TP 4 – Manipulation de chaînes de caractères

1 Objectifs du TP

Ce TP a pour but de découvrir comment :

1. copier des chaînes ;
2. comparer des chaînes ;
3. faire des recherches dans des chaînes ;
4. convertir des chaînes en valeurs numériques,

2 Généralités sur les chaînes

En C, les chaînes de caractères sont représentées par un tableau de caractères terminé par un caractères nul, '\0'. Il faut donc toujours réserver un élément de tableau de plus que de caractères à mettre au maximum dans la chaîne.

Les signatures des sous-programmes de manipulation de chaînes de caractères sont dans le fichier `<string.h>`.

La fonction `strlen` appliquée à une chaîne retourne un entier contenant le nombre de caractères effectifs de la chaîne, c'est à dire le nombre d'octets entre le début de la chaîne et le caractère nul non compris. Attention, cette valeur est différente de la taille (`sizeof`) de la chaîne qui correspond à la taille maximale du tableau définie à sa déclaration.

Créer un nouveau répertoire pour y mettre les fichiers de ce TP. Dans ce répertoire, créer un fichier *copies.c* contenant le code suivant :

```
1.      #include <stdio.h>
1.      #include <string.h>
2.
3.      #define LG_MAX 15
4.
5.      void main() {
6.            char ch[LG_MAX+1] = "Hello";
7.            printf("longueur=%d, taille=%d\n", strlen(ch), sizeof(ch));
8.      }
```

Q. 1 Exécuter le programme et expliquer le résultat.

3 Copies de chaînes

Pour copier une chaîne dans une autre, on peut utiliser la fonction `strcpy` dont le premier argument est la chaîne destination et le second la chaîne source. Ainsi, le code suivant copie la chaîne src dans la chaîne dst :

```
1.      strcpy(dst, src) ;
```

⚠ Attention, il faut absolument avoir : `strlen(src) < sizeof(dst)`. Si ce n'est pas le cas, la chaîne destination va déborder et le résultat est imprévisible !!!

Q. 2 Modifier le programme précédent pour déclarer une autre chaîne `ch2`, copier `ch` dans `ch2` et afficher `ch2`.

Une autre fonction, `strncpy` est plus sure, car elle permet de limiter le nombre de caractères copiés. En effet, elle a un troisième paramètre qui est un entier qui indique le nombre maximal de caractères à copier. *Cependant, si ce nombre maximal est atteint avant de rencontrer le caractère nul, celui-ci n'est pas ajouté à la chaîne destination !!!* Celle-ci ne sera donc pas correctement terminée.

> Modifier le programme précédent pour déclarer une autre chaîne ch3 initialisée avec 15 étoiles, copier les 2 premiers caractères de ch dans ch3 et afficher ch3. Expliquer ce qui se passe.

Une solution pour régler ce problème consiste à ajouter un caractère nul après le dernier caractère copié.

Ajouter à la fin de votre fonction *main* le code suivant :

```
1.          int n, lg;
2.          strcpy(ch3, "***************");
3.          printf("entrer le nombre de caractères à copier : ");
4.          scanf("%d", &n);
5.          lg = (n > LG_MAX) ? LG_MAX : n;
6.          strncpy(ch3, "abcdefghijklmnopqrstuvwxyz", lg);
7.          ch3[lg] = '\0';
8.          printf("%s\n", ch3);
```

L'expression `(condition) ? valeur1 : valeur2` est égale à `valeur1` si la `condition` est vraie et à `valeur2` sinon. La ligne 4 du code précédent affecte donc à `lg` le minimum de `n` et `LG_MAX`. On pourrait remplacer `LG_MAX` par `sizeof(ch3)-1`.

Q. 3 Exécuter le programme avec plusieurs valeurs de n, inférieure à 15, égale à 15, supérieure à 15 et supérieure à 26. Vérifier que le comportement est correct.

Il est aussi possible de concaténer deux chaînes avec les fonctions `strcat` et `strncat`. La première, `strcat`, a pour premier argument la chaîne destination et pour second argument la chaîne source qui est donc ajoutée à la fin de la chaîne destination.

```
1.       strcat(dst, src) ;
```

⚠ Attention, il faut s'assurer que `strlen(dst) + strlen(src) < sizeof(dst)`. Si ce n'est pas le cas, la chaîne destination va déborder et le résultat est imprévisible !!!

