Programmation Langage C
Les chaînes de caractères

Youssef ALJ

30 mars 2020

Pourquoi les chaines de caractères

- Chaine de caractères = texte.
- Jusqu'à maintenant, on sait manipuler les `int`, `float`, etc. et faire des opérations ($+$, $\times$, $\div$, $-$)
- On veut écrire un programme C pour manipuler un texte. Exemples de traitements :
  - Récupérer un sous texte du texte de départ.
  - Voir si un mot se trouve dans un texte.
  - etc.
- On va introduire le type `char` pour stocker une lettre.
- On utilisera ensuite un tableau de `char` pour stocker une chaine de caractères.

Exemple :

B	o	n	j	o	u	r	\0

Le type char

- Le type char est prévu pour stocker une lettre.
- La mémoire ne peut stocker que des nombres.
- on a donc créé une table qui fait la conversion entre les nombres et les lettres correspondantes.
- Si on veut stocker une lettre 'A' en mémoire, l'ordinateur la convertit en nombre.
- La conversion de chaque lettre est prévue par le codage ASCII.
- ASCII : American Standard Code for Information Interchange
- ASCII : C'est une table qui contient tous les caractères que l'utilisateur peut saisir.

- Pour stocker une lettre on saisir la lettre ente apostrophes ' '.
- A la compilation, 'A' sera remplacée par la valeur correspondante.
- `char` permet de stocker des entiers aussi.
- Mais uniquement les entiers entre -128 et 127.
- `unsigned char` stocke les entiers entre 0 et 255.
- Pour afficher un caractère on doit utiliser `%c`.
- Les minuscules ont des valeurs ascii différentes. ATTENTION : 'a' $\neq$ 'A'.

```c
#include <stdio.h>
// on convertit ici une
// lettre en le nombre ASCII correspondant
void main()
{
    char lettreA = 'A';
    printf("Codage ascii de %c est %d\n",
          lettreA, lettreA);
}
```

Résultat : Codage ascii de A est 65.

```c
#include <stdio.h>
// on convertit ici un entier en la
// lettre correspondante dans le codage
     ASCII
void main()
{
    char devinerLettre = 66;
    printf("la lettre devinerLettre est %c
         ", devinerLettre);
}
```

Résultat : La lettre devinerLettre est B.

Lire/écrire un caractère

```c
#include <stdio.h>

void main()
{
    char lettre = 0; //0 ou '\0'
    scanf("%c", &lettre);
    printf("%c\n", lettre);
}
```

- Pour initialiser un caractère, on utilise soit :
  - Le nombre 0.
  - Le caractère nul : `\0`.
- On utilise le spécificateur de format `%c` pour lire ou afficher un caractère.

Définition

Une chaîne de caractères est un tableau de caractères.

Déclaration et initialisation

- La chaine de caractère se déclare comme un tableau.
- On doit ajouter le caractère nul `\0` pour marquer la fin de la chaîne.
- Pour l'afficher on utilise `%s`.

Initialisation : méthode 1

```c
#include <stdio.h>

void main(){
    // methode 1 : comme un tableau
    // Le mot bonjour contient 7 lettres
        => declarer un tableau de taille
        8
    // Le dernier element du tableau
        contient le caractere nul
    char chaine1 [8] = {'B', 'o', 'n', 'j'
        , 'o', 'u', 'r', '\0'};
    printf("chaine1=%s", chaine1);
}
```

Déclaration et initialisation

- On accède à notre tableau de caractères élément par élément.

Initialisation : méthode 2

```c
#include <stdio.h>

void main(){
    // methode 2 : fatigante
    char chaine2 [8];
    chaine2[0] = 'B';
    chaine2[1] = 'o';
    chaine2[2] = 'n';
    chaine2[3] = 'j';
    chaine2[4] = 'o';
    chaine2[5] = 'u';
    chaine2[6] = 'r';
    chaine2[7] = '\0';
    printf("chaine2=%s", chaine2);
}
```

Déclaration et initialisation

- On tape entre guillemets " " la chaine de caractères.
- ATTENTION : **ne pas confondre avec les caractères pour lesquels on utilise les ' '**.
- Le compilateur calcule automatiquement la taille nécessaire :
  - Il compte le nombre de lettres et ajoute 1 pour le caractère `\0`.
- Inconvénient : cela ne marche que pour l'initialisation.
- On n'a pas le droit d'écrire plus loin : `chaine3 = "Salut";`

Initialisation : méthode 3

```c
#include <stdio.h>

void main(){

    // methode 3 plus elegante
    char chaine3 [] = "Bonjour";
    printf("chaine3=%s", chaine3);
}
```

Quelle est la taille d'un `char` en mémoire ?

- ☐ un octet
- ☐ 2 octets
- ☐ 3 octets
- ☐ 4 octets

Quelle est la taille d'un `char` en mémoire ?

