10. Les chaînes de caractères

Chaînes de caractères(1)

- Il n'y a pas de type chaîne de caractères prédéfini en C. Par convention, les chaînes de caractères sont représentées à l'aide de **tableaux de caractères** à un indice. Malgré cela, la manipulation des chaînes de caractères est très souple et très efficace en C.
- Le principe est le suivant, dans un tableau suffisamment grand, on stocke à la suite les différents caractères de la chaîne. A la fin de la chaîne, on rajoute un caractère supplémentaire qui fait office de **marqueur** de fin de chaîne. C'est le caractère dont le code ASCII est zéro, que l'on note '**\0**'. Attention toutefois à ne pas le confondre avec le caractère '**0**' dont le code ASCII est 48!

Chaînes de caractères(2)

- Les caractères stockés dans la suite du tableau seront tout simplement ignorés de toutes les fonctions, la convention voulant que le caractère ’\0’ soit le dernier de la chaîne.
- De plus, si une erreur entraîne l’absence de ce marqueur, les fonctions de manipulation de chaînes de caractères fonctionneront anormalement, puisqu’elles ne le trouveront pas.
- L’inconvénient de cette technique est qu’il faut toujours se souvenir qu’un tableau de N positions ne peut contenir au maximum qu’une chaîne de **longueur** N-1, le dernier élément du tableau devant forcément contenir le marqueur de fin de chaîne.

Chaînes de caractères(3)

- **La déclaration d'une chaîne de caractère se fait par l'une des méthodes suivantes :**

char NomChaine [Longueur];

OU

char * NomChaine ;

- **Exemples:**
  - char string[10]; /* définit une chaîne de 9 caractères */
  - char nom[20] = "ahmed"; /* déclaration avec initialisation */
  - char chaine1 [] = { 't', 'o', 't' , 'o' , '\0'} ; /* déclaration avec initialisation */
  - char chaine2 [20] = {'t', 'o', 't' , 'o' , '\0'} ; /* déclaration avec initialisation */
  - char * CH1 ; /* déclaration sans initialisation */
  - char * CH2= " toto" ; /* déclaration avec initialisation */
  - CH1 = "salut tou" /* affectation permise*/
  - nom = "chaine"; /*affectation non permise
  - char PHRASE [300]; /* déclaration sans initialisation */

Chaînes de caractères(4)

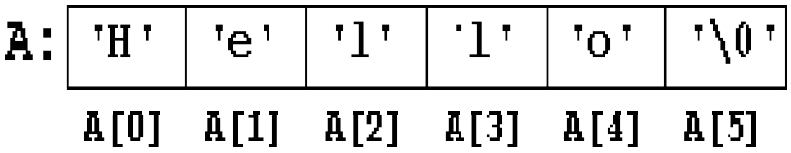
- **L'accès à un élément d'une chaîne de caractères** peut se faire de la même façon que l'accès à un élément d'un tableau. En déclarant une chaîne par:

  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";
```



Chaînes de caractères(5)

Remarques

- Lorsque le compilateur rencontre une chaîne de caractères constante, il la convertit automatiquement en tableau et rajoute le caractère nul ('\0') à la fin.
- Tout comme les tableaux, il n'y a pas d'opérations globales sur les chaînes (**affectation**, **comparaison**, **concaténation**) dans le langage, il y a par contre de nombreuses fonctions qui assurent ces tâches.
- La seule exception est pour l'initialisation, une chaîne peut être initialisée par une chaîne constante lors de sa définition comme on peut le voir dans les exemples ci-dessus. Par ailleurs, le compilateur peut calculer la longueur d'une chaîne facilement. On peut donc omettre la longueur de la chaîne entre les crochets si on veut qu'elle soit ajustée automatiquement.
- Les caractères d'interligne que l'on trouve dans le dernier exemple occupent deux caractères dans le texte du programme : \ et n. Dans le programme exécutable et en mémoire centrale, ils n'occupent par contre plus qu'un seul octet.

Lecture/Ecriture de chaînes(1)

LECTURE

scanf()

- Cette fonction permet de lire une chaîne de caractères en spécifiant un %s dans la chaîne de format.
- Son mécanisme de découpage ne permet toutefois pas de taper des chaînes de caractères contenant des blancs ou des tabulations. La lecture se termine aussitôt qu'un caractère séparateur est rencontré.
- Par ailleurs, il ne faut pas utiliser l'opérateur adresse & lors de la lecture d'une chaîne de caractères, les tableaux étant de toute façon passés par référence.

Lecture/Ecriture de chaînes(2)

- On préfère généralement la fonction

gets(chaine)

- Cette fonction lit une chaîne depuis le flux d'entrée standard stdin et la place dans le tableau de caractères passé en paramètre.
- La lecture se termine à la réception d'un caractère d'interligne (touche return). Le '\n' n'est par inséré dans la chaîne, il est remplacé par un terminateur '\0'.
- Contrairement à scanf(), elle permet l'entrée de chaînes contenant des espaces et des tabulations.

Lecture/Ecriture de chaînes(3)

ECRITURE

printf()

- Cette fonction permet d'afficher une chaîne de caractères en spécifiant un %s dans la chaîne de format.
- Elle ne rajoute pas d'interligne toute seule, mais on peut en mettre un dans la chaîne de format si on le désire.

puts(chaine)

- Cette fonction envoie la chaîne de caractères spécifiée dans le flux de sortie standard stdout. Elle ajoute un caractère d'interligne à la fin.

Lecture/Ecriture de chaînes(4)

- **Exemples**:

