Chaînes de caractères

Caractères et code ASCII

Un caractère (type `char`) est représenté par son code ASCII (initiales du pompeux "American Standard Coding for Information Interchange"), qui est un entier codé sur un octet. Les caractères imprimables sont numérotés de 32 (caractère blanc) à 127 ; pour les pays latins, le code ASCII a été étendu au-delà de 128, sous le nom de code Iso-latin.

Voici un programme qui imprime la table des caractères numérotés de 32 à 127 :

```
void main (void) {
  int i, j;
  char c;

  for (i = 32; i < 48; i++) {
    for (j = 0; j < 6; j++) {
      c = i + 16 * j;
      printf ("%3d %c   ", c, c);
    }
    printf ("\n");
  }
}
```

et voici le résultat de son exécution :

```
32       48 0    64 @    80 P    96 `   112 p
33 !     49 1    65 A    81 Q    97 a   113 q
34 "     50 2    66 B    82 R    98 b   114 r
35 #     51 3    67 C    83 S    99 c   115 s
36 $     52 4    68 D    84 T   100 d   116 t
37 %     53 5    69 E    85 U   101 e   117 u
38 &     54 6    70 F    86 V   102 f   118 v
39 '     55 7    71 G    87 W   103 g   119 w
40 (     56 8    72 H    88 X   104 h   120 x
41 )     57 9    73 I    89 Y   105 i   121 y
42 *     58 :    74 J    90 Z   106 j   122 z
43 +     59 ;    75 K    91 [   107 k   123 {
44 ,     60 <    76 L    92 \   108 l   124 |
45 -     61 =    77 M    93 ]   109 m   125 }
46 .     62 >    78 N    94 ^   110 n   126 ~
47 /     63 ?    79 O    95 _   111 o   127
```

Un programmeur n'est pas tenu (heureusement) de connaître cette table par coeur : un caractère noté entre apostrophes désigne son propre code, et donc 'A'=65, '0'=48, etc. Bien noter que le code ASCII du chiffre 0 n'est pas du tout nul.

On peut faire des calculs arithmétiques sur les caractères ; voici quelques exemples d'utilisation courante :

- Si *n* est un entier compris entre 0 et 9, le code du chiffre correspondant s'écrit `n+'0'` (autrement dit *n*+48).

- Si *n* est un entier compris entre 10 et 15, le code du chiffre hexadécimal (minuscule) correspondant s'écrit `n+'a'-10` (autrement dit *n*+87).

- Pour transformer un caractère majuscule *c* en une minuscule, il suffit de lui ajouter `'a'-'A'`, soit 32.

Les caractères de code ASCII inférieur à 32 sont des caractères spéciaux, pour la plupart tombés en désuétude, à l'exception principalement du caractère de *fin de ligne*, noté '\n' en C.

Chaînes de caractères

Une *chaîne de caractères* (*string* chez les anglo-saxons) est un tableau de caractères, avec une convention spéciale :

Une chaîne de caractères se termine par le caractère nul.

Cette convention est possible car le code ASCII prévoit explicitement qu'aucun caractère n'est numéroté 0, donc on ne risque pas de confondre cette marque de fin de chaîne (qu'on appelle aussi *sentinelle*) avec un caractère de la chaîne. On peut ainsi calculer la *taille* d'une chaîne à partir de son *nom*, ce qui est impossible en général pour les tableaux.

Il n'existe aucun mécanisme magique qui assure le respect de cette convention, laissé à la responsabilité du programmeur, selon la «philosophie C». Si par suite d'une erreur de programmation, une chaîne n'est pas terminée par le caractère nul, le comportement du programme devient imprévisible, et les fonctions standard manipulent alors des *pointeurs fous*, avec toutes les conséquences catastrophiques habituelles.

Une chaîne constante s'écrit entre doubles apostrophes (ou guillemets anglais) ; par exemple

```
"Erreur numéro %d\n"
```

est une chaîne de caractères ; le compilateur lui alloue une région de la mémoire, et donc une adresse. L'exécution de l'instruction :

```
printf ("Erreur numéro %d\n", n);
```

comporte essentiellement les étapes suivantes :

1. l'*adresse* de la chaîne est copiée comme premier argument de *printf* ;
2. cette chaîne est analysée, jusqu'au caractère nul qui la termine, et le motif `%d` est reconnu comme indiquant un format décimal (et donc un argument supplémentaire) ;
3. la chaîne définitive à afficher est construite en remplaçant `%d` par la représentation décimale de *n*, faite de caractères compris entre '0' et '9'.

Noter la différence essentielle, dans un programme, entre le caractère 'A' (qui est l'entier 65) et la chaîne "A" (qui est l'adresse d'un tableau de deux octets, contenant le caractère 'A' suivi du caractère nul). Noter aussi que la *chaîne vide* "" est tout à fait légale : c'est l'adresse d'un tableau constitué d'un seul octet qui vaut zéro ; toutes les fonctions (bien écrites) qui manipulent des chaînes de caractères fournissent des résultats cohérents pour la chaîne vide.

Fonctions standard

Il existe une bibliothèque de fonctions pour manipuler les chaînes de caractères ; leurs déclarations sont regroupées dans le fichier `<string.h>`. La première est nommée *strcpy* et effectue la copie d'une chaîne dans une autre ; il est intéressant d'étudier comment on peut écrire une telle fonction, bien qu'elle soit disponible de façon standard. On peut utiliser un indice :

```
void strcpy (char but[], char source[]) {
  int i;

  for (i = 0; source[i]; i++)
    but[i] = source[i];
}
```

Le test surprenant `source[i]`, dans la boucle `for`, est équivalent à `source[i] != 0` : on effectue la copie tant que la fin de la chaîne `source` n'est pas atteinte.

On peut aussi utiliser des pointeurs (autrement dit des adresses) :

```
void strcpy (char *but, char *source) {
  for ( ; *source; source++, but++)
    *but = *source;
}
```

La boucle `for` ne comporte pas d'initialisation, d'où son aspect déséquilibré, et la plupart des programmeurs C utiliseront plutôt la possibilité de mélanger les opérateurs de déréférencement (*) et d'incrémentation (++) :

```
void strcpy (char *but, char *source) {
  while (*source)
    *but++ = *source++;
}
```

Cette dernière variante a un «charme C» certain, mais le choix entre les différentes versions n'a rien d'essentiel, et dépend de l'humeur du programmeur.

Dans le cadre du module I20, aucune virtuosité dans la manipulation des pointeurs n'est demandée aux étudiant(e)s ; l'essentiel, et de loin, est de comprendre les algorithmes de base pour manipuler les tableaux, et de savoir les traduire clairement en C.

Les principales fonctions standard pour manipuler les chaînes de caractères sont, outre *strcpy* :

- *strcat* : `strcat (s1, s2)` concatène les deux chaînes, c'est-à-dire copie *s2* à la suite de *s1*.
- *strlen* fournit la longueur d'une chaîne (sans compter le zéro final).
- *strcmp* compare deux chaînes dans l'ordre lexicographique, et retourne -1, 0 ou 1 selon que la première est inférieure, égale ou supérieure à la seconde.