- Introduction à l'algorithmique et au langage C -

Chaines de caractères

TP n°4

1re année ESIEA - Semestre 1

L. Beaudoin & R. Erra & A. Gademer & L. Avanthey

2015 - 2016

Avant propos

Nous allons aborder dans ce TP un cas particulier des tableaux : les chaînes de caractères. Nous en avons déjà entendu parler, et il est temps maintenant de comprendre vraiment ce dont il s'agit et d'apprendre à les manipuler.

1 Le cas particulier des chaînes de caractères

Jusqu'ici nous avons appris à manipuler des caractères (`'a'`, `'+'`, etc.) et à définir des tableaux qui nous permettent de déclarer des **tableaux de caractères**.

```
char word[7] = {'B','o','n','j','o','u','r'};
```

Mais nous avons aussi entendu parler des **chaînes de caractères** (avec `printf` et `scanf`) sous la forme de phrases entre guillemets droits : `"Hello World"`.

Chaînes de caractères : En C, les chaînes de caractères sont des tableaux de caractères qui se terminent par le caractère spécial `\0`.

```
char word[8] = {'B','o','n','j','o','u','r','\0'}; // This is a string of 7 letters.
// Note the ending char '\0', the 8th element.
```

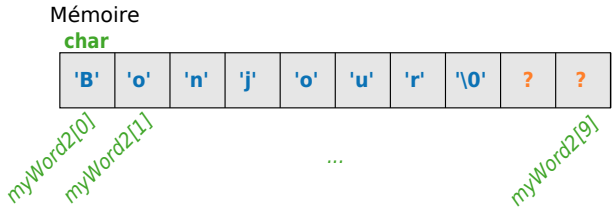
Nous noterons que lors de la déclaration, nous pouvons initialiser les chaînes de caractères de la manière suivante :

```
char myWord[8] = "Bonjour"; // This is also a string. We still need EIGHT characters.
// Here the '\0' is implicitly added because the double quotes define a characters' string
```

Contenu et contenant

Remarquez que vous pouvez tout à fait stocker une chaîne de caractères dans un tableau plus grand que la taille de la chaîne elle-même. Le caractère `'\0'` marquera de toute manière la fin de la chaîne de caractères.

```
char myWord2[10] = "Bonjour";
```



QUESTION 1

Déclarez un tableau de 15 caractères et initialisez-le avec la chaîne de caractères `"I got it!"`.

Avantage du caractère terminal \0 ?

Nous avons vu dans les TD/TP sur les tableaux que nous devions passer, en plus du nom du tableau, la taille des tableaux en paramètre des fonctions afin de savoir quand s'arrêter.
Ce n'est pas nécessaire pour les chaînes de caractères, car nous pouvons déclarer un caractère spécial unique : il suffit alors de nous arrêter dès que nous trouvons ce caractère terminal. La boucle de parcours dans ce cas particulier n'est donc plus itérative mais événementielle !

Dépassement de tableau

Le fait que la taille d'une chaîne de caractères ne soit pas directement liée à celle du tableau peut engendrer de nombreuses erreurs. Soyez très prudents !

En particulier :

- l'initialisation ou la copie d'une chaîne dans un tableau trop petit provoque un avertissement (warning) et des problèmes à l'exécution :

```c
char word[4] = "Goodbye";
```

- l'oubli ou la disparition du caractère `'\0'`, qui ne marque alors plus la fin de la chaîne, provoque des erreurs sournoises à l'exécution :

```c
char word[8] = "Goodbye";
word[7] = '!'; // No more '\0'
```

2 Manipulation des chaînes de caractères

Les chaînes de caractères étant au cœur des systèmes d'information, il existe de nombreuses fonctions qui leur sont spécifiquement dédiées.

2.1 `printf /scanf`

Nous découvrons un nouveau descripteur de type : `"%s"` qui permet d'afficher des mots ou de récupérer des mots saisis au clavier. Si nous ajoutons un nombre devant le `s`, celui-ci limite le nombre maximum de caractères récupérés pour un mot. Par exemple : `"%10s"` ne permet pas de récupérer plus de 10 caractères (`'\0'` non inclus). Cela permet d'éviter de déborder du tableau de caractères préalablement défini.

Mots : En C, les mots sont des suites de caractères séparés par un ou plusieurs espaces, tabulations (\t) ou retours à la ligne (\n).

```c
char word[50];
printf("Please enter a word (30 char max):\n");
if(scanf("%30s", word) != 1) { // WARNING scanf %s takes the whole array in parameter, so we
    give it only the NAME of the char array
   printf("Input error\n");
   exit(-1);
}
printf("You have entered the word: %s\n", word);
```

Attention

La fonction `scanf` récupère toute une chaîne de caractères d'un seul coup. Elle va s'occuper ensuite de mettre elle-même chacun des caractères de la chaîne dans les cases respectives du tableau de caractères et rajoutera le caractère terminal `'\0'`. Nous n'avons plus besoin de boucle et nous lui donnons le tableau dans son ensemble, c'est pourquoi nous envoyons son nom. Il en va de même pour la fonction `printf`.

Ce qui donne comme rendu (seul le premier mot est récupéré par `"%s"`) :

```
Please enter a word (50 char max):
Hello World!
You have entered the word: Hello
```

Notez-bien

Les autres mots que nous n'avons pas récupérés (ici `"World!"`) restent en attente dans le flux d'entrée. Si nous faisons un autre `scanf`, ce sont eux qui viendront en premiers et non les nouveaux mots que nous entrons au clavier.