- ☑ un octet (car on stocke les entiers de -128 à 127 ce qui correspond à 255 nombres différents)
- ☒ deux octets
- ☒ trois octets
- ☒ quatre octets

Dire quelles sont les chaines qui sont initialisées correctement ?

```
1  //exercice initialisation de chaines de caracteres
2  #include <stdio.h>
3  void main(){
4      char a[]   = "un\ndeux\ntrois\n";
5      char b[12] = "un deux trois";
6      char c[]   = 'abcdefg';
7      char d[10] = 'x';
8      char e[5]  = "cinq";
9      char f[]   = "Cette "   "phrase"   "est coupee";
10     char g[2]  = {'a', '\0'};
11     char h[4]  = {'a', 'b', 'c'};
12     char i[4]  = "'o'";
13 }
```

Solution 1/2

```c
//exercice initialisation de chaines de caracteres
#include <stdio.h>
void main(){
    char a[]   = "un\ndeux\ntrois\n";
    char b[12] = "un deux trois";
    char c[]   = 'abcdefg';
    char d[10] = 'x';
    char e[5]  = "cinq";
    char f[]   = "Cette "   "phrase"   "est coupee";
    char g[2]  = {'a', '\0'};
    char h[4]  = {'a', 'b', 'c'};
    char i[4]  = "'o'";
}
```

```
execrcice_1.c: In function 'main':
execrcice_1.c:5:15: warning: initializer-string for array of chars is too long
    5 |   char b[12] = "un deux trois";
      |                ^~~~~~~~~~~~~~~
execrcice_1.c:6:15: warning: character constant too long for its type
    6 |   char c[]   = 'abcdefg';
      |                ^~~~~~~~~
execrcice_1.c:6:15: error: invalid initializer
execrcice_1.c:7:15: error: invalid initializer
    7 |   char d[10] = 'x';
      |                ^~~
```

Solution 2/2

- `char a[] = "un\ndeux\ntrois\n";`
  Déclaration correcte
- `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";`
- `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";`
- `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";`
- `char e[5] = "cinq";`
  Déclaration correcte.
- `char f[] = "Cette " "phrase" "est coupée";`
  Déclaration correcte
- `char g[2] = {'a', '\0'};`
  Déclaration correcte.
- `char h[4] = {'a', 'b', 'c'};`
  Déclaration incorrecte : Dans une liste de caractères, il faut aussi indiquer le symbole de fin de chaîne.
  Correction : `char h[4] = {'a', 'b', 'c', '\0'};`
- `char i[4] = "'o'";`
  Déclaration correcte, mais d'une chaîne contenant les caractères `'''`,`'o'`,`'''` et `'\0'`.

Écriture avec printf

- `printf` avec le spécificateur `%s` permet d'intégrer une chaine de caractères dans une phrase.

Exemple printf

```
#include <stdio.h>

void main(){

    char chaine1 [] =  "ceci est une
          chaine";
    printf("Le contenu de chaine1 est %s",
          chaine1);
}
```

Écriture avec puts

- `puts` est idéale pour écrire une chaîne constante ou le contenu d'une variable dans une ligne isolée.
- `puts(chaine);`
  ⇕
  `printf("%s\n",chaine);`

Exemple puts

```
#include <stdio.h>

void main(){

    char chaine1 [] =  "ceci est une
          chaine";
    //ecriture d une chaine constante
    puts("ceci est une autre chaine");

    //ecriture d une variable chaine de
          caracteres
    puts(chaine1);
}
```

Lecture avec scanf

- `scanf` avec `%s` permet de lire un **mot isolé** à l'intérieur d'une suite de données du même ou d'un autre type.
- les noms des variables numériques (`int`, `float`, etc) doivent être marqués par le symbole &.
- le nom d'une chaine de caractères est aussi l'adresse du premier caractère de la chaine. **Il ne doit pas être précédé de &.**

```c
// exemple lecture avec scanf
#include <stdio.h>
void main(){
    char lieu[30];
    int jour, mois, annee;
    printf("Entrez date de naissance et
        lieu: \n");
    scanf("%d %d %d %s", &jour, &mois, &
        annee, lieu);
}
```

Lecture avec gets

- `gets` est idéale pour lire une ou plusieurs lignes de texte. (e.g. des phrases) terminées par un retour à la ligne.
- `gets(chaine)` lit une ligne de caractères et la copie dans chaine.

Exemple gets

```c
#include <stdio.h>

void main(){

    int MAXI = 1000;
    char ligne[MAXI];
    gets(ligne);
}
```

Exercice 2

Écrire un programme qui lit 3 mots, séparés par des espaces et qui les affiche ensuite dans une ligne, mais dans l'ordre inverse. Les mots sont mémorisés dans 3 variables mot1, mot2 et mot3.
Exemple :
Bonjour ! ca va ?
va ? ca Bonjour

Solution exercice 2

```c
#include <stdio.h>

void main(){
    int taille_max = 30;
    char mot1[taille_max], mot2[taille_max], mot3[taille_max];
    printf("Entrez 3 mots, separes par des espaces :\n");
    scanf("%s %s %s", mot1, mot2, mot3);
    printf("%s %s %s\n", mot3, mot2, mot1);
}
```

Exercice 3

Écrire un programme qui lit une ligne de texte et qui ne dépasse pas 200 caractères et qui affiche ensuite :

- La longueur de la chaine.
- Le nombre de 'e' qui se trouve dans le texte.

Solution : exercice 3

```c
#include <stdio.h>
// solution exercice 3
void main(){
    // on declare une ligne qui ne depasse pas 200 caracateres
    // donc la taille sera 201 maxi
    char ligne[201];
    puts("Veuillez saisir une ligne de texte");
    gets(ligne);
    int i=0, size=0, compte_e=0;
    while (ligne[i] != '\0'){
        size++;
        if (ligne[i] == 'e'){
            compte_e++;
            }
        i++;
    }
    printf("size ligne=%d\n", size);
    printf("nombre de e=%d\n", compte_e);
}
```