```
char msg [6] = "hello";
char ch[5] ;
printf ("%s",msg) ;        /* affiche hello sans retour à la ligne*/
printf ("%.4s", msg) ;     /* affiche hell   */
scanf ("%s", ch) ;         /* ch contient l'adresse du premier caractère
                            de la chaine */
puts ( msg) ;              /* affiche hello avec un retour à la ligne */
puts (" bonjour") ;        /* affiche bonjour avec un retour à la ligne */
gets(ch) ;     /* lit une ligne de caractères de stdin et la copie à
                  l'adresse indiquée par ch */
```

Traitement de chaînes de caractères(1)

- La bibliothèque <***string.h***> fournit une multitude de fonctions pratiques pour le traitement de chaînes de caractères. Voici une brève description des fonctions les plus fréquemment utilisées.

  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>

Traitement de chaînes de caractères(2)

- ***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:

  ~~A="Hello"~~

  ➔

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

  strcpy(A, "Hello");

Traitement de chaînes de caractères(3)

- 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**.
- **Exemple:**

```
#include <string.h>
main ()
{
char ch1[50]=’’bonjour’’ ;
     char *ch2=’’ monsieur”;
     printf (“avant  : %s\n “, ch1);
     strcat (ch1, ch2);
     printf (“après : %s\n’’,ch 1);
     strncat (ch1,ch2,2);
     printf (“après : %s\n’’,ch 1);
}
```

Résultat

avant : bonjour
après : bonjour monsieur
après : bonjour monsieur m

Les fonctions de conversion(1)

La bibliothèque <***stdlib.h***> 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

Les fonctions de conversion(2)

- **Exemples**:

```
#include <stdio. h>
#include <stdlib. h>
main()
{
        char ch[40] ;
        int n;
        do
        {
            printf(''donnez une chaîne :'') ;
            gets(ch);
            printf(''int : %d\n'', atoi(ch));
            printf("double : %e\n" ,atof(ch));
        } while(n);
}
```

Résultat

donnez une chaîne : 123
int : 123
double : 1.230000e+02

Les fonctions de conversion(3)

- **Conversion de nombres en chaînes de caractères**

  Le standard ANSI-C ne contient pas de fonctions pour convertir des nombres en chaînes de caractères. Si on se limite aux systèmes fonctionnant sous DOS, on peut quand même utiliser les fonctions **itoa**, **ltoa** et **ultoa** qui convertissent des entiers en chaînes de caractères.

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

  ltoa (<n_long>, <s>, <b>)

  ultoa (<n_uns_long>, <s>, <b>)

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

Exemple:

```
#include <stdlib. h>
main()
{
char ch[40];
int n,b;
do
{
    printf(''Donner un nombre et une
base :'') ;
    scanf(''%d  %d", &n, &b);
    printf("%s\n", itoa(n, ch, b));
}while(n) ;
}
```

Fonctions de classification et de conversion(1)

- Les fonctions de <***ctype.h***> servent à classifier et à convertir des caractères. Les symboles nationaux (é, è, ä, ü, ß, ç, ...) ne sont pas considérés. Les fonctions de *<ctype>* sont indépendantes du code de caractères de la machine et favorisent la portabilité des programmes.
- Les fonctions de ***classification*** suivantes fournissent un résultat du type **int** différent de zéro, si la condition respective est remplie, sinon zéro. <c> représente une valeur du type **int** qui peut être représentée comme caractère.

  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>)**

Fonctions de classification et de conversion(2)

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

Les fonctions de recherche dans une chaîne(1)

- On trouve des fonctions dans **string.h** de recherche de “l’occurrence’’ dans une chaîne, d’un caractère ou d’une autre chaîne (nommée alors sous-chaîne). Ces fonctions fournissent comme résultat :
  - **l’adresse** de l’information cherchée en cas de succès
  - le pointeur **null** dans le cas contraire.
- **strchr (char *chaine, char caractere)**

  Recherche, dans *chaîne* la première position où apparaît le caractère mentionné
- **strrchr (char *chaine, char caractere)**

  Réalise le même traitement que *strchr,* mais en explorant la chaîne mentionné à partir de la fin. Elle fournit donc la dernière occurrence du caractère mentionné.
- **strstr (char *chaine, char *sous-chaine)**

  Recherche dans *chaîne* la première occurrence complète de la sous-chaîne mentionnée (respectant MIN et MAJ).
- **strpbrk (char *chaine 1, char *chaine2)**

  Recherche, dans *chaîne1* la première occurrence d’un caractère quelconque de *chaine2.*

Les fonctions de recherche dans une chaîne(1)