Q. 4 Ajouter à la fin de votre programme la concaténation du mot `World` à la chaîne `ch` et l'affichage du résultat.

La fonction `strncat` a un troisième argument entier qui limite le nombre de caractères ajoutés à la fin de la chaîne destination. Contrairement à `strncpy`, la chaîne destination reçoit toujours un caractère nul après le dernier caractère de la chaîne source ajouté.

On peut donc utiliser `strncat` à la place de `strncpy` pour faire une copie sûre. Il suffit de « vider » la chaîne `dst` avant de faire la concaténation :

```
1.          dst[0] = '\0';
2.          strncat(dst, src, n);
```

⚠ Les 4 fonctions définies dans cette section, `strcpy`, `strncpy`, `strcat` et `strncat` retournent un pointeur sur la chaîne destination.

4 Comparaisons de chaînes

On ne peut pas comparer directement des chaînes car ça reviendrait à comparer leurs adresses car un nom de tableau représente l'adresse de son premier élément.

Créer un nouveau fichier *comparaisons.c* contenant le code suivant :

```
1. #include <stdio.h>
2. #include <string.h>
3.
4. void main() {
5.
6.          char ch1[] = "abcd";
7.          char ch2[] = "abcd";
8.          char ch3[] = "xyz";
9.
10.             if (ch1 == ch2) {
11.                     printf("%s = %s\n", ch1, ch2);
12.             } else {
13.                     if (ch1 < ch2) {
14.                             printf("%s < %s\n", ch1, ch2);
15.                     } else {
16.                             printf("%s > %s\n", ch1, ch2);
17.                     }
18.             }
19.
```

```
20.             if (ch2 == ch3) {
21.                     printf("%s = %s\n", ch2, ch3);
22.             } else {
23.                     if (ch2 < ch3) {
24.                             printf("%s < %s\n", ch2, ch3);
25.                     } else {
26.                             printf("%s > %s\n", ch2, ch3);
27.                     }
28.             }
29.
30.             if (ch3 == ch1) {
31.                     printf("%s = %s\n", ch3, ch1);
32.             } else {
33.                     if (ch3 < ch1) {
34.                             printf("%s < %s\n", ch3, ch1);
35.                     } else {
36.                             printf("%s > %s\n", ch3, ch1);
37.                     }
38.             }
39.     }
```

Q. 5 Exécuter le programme et expliquer les résultats affichés

Pour comparer deux chaînes de caractères il faut utiliser la fonction `strcmp` qui a deux arguments qui sont les chaînes à comparer et qui retourne un entier :

```
1.      comp = strcmp(ch1, ch2) ;
```

On a `comp < 0` si `ch1 < ch2`, `comp = 0` si `ch1 = ch2` et `comp > 0` si `ch1 > ch2`.

Q. 6 Modifier votre programme de façon à faire les bonnes comparaisons

Une autre fonction, `strncmp`, a un troisième argument entier qui permet de limiter la comparaison aux premiers caractères des deux chaînes :

```
1.      strncmp(ch1, ch2, n) ;
```

Le résultat est le même que pour `strcmp`, mais la comparaison concerne au plus les `n` premiers caractères de chaque chaîne.

5 Recherches

On peut rechercher un caractère ou une chaîne dans une chaîne.

La fonction `strchr` a un premier argument qui est une chaîne et un second argument qui est un caractère et elle retourne un pointeur de caractère :

```
1.      p = strchr(ch, car);
```

Le pointeur `p` désigne la première occurrence de `car` dans `ch` ou contient la valeur `NULL` si `car` n'apparaît pas dans `ch`. Si on souhaite connaître l'indice de `car` dans `ch`, sa valeur est `p-ch`.

Créez un nouveau fichier *recherches.c* contenant le code suivant :

```
1.      #include <stdio.h>
2.      #include <string.h>
3.
4.      void main() {
5.          char ch[] = "une chaine de caracteres";
6.          char* p;
7.
8.          p = strchr(ch, 'e');
9.          printf("L'adresse du premier 'e' dans la chaîne est : %x\n", p);
10.         printf("Son indice est : %d\n", p-ch);
11.     }
```

Q. 7 Ajouter à ce programme le remplacement dans la chaîne de ce premier 'e' par une majuscule, en utilisant directement le pointeur, refaire une recherche de 'e' pour constater que la recherche est sensible à la casse, puis y remettre une minuscule en utilisant l'indice.

