RÉVISION TABLEAUX ET CHAINES DE CARACTÈRES

AÏCHA EL GOLLI
aicha.elgolli@essai.ucar.tn
Novembre-décembre 2022

ECOLE SUPÉRIEURE DE LA STATISTIQUE
ET DE L'ANALYSE DE L'INFORMATION

Initialisation avec une longueur explicite

Comme pour n'importe quel tableau, l'initialisation se réalise à l'aide d'une liste d'initialisation. L'exemple ci-dessous définit donc un tableau de vingt-cinq char et initialise les sept premiers avec la suite de lettres « Bonjour ».

```
char chaine[25] = { 'B', 'o', 'n', 'j', 'o', 'u', 'r' };
```

Étant donné que seule une partie des éléments est initialisée, les autres sont implicitement mis à zéro, ce qui nous donne une chaîne de caractères valide puisqu'elle est bien terminée par un caractère nul. Faites cependant attention à ce qu'il y ait *toujours* de la place pour un caractère nul.

Initialisation avec une longueur implicite

Dans le cas où vous ne spécifiez pas de taille lors de la définition, il vous faudra ajouter le caractère nul à la fin de la liste d'initialisation pour obtenir une chaîne valide.

```
char chaine[] = { 'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0' };
```

strlen

La fonction strlen() vous permet de connaître la taille d'une chaîne fournie en argument.

```
#include <stdio.h>
#include <string.h>
int main(void) {
        printf("Longueur : %d\n", strlen("Bonjour"));
        return 0;
}
```

Longueur : 7

strcpy

```
char *strcpy(char *destination, char *source);
```

La fonction strcpy() copie le contenu de la chaîne source dans la chaîne destination, caractère nul compris. La fonction retourne l'adresse de destination. L'exemple ci-dessous copie la chaîne « Au revoir » dans la chaîne chaine.

```
#include <stdio.h>
int main(void) {
char chaine[25] = "Bonjour\n";
strcpy(chaine, "Au revoir");
printf("%s\n", chaine);
return 0;
}
```

Au revoir

strcat

```
char *strcat(char *destination, char *source);
```

La fonction strcat() ajoute le contenu de la chaîne source à celui de la chaîne destination, caractère nul compris. La fonction retourne l'adresse de destination.
L'exemple ci-dessous ajoute la chaîne « tout le monde » au contenu de la chaîne chaine.

```
#include <stdio.h>
#include <string.h>
int main(void) {
char chaine[25] = "Bonjour";
strcat(chaine, " tout le monde");
printf("%s\n", chaine); return 0;
}
```

Bonjour tout le monde

Comme strcpy(), la fonction strcat() n'effectue *aucune* vérification. Vous devez donc vous assurer que la chaîne de destination dispose de suffisamment d'espace pour accueillir la chaîne qui doit être ajoutée (caractère nul compris !)

LE CARACTÈRE DE FIN DE CHAÎNE

- Le zéro de fin de chaîne correspond au premier caractère de la table ASCII (à ne pas confondre avec le caractère '0' qui a pour code ASCII 48)
- Le caractère de fin de chaîne peut être représenté de différentes façons :
  - sous sa forme décimale : 0
  - sous sa forme hexadécimale : 0x00
  - sous forme de caractère : '\0‘

Exercice: On donne la déclaration de la chaîne suivante :

```
char str[10] = "abc";
```

À l'aide d'une boucle, parcourir un à un les caractères du tableau afin de les afficher sous forme :

- de caractères
- décimale
- hexadécimale

Voici l'affichage attendu de cet exercice :

```
int i;
for (i=0 ;i<10;i ++)//ou != 0 {
printf ("str[%d]\t= '%c'\t= %2d\t= 0x%x\n", i, str[i], str[i], str[i]);
```

```
str[0] = 'a'  = 97   = 0x61
str[1] = 'b'  = 98   = 0x62
str[2] = 'c'  = 99   = 0x63
str[3] = ''   = 0    = 0x00
str[4] = ''   = 0    = 0x00
str[5] = ''   = 0    = 0x00
str[6] = ''   = 0    = 0x00
str[7] = ''   = 0    = 0x00
str[8] = ''   = 0    = 0x00
str[9] = ''   = 0    = 0x00
```

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.

a) `char a[] = "un\ndeux\ntrois\n";`

Déclaration correcte

Espace: 15 octets

b) `char b[12] = "un deux trois";`

Déclaration incorrecte: la chaîne d'initialisation dépasse le bloc de mémoire réservé.

