez besoin.**
- **Si vous ne le faites pas, l'ordinateur ne connaîtra pas ces fonctions car il n'aura pas les prototypes, et la compilation plantera**

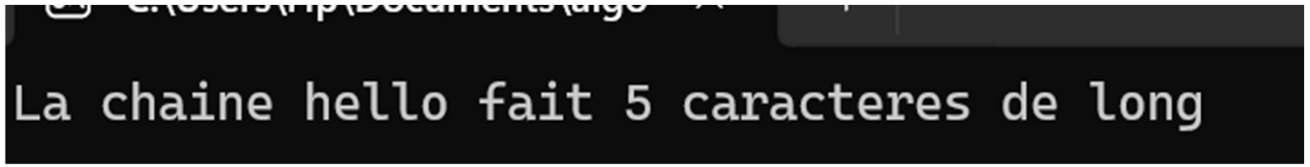
strlen : calculer la longueur d'une chaîne

strlen est une fonction qui calcule la longueur d'une chaîne de caractères (sans compter le caractère '\0').

Exemple

```
int main() {
 char chaine[] = "hello";
 int longueurChaine;
longeurChaine = strlen(chaine); // On affiche la longueur de la chaine
 printf("La chaine %s fait %d caracteres de long", chaine ,longueurChaine);
return 0;}
```

Affichage



strcpy : copier une chaîne dans une autre

La fonction strcpy (comme string copy) permet de copier une chaîne à l'intérieur d'une autre.

Exemple

```
int main()
{ char chaine[] = "Texte", copie[100] = {'c',':',' '};
   printf("copie vaut : %s\n", copie);
  strcpy(copie, chaine); // On copie "chaine" dans "copie"
  printf("chaine vaut : %s\n", chaine);
 printf("copie vaut : %s\n", copie);
return 0;
}
```

Affichage

```
copie vaut : c:
chaine vaut : Texte
copie vaut : Texte
```

strcat : concaténer 2 chaînes

Cette fonction ajoute une chaîne à la suite d'une autre. On appelle cela la concaténation.

Exemple

```
int main() {
char chaine1[100] = " licence ", chaine2[] = "informatique";
 strcat(chaine1, chaine2); // On concatene chaine2 dans chaine1
printf("chaine1 vaut : %s\n", chaine1);
printf("chaine2 vaut toujours : %s\n", chaine2);
 return 0;
}
```

Affichage

```
chaine1 vaut : licence informatique
chaine2 vaut toujours : informatique
```

strcmp : comparer 2 chaînes

La fonction strcmp compare 2 chaînes entre elles.

```
int main() {
char chaine1[] = "informatique", chaine2[] = "informatique";
if (strcmp(chaine1, chaine2) == 0) {
printf("Les chaines sont identiques\n"); }
else
{ printf("Les chaines sont differentes\n"); }
return 0;
 }
```

Il est important de récupérer ce que la fonction renvoie. En effet, strcmp renvoie : 0 si les chaînes sont identiques; une autre valeur (positive ou négative) si les chaînes sont différentes.

Précédence alphabétique des caractères

- **Dans un ordinateur, les caractères ne sont pas rangés comme dans notre alphabet. Ils sont rangés selon un code. La précédence de ces caractères dépendant de ce code.**
- **En général, l'ordinateur met d'abord les chiffres (0–9), puis les lettres majuscules (A–Z), puis les lettres minuscules (a–z).**
- **Les symboles (+, -, /, {, …) et les lettres accentuées (é, è, à, …) sont placés avant, entre ou après, selon le code.**
- **Leur précédence ne correspond à aucune règle d'ordre spécifique. On peut déduire une relation de précédence 'est inférieur à' sur l'ensemble des caractères. Ainsi, on peut dire que '0' est inférieur à 'Z' et noter '0' < 'Z'.**
- **Car le code du caractère '0' (ASCII : 48) est inférieur au code du caractère 'Z' (ASCII : 90).**

Précédence lexicographique des chaînes de caractères

La précédence lexicographique pour les chaînes de caractères suit l'ordre du dictionnaire et est définie de façon récurrente :

- **La chaîne vide "" précède lexicographiquement toutes les autres chaînes.**
- **La chaîne A = "a1 a2 a ... ap " (p caractères) précède lexicographiquement la chaîne B = "b1 b2 b … bm " (m caractères) si l'une des deux conditions suivantes est remplie :**
    1. **'a1' < 'b1'**
    2. **'a1' = 'b1' et "a2a3 ap " précède lexicographiquement "b2b3 bm "**

Exemples :

1. **"ABC" précède "BCD" car 'A'<'B'**
2. **"ABC" précède "B" car 'A'<'B'**
3. **"Abc" précède "abc" car 'A'<'a'**
4. **"ab" précède "abcd" car "" précède "cd«**
5. **" ab" précède "ab" car ' '<'a' le code ASCII de ' ' est 32, et le code ASCII de 'a' est 97**

Exemple de lexicographique des chaînes de caractères

En tenant compte de l'ordre alphabétique des caractères, on peut contrôler le type du caractère (chiffre, majuscule, minuscule).

```
int main()
 {char C=' ';
 C=getchar();
 if (C>='0' && C<='9') printf("Chiffre\n", C);
 if (C>='A' && C<='Z') printf("Majuscule\n", C);
 if (C>='a' && C<='z') printf("Minuscule\n", C);
 // Il est facile, de convertir des lettres majuscules dans des minuscules:
 if (C>='A' && C<='Z') C = C-'A'+'a'; //ou vice-versa:
 if (C>='a' && C<='z') C = C-'a'+'A';
 putchar(C);
 return 0;}
```

Exercice d'application

Écrire un programme qui lit un mot composé uniquement de lettres puis affiche :

1. **sa longueur ;**
2. **le nombre de voyelles ;**
3. **le nombre de consonnes.**

Solution

```
##include <stdio.h>
#include <string.h>
void  main()
{ char mot[100];int i, nbVoyelles = 0, nbConsonnes = 0, longueur;
   printf("Entrez un mot : "); // on suppose que le mot ne contient que les lettres
   scanf("%s", mot);
   longueur = strlen(mot);
   for(i = 0; i < longueur; i++){
  if(mot[i]=='a' || mot[i]=='e' ||mot[i]=='i' || mot[i]=='o' ||mot[i]=='u' || mot[i]=='y' ||
mot[i]=='A' || mot[i]=='E' ||  mot[i]=='I' || mot[i]=='O' || mot[i]=='U' || mot[i]=='Y')
       {nbVoyelles++; }
       else{ nbConsonnes++;    }}
   printf("\nLongueur = %d", longueur);
   printf("\nNombre de voyelles = %d", nbVoyelles);
   printf("\nNombre de consonnes = %d", nbConsonnes); }
```

Exercice d’application

Écrire un programme qui lit un mot puis vérifie qu'il est composé uniquement de lettres alphabétiques (majuscules ou minuscules).

Si le mot contient au moins un caractère non alphabétique, le programme doit afficher un message d'erreur ; sinon, il doit indiquer que le mot est valide.