Q. 8 Ajouter la recherche du caractère 'z' dans la chaîne `ch` et vérifier que le pointeur retourné est bien nul.

La fonction `strrchr` (remarquer le deuxième r, qui signifie reverse) fait le même travail, mais pointe la dernière occurrence du caractère cherché.

Q. 9 Ajouter à votre programme les mêmes opérations que précédemment mais en faisant la recherche de la dernière occurrence.

La fonction `strstr` recherche une sous-chaîne dans une chaîne. Les deux arguments sont des chaînes de caractères. La première est la chaîne dans laquelle on cherche et la seconde est celle que l'on cherche. La valeur retournée est un pointeur sur la première occurrence de la chaîne cherchée si elle existe et un pointeur nul sinon.

Ajouter le code suivant à votre programme :

```
1.      char* debut = ch;
2.      char motif[] = "ne";
```

```
3.       p = strstr(debut, motif);
4.       while (p != NULL) {
5.         printf("\"%s\" se trouve en %d dans \"%s\"\n", motif, p-ch, ch);
6.         debut = p + strlen(motif);
7.         p = strstr(debut, motif);
8.       }
```

Q. 10 Exécuter le programme et découvrir ce qu'il fait.

6 Conversion de valeurs numériques

Il existe des fonctions pour convertir des chaînes contenant une valeur numérique en entier ou en flottant.

Les signatures de ces fonctions se trouvent dans `<stdlib.h>`.

La fonction strtol converti une chaîne représentant un entier. Elle a trois arguments. Le premier est la chaîne à convertir, le second est un pointeur que nous n'utiliserons pas ici et qui sera nul et le troisième est la base dans laquelle le nombre est représenté. Cette base doit être comprise entre 2 et 36 ou avoir la valeur spéciale 0. La fonction retourne un entier de type long.

La chaîne à convertir doit contenir une suite de chiffres de la base éventuellement précédée d'espaces et d'un signe + ou -. La conversion s'arrête dès la rencontre d'un caractère illégal. En particulier, si un caractère illégal est rencontré avant le premier chiffre, la fonction retourne 0.

Créez un nouveau fichier *conversions.c* contenant le code suivant :

```
1.       #include <stdio.h>
2.       #include <string.h>
3.       #include <stdlib.h>
4.       void main() {
5.           int i;
6.           char entier[10];
7.
8.           strcpy(entier, "100");
9.           i = strtol(entier, NULL, 10);
10.          printf("%s interprété en base 10 : %3d\n", entier, i);
11.          i = strtol(entier, NULL, 16);
12.          printf("%s interprété en base 16 : %3d\n", entier, i);
13.          i = strtol(entier, NULL, 8);
14.          printf("%s interprété en base  8 : %3d\n", entier, i);
15.          i = strtol(entier, NULL, 0);
16.          printf("%s interprété en base  0 : %3d\n", entier, i);
17.      }
```

Q. 11 Exécuter le programme puis le modifier pour essayer les chaînes suivantes : "0100", "0x100". En déduire la signification de la pseudo base 0.

Q. 12 Modifier le programme pour essayer les chaînes suivantes : "-100+1, " +18", " 2a". Vérifier le traitement des caractères invalides.

La fonction `strtof` converti une chaîne représentant un réel. Elle a deux arguments. Le premier est la chaîne à convertir et le second est un pointeur que nous n'utiliserons pas ici et qui sera nul. Elle retourne une valeur de type `float`. La fonction `strtod` fait la même conversion, mais le résultat est de type `double`.

7 Bilan

Vous savez maintenant :

- copier des chaînes ;
- comparer des chaînes ;
- faire des recherches dans des chaînes ;
- convertir des chaînes en valeurs numériques,

8 Pour aller plus loin

N'oubliez pas d'utiliser la commande `man` qui vous permet de retrouver toutes les informations sur une fonction de la bibliothèque C. Par exemple :

```
man strlen
```

vous donne la documentation de la fonction `strlen`.

Vous pouvez aussi avoir la liste de nombreuses fonctions de manipulation de chaîne en tapant :

```
man string
```