

Chapitre 2

Les chaînes de caractères

1. Introduction aux chaînes de caractères en C

- Il n'existe pas de type spécial chaîne ou string en C.
- Une chaîne de caractères en C est traitée comme un tableau unidimensionnel de caractères.
- Il existe toutefois plusieurs fonctions spéciales pour le traitement de tableaux de caractères.

char identificateur de variable [dimension du tableau];

Exemples :

```
char Nom[20], Prenom[20];  
char Phrase[300];
```

```
// L'identificateur représente l'adresse  
// du premier caractère de la chaîne.
```


- L'espace mémoire nécessaire pour stocker n caractères est n+1 octets car la chaîne de caractères doit se terminer par le symbole `\0`.
- C'est au programmeur de prévoir cet octet supplémentaire. Le compilateur C n'intervient pas à ce niveau; les répercussions se feront sentir à l'exécution du programme.

Les chaînes de caractères constantes en C

- Elles sont indiquées entre guillemets.

"Ceci est une chaine."

""  une chaîne vide

"Affichage de \"guillemets\" \n"  \" représente le symbole "

{'a', 'e', 'i', '\', 'o', 'u', 'y', '\0'} \' représente le symbole '

printf("Voici le caractere \0 de fin de chaine.");  Voici le caractere

- Plusieurs chaînes de caractères constantes séparées par des espaces, des tabulateurs ou des sauts de lignes dans le texte du programme seront réunies en une seule chaîne constante.

```
printf("Ainsi, il est possible" "\t" "de definir de longues "  
      "chaines de caracteres constantes en utilisant" "\n"  
      "plusieurs lignes dans le texte du programme."  
      );
```

Initialisation de chaînes de caractères en C

- Les tableaux peuvent être initialisés par l'indication de la liste des éléments du tableau entre accolades.

Exemple :

```
char Nom[] = {'L', 'u', 'c', '\0'}; // Le # d'octets nécessaires (i.e. 4) est réservé.
```

- Une approche plus simple consiste à utiliser une chaîne de caractères constante.

Exemple :

```
char Nom[] = "Luc" // Le # d'octets nécessaires (i.e. 4) est réservé.
```

- On peut indiquer explicitement le nombre d'octets à réserver, si celui-ci est supérieur ou égal à la longueur de la chaîne d'initialisation.

Exemple :

```
char Nom[3] = "Luc" // Erreur.  
char Nom[5] = "Luc"
```

L	u	c	\0	0
---	---	---	----	---

Accès aux éléments d'une chaîne de caractères en C

- Idem à l'accès à un élément d'un tableau.

Exemple

```
char A[6] = "Hello";
```

A:	'H'	'e'	'l'	'l'	'o'	'\0'
	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]

Précédence des caractères dans l'alphabet d'une machine

- Elle est dépendante du code de caractères utilisé. Pour le code ASCII, nous avons l'ordre suivant :

. . . , 0, 1, 2, . . . , 9, . . . , A, B, C, . . . , Z, . . . , a, b, c, . . . , z, . . .

```
#include <stdio.h>

void main()
{
    int i;

    printf("L'ensemble des caracteres ASCII\n");

    for (i = 0; i < 256; i++)
    {
        if ((i%32) == 0) printf("\n");
        printf("%c ", (char) i);
    }
    printf("\n");
}
```

Les symboles spéciaux et les lettres accentuées n'ont aucune règle d'ordre spécifique.

Ex. : 'M' < 'm'

Exemples de chaînes de caractères

Exemples :

```
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';
```



Plusieurs librairies de fonctions de C sont disponibles pour le traitement de chaînes de caractères.

La fonction puts de stdio.h

puts(chaîne constante ou le contenu d'une variable)

Exemple : puts(Txt); est équivalent à printf("%s\n", Txt);

```
char TEXTE[] = "Voici une première ligne.";
puts(TEXTE);
puts("Voici une deuxième ligne.");
```

La fonction gets de stdio.h

- **gets** lit une chaîne de caractères incluant l'espace et termine par un retour de fin de ligne.

```
#include <stdio.h>

void main()
{
    char Lieu[25];

    gets(Lieu);
    puts(Lieu);
}
```

- Le retour de fin de ligne est remplacé par le symbole de fin de chaîne \0.

Ex. d'utilisation des fonctions de `stdio.h`

Lire un texte de moins de 200 caractères et remplacer chaque série d'espaces par un seul espace.

```
#include <stdio.h>
void main()
{
    char texte[200];
    int i, j = 0;
    gets(texte);
    for (i = 0; i < 200; i++)
    {
        if(texte[i] == '\0') break;
        if(
            texte[i] != ' ' || (texte[i] == ' ' && texte[i+1] != ' '))
        {
            texte[j] = texte[i];
            j = j+1;
        }
    }
    texte[j] = '\0';
    puts(texte);
}
```

Les fonctions de string.h

Cette librairie fournit une multitude de fonctions pratiques pour le traitement de chaînes de caractères.