```
#define c ’e’
#define sch  "re"
#define voy  "aeiou"
main()
{
char  mot[40]; char *adr ; printf(’’donnez un mot : "); gets (mot);
if (adr=strchr(mot,c))
      printf(’’Première occurrence de%c en %s\n et pos = %d \n’’,c,adr, adr-mot);
if (adr=strrchr(mot,c))
      printf(’’Dernière occurrence de %c en %s\n et pos = %d \n’’,c, adr, adr-mot) ;
if (adr = strstr(mot,sch))
      printf (“Première occurrence de %s en %s\n et pos = %d \n’’,sch,adr, adr-mot) ;
if (adr =strpbrk(mot,voy))
      printf (“Première occurrence de l’une des lettres de %s en %s\n et pos = %d \n’’,voy,adr,adr-
      mot) ;
}
```

Tableau de chaînes de caractères(1)

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

  La déclaration **char JOUR[7][9];** réserve l'espace en mémoire pour 7 mots contenant 9 caractères (dont 8 caractères significatifs).

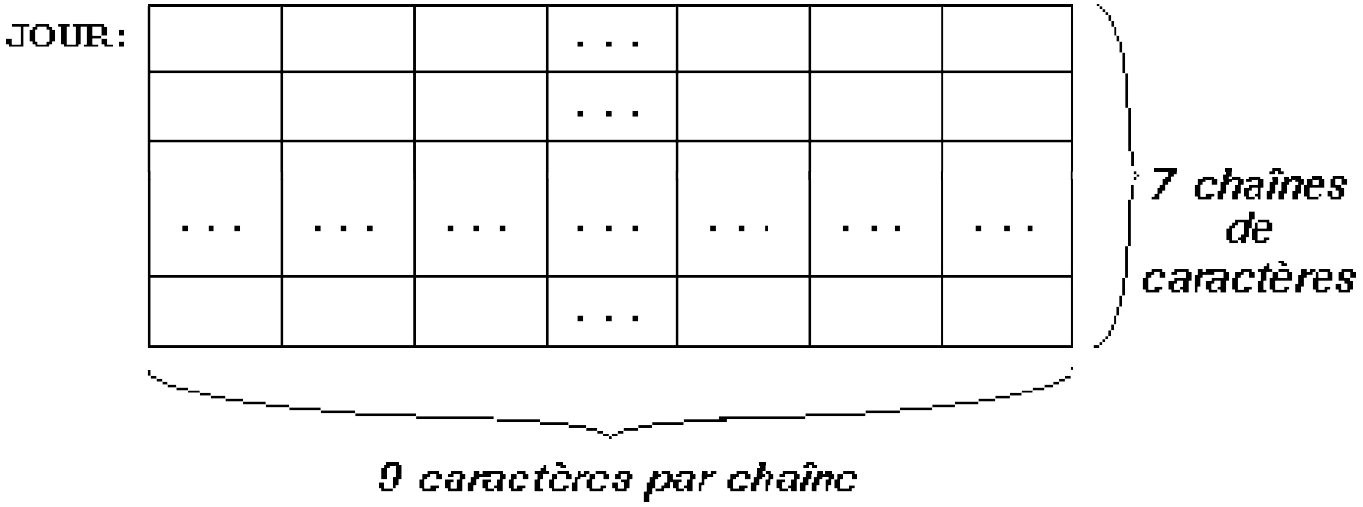


Tableau de chaînes de caractères(2)

- ***Initialisation***

  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'

- ***Mémorisation***

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

Tableau de chaînes de caractères(3)

- ***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]);
```

  affiche la phrase:

  Aujourd'hui, c'est mercredi !