CHAPITRE 7 : LES CHAÎNES DE CARACTÈRES

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). Il existe quand même des notations particulières et une bonne quantité de fonctions spéciales pour le traitement de tableaux de caractères.

I) Déclaration et mémorisation :

1) Déclaration :

Déclaration de chaînes de caractères en langage algorithmique :

```
chaîne <NomVariable>
```

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 ...

2) Mémorisation :

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 :

II) Les chaînes de caractères 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'}
```

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

III) Initialisation de chaînes de caractères :

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'est-à-dire : 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.

1) Accès aux éléments d'une chaîne :

L'accès à un élément d'une chaîne de caractères peut se faire de la même façon que l'accès à un élément d'un tableau. En déclarant une chaîne par :

```
char Tab[11];
```

nous avons défini un tableau Tab avec 11 éléments, auxquels on peut accéder par :

```
Tab[0], Tab[1], ... , Tab[10]
```

IV) Travailler avec des chaînes de caractères :

1) Lecture de lignes de caractères :

gets est idéal pour lire une ou plusieurs lignes de texte terminées par un retour à la ligne.

Syntaxe : **gets(<Chaîne>)**
Effet : gets lit une ligne de caractères de stdin et la copie à l'adresse indiquée par <Chaîne>.

Le retour à la ligne final est remplacé par le symbole de fin de chaîne '**\0**'.

Exemple :

```
int MAXI = 1000;
char LIGNE[MAXI];
gets(LIGNE);
```

2) Les fonctions de <string> :

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.

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>	

Remarques

- Comme 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 :

Il faut bien copier la chaîne caractère par caractère ou utiliser la fonction **strcpy** respectivement **strncpy** :

strcpy(A, "Hello");

- La concaténation de chaînes de caractères en C ne se fait pas par le symbole '+' comme en langage algorithmique ou en Pascal. Il faut ou bien copier la deuxième chaîne caractère par caractère ou bien utiliser la fonction **strcat** ou **strncat**.
- La fonction **strcmp** est dépendante du code de caractères et peut fournir différents résultats sur différentes machines

3) Les fonctions de <stdlib> :

La bibliothèque *<stdlib>* contient des déclarations de fonctions pour la conversion de nombres en chaînes de caractères et vice-versa.

Conversion de chaînes de caractères en nombres :

atoi(<s>)	retourne la valeur numérique représentée par <s> comme **int**
atol(<s>)	retourne la valeur numérique représentée par <s> comme **long**
atof(<s>) (!)	retourne la valeur numérique représentée par <s> comme **double**

Règles générales pour la conversion :

- Les espaces au début d'une chaîne sont ignorés
- La conversion s'arrête au premier caractère non convertible
- Pour une chaîne non convertible, les fonctions retournent zéro

V) Tableaux de chaînes de caractères :

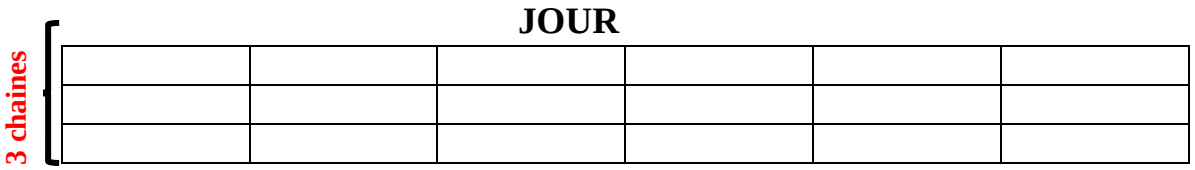
Souvent, il est nécessaire de mémoriser une suite de mots ou de phrases dans des variables. Il est alors pratique de créer un tableau de chaînes de caractères, ce qui allégera les déclarations des variables et simplifiera l'accès aux différents mots (ou phrases).

1) Déclaration, initialisation et mémorisation :

Un tableau de chaînes de caractères correspond à un tableau à deux dimensions du type **char**, où ***chaque ligne contient une chaîne de caractères***.

a) Déclaration

La déclaration **char JOUR[3][6] ;** réserve l'espace en mémoire pour 3 mots contenant 6 caractères.



Lors de la déclaration il est possible d'initialiser toutes les composantes du tableau par des chaînes de caractères constantes :

```
char JOUR[7][9]= {"lundi", "mardi",
                  "mercredi", "jeudi",
                  "vendredi", "samedi",
                  "dimanche"};
```

JOUR

‘l’	‘u’	‘n’	‘d’	‘i’	‘/0’			
‘m’	‘a’	‘r’	‘d’	‘i’	‘/0’			
‘m’	‘e’	‘r’	‘c’	‘r’	‘e’	‘d’	‘i’	‘/0’
‘j’	‘e’	‘u’	‘d’	‘i’	‘/0’			
‘v’	‘e’	‘n’	‘d’	‘r’	‘e’	‘d’	‘i’	‘/0’
‘s’	‘a’	‘m’	‘e’	‘d’	‘i’	‘/0’	‘’	‘’
‘d’	‘i’	‘m’	‘a’	‘n’	‘c’	‘h’	‘e’	‘/0’

7 chaines

9 caractères par chaine

Les tableaux de chaînes sont mémorisés ligne par ligne. La variable JOUR aura donc besoin de 7*9*1 = 63 octets en mémoire.

2) Accès aux différentes composantes :

a) Accès aux chaînes

Il est possible d'accéder aux différentes *chaînes de caractères* d'un tableau, en indiquant simplement la ligne correspondante.

Exemple :

```
L'exécution des trois instructions suivantes :

char JOUR[7][9]= {"lundi", "mardi", "mercredi",
"jeudi", "vendredi",
"samedi", "dimanche"};
```

```
int I = 2;

printf("Aujourd'hui, c'est %s !\n", JOUR[I]);
affichera la phrase :
Aujourd'hui, c'est mercredi !
```

L'attribution d'une chaîne de caractères à une composante d'un tableau de chaînes se fait en général à l'aide de la fonction strcpy :

Exemple :

La commande
strcpy(JOUR[4], "Friday");
changera le contenu de la 5e composante du tableau JOUR de "vendredi" en "Friday".

Accès aux caractères :

Evidemment, il existe toujours la possibilité d'accéder directement aux différents ***caractères*** qui composent les mots du tableau.

Exemple :

L'instruction

```
for(I=0; I<7; I++)
     printf("%c   ", JOUR[I][0]);
```

va afficher les premières lettres des jours de la semaine :

```
l   m   m   j   v   s   d
```