Voici une brève description des fonctions les plus utilisées :

strlen(s)	fournit la longueur de la chaîne sans compter \0.
strcpy(s, t)	copie t vers s (\0 inclus); retourne s.
strcat(s, t)	ajoute t à la fin de s; retourne s.
strcmp(s, t)	compare s et t lexicographiquement et fournit un résultat : négatif si s précède t, nul si s est égal à t, positif si s suit t.
strncpy(s, t, n)	copie au plus n caractères de t vers s; retourne s.
strncat(s, t, n)	ajoute au plus n caractères de t à la fin de s et termine s par \0; retourne s.
strchr(s, c)	retourne la sous-chaîne de s débutant par la première occurrence de c dans s.

Légende :

n désigne un nombre de type int, c un caractère.
s et t désignent une chaîne de caractères constante,
le nom d'une variable déclarée comme tableau de type char ou un pointeur vers un type char.

Remarques sur les fonctions de string.h

- Puisque le nom d'une chaîne de caractères représente une adresse fixe en mémoire, on ne peut pas affecter une autre chaîne au nom d'un tableau :

~~A = "Hello";~~

Il faut copier la chaîne caractère par caractère ou bien utiliser la fonction strcpy ou strncpy.

```
strcpy(A, "Hello");
```

- En C, la concaténation de chaînes de caractères ne se fait pas par le symbole + mais avec strcat ou strncat.
- En C, la fonction strcmp est dépendante du code de caractères.

```
#include <stdio.h>
#include <string.h>
void main()
{
    char prenom1[20] = "Luc"; char prenom2[20] = "Marc";
    if (strcmp(prenom1, prenom2) < 0) puts("Luc vient avant Marc.");
}
```

Exemple sur les fonctions de string.h

```
#include <stdio.h>
#include <string.h>

void main()
{
    char Proverbe[75] = {"Rien ne sert de courir, il faut partir a point."};
    char c;

    printf("Extraire la sous-chaine debutant par le caractere ");
    scanf("%c", &c);
    printf("%s\n", strrchr(Proverbe, c));
}
```

Extraire la sous-chaine debutant par le caractere s
sert de courir, il faut partir a point.

Note : `strrchr` est idem à `strchr` sauf que l'on considère la dernière occurrence de `c` au lieu de la première.

Les fonctions de `stdlib.h`

- Ces fonctions permettent la conversion de nombres en chaînes de caractères et vice versa.

chaîne → nombre

`atoi(s)` retourne la valeur numérique de type `int` représentée par `s`.

`atol(s)` retourne la valeur numérique de type `long` représentée par `s`.

`atof(s)` retourne la valeur numérique de type `double` représentée par `s`.

Légende :

`s` désigne une chaîne de caractères constante, le nom d'une variable déclarée comme tableau de type `char` ou un pointeur vers un type `char` (chap. 9).

Note :

- Les espaces au début d'une chaîne sont ignorés.
- Pour une chaîne impossible à convertir, zéro est retourné.
- La conversion s'arrête au premier caractère impossible à convertir.

Les fonctions de stdlib.h - exemple

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    char str[200];
    puts("Entrez un nombre : ");
    gets(str);
    printf("Entree  = %s \n", str);
    printf("int     = %d \n", atoi(str));
    printf("long    = %ld \n", atol(str));
    printf("double  = %f \n", atof(str));
}
```

Les fonctions de ctype.h

- Ces fonctions servent à classer et à convertir des caractères. Les symboles (é, è, â, ...) ne sont pas considérés.

Légende : c représente une valeur de type **int** qui peut être représentée comme caractère.

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')

Les fonctions de ctype.h

- Les fonctions de conversion suivantes fournissent une valeur de 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

Exemple :

```
#include <stdio.h>
#include <ctype.h>

void main()
{
    char str = 'a';
    printf("%c", toupper(str));
}
```

Activité 2

- Ecrire un programme en C qui lit un texte TXT (de moins de 200 caractères) et qui enlève toutes les apparitions du caractère 'e' en tassant les éléments restants. Les modifications se feront dans la même variable TXT

Exemple :

- Entrée:

TXT="Cette ligne contient quelques lettres e."

- Résultat:

TXT="Ctt lign contint qulqus lttrs."

Activité 3

En utilisant les pointeurs et les fonctions sur les chaînes de caractères, écrire un programme C qui :

- Lit deux chaînes de 10 caractères chacune.
- Appelle une fonction **FUSION** qui permet de fusionner les deux chaînes dans une troisième d'une façon alternative : copier un caractère de la première chaîne dans la troisième ensuite un caractère de la deuxième chaîne dans la troisième et ainsi de suite.
- Appelle une fonction **DELETE** qui supprime de la troisième chaîne la répétition successive d'un même caractère. (n'utilisez pas une chaîne intermédiaire).
- Appelle une fonction **SYM** qui retourne vrai si la troisième chaîne est symétrique et faux dans le cas contraire.
- Affiche la troisième chaîne