Pour récupérer une ligne entière nous utilisons le descripteur `"\n%[^\n]"`. Nous pouvons aussi limiter la taille de la chaîne en entrée avec la notation `"\n%50[^\n]"`.

```c
char sentence[50];
printf("Please enter a sentence (50 char max):\n");
if(scanf("\n%50[^\n]", sentence) != 1) { // WARNING scanf take only the NAME of the char array
   printf("Input error\n");
   exit(-1);
}
printf("You have entered the sentence: %s\n", sentence);
```

Ce qui donne comme rendu (nous récupérons toute la ligne) :

```
Please enter a sentence (50 char max):
Hello World!
You have entered the sentence: Hello World!
```

2.2 Bibliothèque `string.h`

La bibliothèque `string.h` (`#include <string.h>`) propose de nombreuses fonctions orientées autour de la manipulation de chaînes de caractères dont :

- `strlen` qui donne la taille d'une chaîne de caractères.

```c
char sentence[50] = "Hello World!";
int len;
len = strlen(sentence);
printf("The sentence '%s' has %d characters\n", sentence, len);
```

```
The sentence 'Hello World!' has 12 characters
```

3 Arithmétique des caractères

Dans le TD « *Introduction aux tableaux* », nous avons vu comment les caractères étaient associés à un nombre par le biais de la table ASCII [1].

Nous rappelons qu'il est possible de visualiser l'ensemble de ces correspondances caractère-nombre de la table ASCII en entrant la commande `man ascii`.

En voici de nouveau un extrait :

'+'	...	'0'	'1'	...	'9'	...	'A'	'B'	...	'Z'	...	'a'	'b'	...	'z'
43		48	49		57		65	66		90		97	98		122

Nous avions remarqué :
- que la disposition des caractères est arbitraire (en particulier pour les symboles),
- que les chiffres et les lettres restent ordonnés et contigus par bloc.

Nous avions donc vu qu'il était possible d'utiliser les opérations de bases :
- `'0' + 9 = 48 + 9 = 57 = '9'`
- `'A' + 25 = 65 + 25 = 90 = 'z'`
- `'m' - '3' = 109 - 3 = 106 = 'j'`

Nous pouvons utiliser ces opérations de bases pour effectuer des actions particulières :
- `'b' - 'a' = 98 - 97 = 1`
- `'b' - 'a' + 'A' = 98 - 97 + 65 = 1 + 65 = 66 = 'B'`

Nous en déduisons donc quelques algorithmes de base :
- Un caractère `cVal` est une minuscule si `cVal >= 'a' && cVal <= 'z'`.
- Un caractère `cVal` est une majuscule si `cVal >= 'A' && cVal <= 'Z'`.
- Nous transformons un caractère `cVal` minuscule en une majuscule en lui appliquant le décalage suivant : `cVal - 'a' + 'A'` .
- Nous transformons un caractère `cVal` majuscule en une minuscule en lui appliquant le décalage suivant : `cVal - 'A' + 'a'` .

4 Mise en application

Taille du tableau de caractères

Sauf précisions ultérieures, nous considérerons que nos chaînes de caractères ne font jamais plus de 50 caractères `'\0'` compris. Les tableaux de caractères alloués dans le `main` auront donc 50 cases (même si l'utilisateur entre des mots plus petits).

EXERCICE 1

Écrivez la fonction `void loadWord(char word[])` qui remplit le tableau `word` avec un **mot** saisi par l'utilisateur. Récupérez uniquement le premier mot. Pensez à protéger votre tableau en évitant les débordements. Cette fonction ne contient pas de `printf`, ces derniers sont réservés au `main`.

Exemple de rendu :

```
Please enter a word (50 char max):
Hello
You have entered the word: Hello
```

1. American Standard Code for Information Interchange.

EXERCICE 2

 Écrivez la fonction `void loadSentence(char sentence[])` qui remplit le tableau `sentence` avec une chaîne de caractères saisie par l'utilisateur. Récupérez toute la ligne. Pensez à protéger votre tableau en évitant les débordements. Cette fonction ne contient pas de `printf`, ces derniers sont réservés au `main`.

Exemple de rendu de votre `main` :

```
Please enter a sentence (50 char max):
Hello World!
You have entered the sentence: Hello World!
```

EXERCICE 3

 Écrivez la fonction `void toUpperCase(char sentence[])` qui parcourt la chaîne de caractères (attention, seul le `'\0'` marque l'arrêt) et change toutes les minuscules en majuscules. Cette fonction ne contient pas de `printf`, ces derniers sont réservés au `main`.

Exemple de rendu :

```
Please enter a sentence (50 char max):
Hello World!
Uppecase: HELLO WORLD!
```

EXERCICE 4

 Écrivez la fonction `void toLowerCase(char sentence[])` qui parcourt la chaîne de caractères (attention, seul le `'\0'` marque l'arrêt) et change toutes les majuscules en minuscules. Cette fonction ne contient pas de `printf`, ces derniers sont réservés au `main`.

Exemple de rendu :

```
Please enter a sentence (50 char max) :
Hello World!
Lowercase : hello world!
```

EXERCICE 5

 Écrivez la fonction `void mirror(char sentence[])` qui inverse l'ordre des caractères **sans utiliser de second tableau** (pensez à l'algorithme d'échange). Vous aurez besoin de la fonction `strlen` décrite dans la bibliothèque `string.h`. Cette fonction ne contient pas de `printf`, ces derniers sont réservés au `main`.

Exemple de rendu :

```
Please enter a sentence (50 char max):
Hello World!
Mirror: !dlroW olleH
```

Mémo

« Une chaîne de caractères se termine toujours par le caractère terminal `'\0'`. »
« Le parcours d'une chaîne de caractères est événementielle. »
« Il faut faire attention à ne pas récupérer une chaîne de caractères plus grande que la taille du tableau. »