Une chaîne de caractères est traitée comme un *tableau à une dimension de caractères* (vecteur de caractères).

1. Déclaration et mémorisation

char <NomVariable> [<Longueur>];

Exemples

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

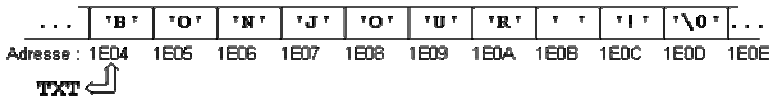
Lors de la déclaration, nous devons indiquer l'espace à réserver en mémoire pour le stockage de la chaîne.

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.

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 !";



2. 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'}

* 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"

3. Initialisation de chaînes de caractères

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

Pour le cas spécial des tableaux de caractères, nous pouvons écrire :

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 XT [5] = “Hello” ; ➔ Erreur à l’exécution

Char TXT [4] = “Hello” ; ➔ Erreur à la compilation

Exercice

Lesquelles des chaînes suivantes sont initialisées correctement ?

a) char a[] = "un\ndeux\ntrois\n";
b) char b[12] = "un deux trois";
c) char c[] = 'abcdefg';
d) char d[10] = 'x';
e) char e[5] = "cinq";
f) char f[] = "Cette " "phrase" "est coupée";
g) char g[2] = {'a', '\0'};
h) char h[4] = {'a', 'b', 'c'};
i) char i[4] = "'o'";

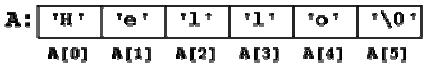
4. Accès aux éléments d'une chaîne

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 A[6];**

nous avons défini un tableau A avec six éléments, auxquels on peut accéder par: **A[0], A[1], ... , A[5]**

Exemple

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



5. Manipulation sur les chaînes de caractères

Les bibliothèques de fonctions de C contiennent une série de fonctions spéciales pour le traitement de chaînes de caractères.

5.1. Les fonctions de <stdio.h>

- Affichage de chaînes de caractères

Exemples

```
char NOM[] = "hello, world";
printf(":%s:", NOM);          -> :hello, world:
```

puts est idéale pour écrire une chaîne constante ou le contenu d'une variable dans une ligne isolée.

Syntaxe: **puts(<Chaîne>)**

Effet: **puts** écrit la chaîne de caractères désignée par <Chaîne> sur *stdout* et provoque un retour à la ligne. En pratique,
puts(TXT); est équivalent à **printf("%s\n",TXT);**

Exemples

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

- Lecture de chaînes de caractères

scanf avec le spécificateur **%s** permet de lire un mot isolé à l'intérieur d'une suite de données du même ou d'un autre type.

Effet: **scanf** avec le spécificateur **%s** lit un ***mot*** du fichier d'entrée standard *stdin* et le mémorise à l'adresse qui est associée à **%s**.

Exemple


```
char LIEU[25];
int JOUR, MOIS, ANNEE;
printf("Entrez lieu et date de naissance : \n");
scanf("%s %d %d %d", LIEU, &JOUR, &MOIS, &ANNEE);
```

gets est idéal pour lire une ou plusieurs lignes de texte (p.ex. des phrases) terminées par un retour à la ligne.

Syntaxe: **gets(<Chaîne>)**

Effet: **gets** lit une ***ligne*** de 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);
```

Exercice

Ecrire un programme qui lit 5 mots, séparés par des espaces et qui les affiche ensuite dans une ligne, mais dans l'ordre inverse. Les mots sont mémorisés dans 5 variables M1, ... ,M5.

Exemple

```
voici une petite phrase !
! phrase petite une voici
```

Exercice

Ecrire un programme qui lit une ligne de texte (ne dépassant pas 200 caractères) la mémorise dans une variable TXT et affiche ensuite:

a) la longueur L de la chaîne.

b) le nombre de '**e**' contenus dans le texte.

c) toute la phrase à rebours, sans changer le contenu de la variable TXT.

d) toute la phrase à rebours, après avoir inversé l'ordre des caractères dans TXT:

```
voici une petite phrase !
! esarhp etitep enu iciov
```

5.2. Les fonctions de <string>

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>

Exercice

Ecrire un programme qui demande l'introduction du nom et du prénom de l'utilisateur et qui affiche alors la longueur totale du nom sans compter les espaces. Employer la fonction **strlen**.

Exercice

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2, les compare lexicographiquement et affiche le résultat:

Exemple:

Introduisez la première chaîne: ABC
Introduisez la deuxième chaîne: abc
"ABC" précède "abc"

Exercice

Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 et qui copie la première moitié de CH1 et la première moitié de CH2 dans une troisième chaîne CH3. Afficher le résultat.

a) Utiliser les fonctions spéciales de *<string>*.

b) Utiliser uniquement les fonctions **gets** et **puts**.

Exercice

Ecrire un programme qui lit un verbe régulier en "er" au clavier et qui en affiche la conjugaison au présent de l'indicatif de ce verbe. Contrôlez s'il s'agit bien d'un verbe en "er" avant de conjuguer. Utiliser les fonctions **gets**, **puts**, **strcat** et **strlen**.

Exemple:

Verbe : fêter
je fête
tu fêtes
il fête
nous fêtons
vous fêtez
ils fêtent

5.3. 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
- Il n'y a pas de contrôle du domaine de la cible
- La conversion s'arrête au premier caractère non convertible
- Pour une chaîne non convertible, les fonctions retournent zéro

Conversion de nombres en chaînes de caractères

itoa (<n_int>, <s>, <b>)

convertit son premier argument en une chaîne de caractères qui sera ensuite attribuée à <s>. La conversion se fait dans la base <b>.

Remarque

Il existe la possibilité d'employer la fonction **sprintf** pour copier des données formatées ***dans une variable*** de la même façon que **printf** les imprime à l'écran.

Syntaxe:

sprintf(<chaîne cible>, <chaîne de formatage>, <expr1>, <expr2>, . . .)

5.4. Les fonctions de <ctype>

Les fonctions de ***classification*** suivantes fournissent un résultat du type **int** différent de zéro, si la condition respective est remplie, sinon zéro.

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**)
isspace(<c>)	si <c> est un signe d'espacement
	(' ', **'\t'**, **'\n'**, **'\r'**, **'\f**)

Les fonctions de ***conversion*** suivantes fournissent une valeur du type **int** qui peut être représentée comme caractère; la valeur originale de <c> reste inchangée:

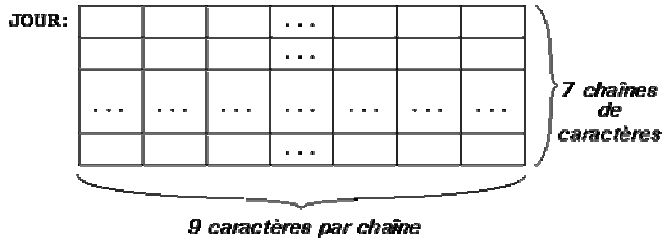
tolower(<c>)	retourne <c> converti en minuscule si <c> est une majuscule
toupper(<c>)	retourne <c> converti en majuscule si <c> est une minuscule

6. Tableaux de chaînes de caractères

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

La déclaration **char JOUR[7][9];**

réserve l'espace en mémoire pour 7 mots contenant 9 caractères (dont 8 caractères significatifs).



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'
	...	...	...	...	...	...	...	...	...
	'd'	'i'	'm'	'a'	'n'	'c'	'h'	'e'	'\0'

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

```
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'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