CHAPITRE 5

LES CHAINES DE CARACTERES EN C

I.INTRODUCTION

Il n'existe pas de type spécial *chaîne* ou *string* en C. Une chaîne de caractères est traitée comme un *tableau à une dimension de caractères* (vecteur de caractères).

II. DECLARATION ET MEMORISATION

II.1.DECLARATION

Déclaration de chaînes de caractères en C

char <NomVariable> [<Longueur>];

Exemples

```
char NOM [20];
char PRENOM [20];
char PHRASE [300];
```

Espace à réserver

Lors de la déclaration, nous devons indiquer l'espace à réserver en mémoire pour le stockage de la chaîne.

La représentation interne d'une chaîne de caractères est terminée par le symbole **'\0'** (NUL). Ainsi, pour un texte de **n** caractères, nous devons prévoir **n+1** octets.

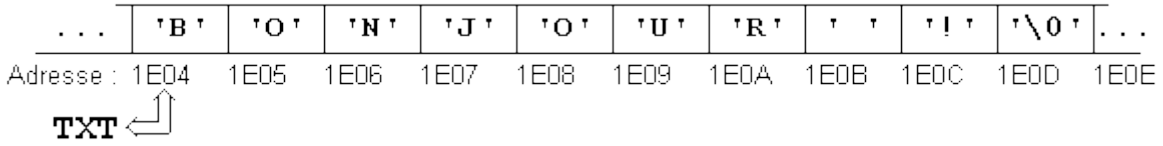
Malheureusement, le compilateur C ne contrôle pas si nous avons réservé un octet pour le symbole de fin de chaîne; l'erreur se fera seulement remarquer lors de l'exécution du programme ...

II.2.MEMORISATION

Le nom d'une chaîne est le représentant de ***l'adresse du premier caractère*** de la chaîne. Pour mémoriser une variable qui doit être capable de contenir un texte de N caractères, nous avons besoin de N+1 octets en mémoire:

Exemple: Mémorisation d'un tableau

```
char TXT[10] = "BONJOUR !";
```



III. LES CHAINES DE CARACTERES CONSTANTES

* Les chaînes de caractères constantes *(string literals)* sont indiquées entre guillemets. La chaîne de caractères vide est alors: **""**

* Dans les chaînes de caractères, nous pouvons utiliser toutes les séquences d'échappement définies comme caractères constants:

```
"Ce \ntexte \nsera réparti sur 3 lignes."
```

* Le symbole " peut être représenté à l'intérieur d'une chaîne par la séquence d'échappement \":

```
"Affichage de \"guillemets\" \n"
```

* Le symbole ' peut être représenté à l'intérieur d'une liste de caractères par la séquence d'échappement \' :

```
{'L','\'','a','s','t','u','c','e','\0'}
```

* Plusieurs chaînes de caractères constantes qui sont séparées par des signes d'espacement (espaces, tabulateurs ou interlignes) dans le texte du programme seront réunies en une seule chaîne constante lors de la compilation:

```
"un "  "deux"
                " trois"
```

sera évalué à **"un deux trois"**

Ainsi il est possible de définir de très longues chaînes de caractères constantes en utilisant plusieurs lignes dans le texte du programme.

Observation

Pour la mémorisation de la chaîne de caractères "Hello", C a besoin de **six (!!)** octets.

'x' est un *caractère constant*, qui a une valeur numérique:

P.ex: 'x' a la valeur 120 dans le code ASCII.

"x" est un *tableau de caractères* qui contient deux caractères:

la lettre 'x' et le caractère NUL: '\0'

'x' est codé dans un octet

"x"
est codé dans deux octets

NB : En langage C il existe des fonction spécifique pour saisir une chaine au clavier ou d'afficher une chaine à l'écran :

La fonction **gets(chaine) ;** permet de saisir une chaine puis le stocke dans la variable **« chaine ».**
La fonction **puts(chaine) ;** permet d'afficher le contenu de la variable **« chaine »** à l'écran .
Pour la saisie et l'affichage nous pouvons utiliser les fonctions scanf et printf avec le formatage %s mais le problème qui se pose est que %s considère l'espace comme fin de chaine et non comme un caractère de la chaine.

IV. INITIALISATION DE CHAINES DE CARACTERES

En général, les tableaux sont initialisés par l'indication de la liste des éléments du tableau entre accolades:

```
char CHAINE[] = {'H','e','l','l','o','\0'};
```

Pour le cas spécial des tableaux de caractères, nous pouvons utiliser une initialisation plus confortable en indiquant simplement une chaîne de caractère constante:

```
char CHAINE[] = "Hello";
```

Lors de l'initialisation par `[]`, l'ordinateur réserve automatiquement le nombre d'octets nécessaires pour la chaîne, c.-à-d.: le nombre de caractères **+ 1** (ici: 6 octets). Nous pouvons aussi indiquer explicitement le nombre d'octets à réserver, si celui-ci est supérieur ou égal à la longueur de la chaîne d'initialisation.

Exemples

```
char TXT[] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'

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

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'

```
char TXT[8] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'	0	0

```
char TXT[5] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	☠

ERREUR pendant l'exécution

```
char TXT[4] = "Hello";
```

ERREUR pendant la compilation

V . LES FONCTIONS DE MANIPULATION DES CHAINES <string.h>

La bibliothèque <*string*> fournit une multitude de fonctions pratiques pour le traitement de chaînes de caractères. Voici une brève description des fonctions les plus fréquemment utilisées.
Dans le tableau suivant, <n> représente un nombre du type **int**. Les symboles <s> et <t> peuvent être remplacés par :
* une chaîne de caractères constante
* le nom d'une variable déclarée comme tableau de **char**
* un pointeur sur **char**

Fonctions pour le traitement de chaînes de caractères

strlen(<s>)	fournit la longueur de la chaîne ***sans*** compter le '\0' final	
strcpy(<s>, <t>)	copie <t> vers <s>	
strcat(<s>, <t>)	ajoute <t> à la fin de <s>	
strcmp(<s>, <t>)	compare <s> et <t> lexicographiquement et fournit un résultat:	Négatif : si <s> précède <t> Zéro : si <s> est égal à <t> Positif : si <s> suit <t>
strncpy(<s>, <t>, <n>)	copie au plus <n> caractères de <t> vers <s>	
strncat(<s>, <t>, <n>)	ajoute au plus <n> caractères de <t> à la fin de <s>	