Correction: char b[14] = "un deux trois";

ou mieux: char b[] = "un deux trois";

Espace: 14 octets

c) `char c[] = 'abcdefg';`

Déclaration incorrecte: Les symboles '' encadrent des caractères; pour initialiser avec une chaîne de caractères, il faut utiliser les guillemets (ou indiquer une liste de caractères).

Correction: char c[] = "abcdefg";

Espace: 8 octets

d) `char d[10] = 'x';`

Déclaration incorrecte: Il faut utiliser une liste de caractères ou une chaîne pour l'initialisation

Correction: char d[10] = {'x', '\0'}

ou mieux: char d[10] = "x";

Espace: 2 octets

```
e) char e[5] = "cinq";
```
Déclaration correcte

Espace: 5 octets

```
f) char f[] = "Cette " "phrase" "est coupée";
```
Déclaration correcte

Espace: 23 octets

```
g) char g[2] = {'a', '\0'};
```
Déclaration correcte

Espace: 2 octets

```
h) char h[4] = {'a', 'b', 'c'};
```
Déclaration correcte. Étant donné que seule une partie des éléments est initialisée, les autres sont implicitement mis à zéro, ce qui nous donne une chaîne de caractères valide puisqu'elle est bien terminée par un caractère nul

Espace: 4 octets

```
i)  char i[4] = "'o'";
```
Déclaration correcte, mais d'une chaîne contenant les caractères ', 'o', ' et '\0'.

Espace: 4 octets

On désire écrire un algorithme qui construit un tableau de caractères, calcul et affiche le nombre d'occurrences d'une la lettre et insère un caractère. L'algorithme, commenté par la suite, est le suivant :

Algorithme tableauAlea
variables tabCarac [100] **: caractère ;**
nbVal : **entier ;**
Début

{Procédure init qui initialise un tableau de n caractères est décrit ci-dessous}
init (tabCarac, nbVal) ;

{Appel à la fonction rechercher qui renvoie le nombre d'occurrences d'une lettre dans un tableau caractères et décrite ci-dessous}
afficher ("le nombre d'occurrences de la lettre e est =: ", ***rechercher*** (tabCarac, nbVal, 'e')) ;

{Procédure insert qui insère un caractère dans un tableau de n caractères est décrite ci-dessous}
insert (tabCarac, nbVal) ;
fin

1. Définir une procédure **init** qui demande à l'utilisateur de saisir la valeur d’un entier ***n*** positif ou nul (vérifier que la valeur saisie est bonne, et redemander si nécessaire), et qui ensuite saisit les ***n*** éléments d’un tableau de caractères. N’oubliez pas que le tableau ainsi que sa taille doivent être des paramètres données/résultat.

2. Définir une fonction **rechercher** qui reçoit un tableau de ***n caractères*** en paramètre ainsi que la lettre à rechercher, et qui calcule et renvoie le nombre d'occurrences d'une lettre donnée dans le tableau.

3. Définir une procédure **insert** qui permet d'insérer un caractère dans un tableau de caractères. La procédure **insert** reçoit en paramètre un tableau ***T*** de caractères et l'entier **n** représentant la taille réelle du tableau T. Cette procédure **insert** demande à l'utilisateur le caractère à insérer et la position dans laquelle sera insérer le caractère (cette position doit être comprise entre **0** et **n**).

4. Dites ce que fait cette procédure?

```
Procédure Mystere(d/r t [ ]:entier, d/r n : entier)
i, j : entier ;
Début
    j←0;
    Pour(i←0 à n-1)faire
        t[j]←t[i];
        Si(t[i]<>0) alors     j←j+1;
    FPour ;
    n←j;
Fin ;
```

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 ! ! esarhp etitep enu iciov
```