

Chapitre VII

LES CHAINES DE CARACTERES

Objectifs

Général

⇒ Assimiler le concept de chaîne de caractères dans la programmation C.

Spécifiques

- Connaitre la notion de chaîne de caractère dans le langage C ;
 - Savoir manipuler les chaînes de caractères ;
 - Comprendre la notion de tableau de chaînes de caractères.
-

1. Déclaration et mémorisation

▪ Déclaration :

```
char <NomVariable> [<Longueur>];
```

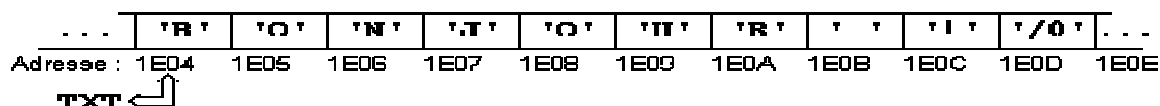
Exemple : déclaration

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

▪ Mémorisation :

Exemple: Mémorisation d'un tableau

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



2. Les chaînes de caractères constantes :

Les chaînes de caractères constantes sont indiquées entre guillemets. Dans les chaînes de caractères, nous pouvons utiliser toutes les séquences d'échappement définies comme caractères constants:

```
TXT1[] = "Ce \ntexte \nsera réparti sur 3 lignes."
```

```
TXT2[] = "Affichage de \"guillemets\" \n"
```

```
TXT3[] = {'L', '\\', 'a', 's', 't', 'u', 'c', 'e', '\\0'}
```

```
TXT4[] = "un " "deux " "trois "
```

3. Initialisation de chaînes de caractères :

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

4. Accès aux éléments d'une chaîne :

Exemple :

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

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

5. Les bibliothèques de fonctions :

5.1. Les fonctions <stdio.h>

- Affichage de chaînes de caractères : *printf()*, *puts()*.

```
char TEXTE[] = "Voici une première ligne";
printf(":%s:", TEXTE); → : Voici une première ligne:
puts(TEXTE); → Voici une première ligne
puts("Voici une deuxième ligne"); → Voici une deuxième ligne
```
- Lecture de chaînes de caractères : *scanf()*, *gets()*

Exemple 1 :

```
char LIEU[25];
printf("Entrez le lieu de naissance : \n");
scanf("%s %d %d %d", LIEU);
```

Exemple 2 :

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

5.2. Autre fonctions (Voir TP) : <string.h>, <stdlib.h> ...

6. Tableaux de chaînes de caractères :

6.1. 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*.

Exemple :

```
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 $7 \times 9 \times 1 = 63$ octets en mémoire.

6.2. Accès aux différentes composantes :

▪ Accès aux chaînes :

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

```
int I = 2;

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

▪ Affectation :

```
strcpy(JOUR[4], "Friday");
```

▪ Accès aux caractères :

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

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

7. Exercices d'application :

Exercice 6.1

Lesquelles des chaînes suivantes sont initialisées correctement ? Corrigez les déclarations fausses et indiquez pour chaque chaîne de caractères le nombre d'octets qui sera réservé en mémoire.

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

Exercice 6.2

Ecrire un programme qui lit un nombre entre 1 et 7 et qui affiche le nom du jour de la semaine correspondant : *"lundi" pour 1, "mardi" pour 2 ... "dimanche" pour 7*

Utiliser le premier élément du tableau pour mémoriser un petit message d'erreur.

Solution

```

int N; /* nombre entré */
char JOUR[8][9] = {"\aErreur!", "lundi", "mardi", "mercredi",
                  "jeudi", "vendredi", "samedi", "dimanche"};
/* Saisie du nombre */
printf("Entrez un nombre entre 1 et 7 : ");
scanf("%d", &N);
/* Affichage du résultat - pour perfectionnistes */
if (N>0 && N<8)
    printf("Le %de%c jour de la semaine est %s.\n",
           N, (N==1)?'r':' ', JOUR[N]);
else
    puts(JOUR[0]);

```

Exercice 6.3

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

Solution

```

char M1[30], M2[30], M3[30], M4[30], M5[30];
printf("Entrez 5 mots, séparés par des espaces :\n");
scanf ("%s %s %s %s %s", M1, M2, M3, M4, M5);
printf("%s %s %s %s %s\n", M5, M4, M3, M2, M1);

```

Exercice 6.4

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 !
! esarphtetip enu iciov*

Solution

```

/* Déclarations */
char TXT[201]; /* chaîne donnée */
int I, J; /* indices courants */
int L; /* longueur de la chaîne */
int C; /* compteur des lettres 'e' */
int AIDE; /* pour l'échange des caractères */

/* Saisie des données */
printf("Entrez une ligne de texte (max.200 caractères) :\n");
gets(TXT); /* L'utilisation de scanf est impossible pour */
/* lire une phrase contenant un nombre variable de mots. */

/* a) Compter les caractères */

```

```

/* La marque de fin de chaîne '\0' est */
/* utilisée comme condition d'arrêt. */
for (L=0; TXT[L]; L++)
    ;
printf("Le texte est composé de %d caractères.\n",L);

/* b) Compter les lettres 'e' dans le texte */
C=0;
for (I=0; TXT[I]; I++)
    if (TXT[I]=='e') C++;
printf("Le texte contient %d lettres 'e'.\n",C);

/* c) Afficher la phrase à l'envers */
for (I=L-1; I>=0; I--)
    putchar(TXT[I]); /* ou printf("%c",TXT[I]); */
putchar('\n');      /* ou printf("\n"); */

/* d) Inverser l'ordre des caractères */
for (I=0,J=L-1 ; I<J ; I++,J--)
{
    AIDE=TXT[I];
    TXT[I]=TXT[J];
    TXT[J]=AIDE;
}
puts(TXT); /* ou printf("%s\n",TXT); */

```

Exercice 6.5

Ecrire un programme 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:

*Cette ligne contient quelques lettres e.
Ctt lign contint qulqus ltrrs .*

Solution

```

/* Déclarations */
char TXT[201]; /* chaîne donnée */
int I,J; /* indices courants */

/* Saisie des données */
printf("Entrez une ligne de texte (max.200 caractères) :\n");
gets(TXT);
/* Eliminer les lettres 'e' et comprimer : */
/* Copier les caractères de I vers J et incrémenter J */
/* seulement pour les caractères différents de 'e'. */
for (J=0,I=0 ; TXT[I] ; I++)
{
    TXT[J] = TXT[I];
    if (TXT[I] != 'e') J++;
}
/* Terminer la chaîne !! */
TXT[J]='\0';
/* Edition du résultat */
puts(TXT);

```

Exercice 6.6

Ecrire un programme qui lit une chaîne de caractères CH et qui convertit toutes les majuscules dans des minuscules et vice-versa. Le résultat sera mémorisé dans la même variable CH et affiché après la conversion.

Solution

```
/* Déclarations */
char CH[100]; /* chaîne à convertir */
int I;        /* indice courant */

/* Saisie de la chaîne */
printf("Entrez la chaîne à convertir : ");
gets(CH);
/* Conversion de la chaîne */
for (I=0; CH[I]; I++)
{
    if (CH[I]>='A' && CH[I]<='Z')
        CH[I] = CH[I] - 'A' + 'a';
    else if (CH[I]>='a' && CH[I]<='z')
        CH[I] = CH[I] - 'a' + 'A';
}
/* Affichage de la chaîne convertie */
printf("Chaîne convertie : %s\n", CH);
```
