LES CHAINES DE CARACTERES

SANE ARNAUD

Table des matières

Objectifs

- Etre capable de déclarer et d'initialiser une chaîne de caractères
- Etre capable de manipuler une chaîne de caractères à partir de ses fonctions

DECLARATION ET INITIALISATION D'UNE CHAINE DE CARACTERES

DECLARATION D'UNE CHAINE DE CARACTERES

Une chaîne de caractères sert à stocker des valeurs de type alphanumériques. Elle est constituée d'un ensemble de caractères, on parle donc de tableau de caractères car en langage C, elle est déclarée comme un tableau.

Syntaxe :

char nom_chaine[taille] ;

ou

char nom_chaîne[] ;

NB :

- le dernier caractère en langage C contient un caractère NULL ('\0'), ainsi si l'on a 15 caractères à saisir, il faut prévoir comme espace mémoire du tableau, 16 caractères.
- le type utilisé pour déclarer une chaîne de caractères est *char* et non string comme c'est le cas dans certain programmes. Ce type Char permet de stocker des nombres compris entre -128 et 127. le typer char est prévu pour stocker une lettre. Le compilateur du C fait une traduction entre la lettre définie entre les quottes et le nombre entier correspondant à son code ASCII.

Exemple :

char val[] ;

ou

char val[15] ; //ici les caractères renseignés partent de val[0] à val[14] qui vaut '\0'.

INITIALISATION D'UNE CHAINE DE CARACTERES

L'initialisation d'une chaîne de caractères peut se faire pendant la déclaration de la variable.

Ainsi, il existe plusieurs méthodes d'initialisation d'une variable lors de sa déclaration :

Methode 1 : char val[]={'A','R','N','A','U','D','\0'} ;

Methode 2 : char val[]="ARNAUD" ; // ici, lors de l'initialisation, l'ordinateur réserve automatiquement le nombre d'octets nécessaires pour la chaîne c'est à dire dans notre cas, 6 caractères plus un (réservé au caractère null).

Methode 3 : char val[7]="ARNAUD" ; //ici, le nombre de caractères défini entre les crochets doit être supérieur au nombre de caractères de la chaîne assignée au tableau (pour prendre en compte le caractère de NULL de fin de chaîne. Dans notre cas "ARNAUD" comprend 6 caractères et par conséquent, la taille définie doit être supérieure à cette valeur (raison pour laquelle nous l'avons mise à 7.

LES ENTREES / SORTIES LIEES AUX CHAINES DE CARACTERES

1. *Les fonctions d'entrée de chaîne de caractères*

    - *La fonction scanf*

    La récupération d'une chaîne de caractères peut être faite à l'aide de la fonction scanf, mais tout en spécifiant le format.(%s pour la chaîne de caractères ou %c pour un caractère) et l'adresse de la variable en second paramètre. Exemple : scanf("%s",&titre).

    - *La fonction gets*

    La récupération d'une chaîne de caractères peut également être faite à l'aide de la fonction gets, mais cette fois, sans spécifier de format. Il suffit simplement de définir en paramètres le nom de la variable. La fonction lit une ligne jusqu'au retour chariot et remplace le '\n' par '\0' dans l'affectation de la chaîne.
    Exemple : gets(titre) ;
2. *Les fonctions d'affichage de chaîne de caractères*

    - *La fonction printf*

    L'affichage d'une chaîne de caractères peut se faire à l'aide de la fonction printf qui prend en paramètre le format d'affichage en premier et la variable en second paramètre. *Exemple printf("%s",titre) ;*

    - *La fonction puts*

    L'affichage d'une chaîne de caractères peut également se faire à l'aide de la fonction puts, qui prend en paramètre le nom de la variable. cette fonction applique un retour chariot juste après l'affichage de la chaîne de caractères.
    Exemple : puts(titre) ; est identique à l'instruction *printf("%s\n",titre) ;*

EVALUATION SUR LA DECLARATION DES CHAINES DE CARACTERES

Exercice 1

De ces instructions, qu'elle est celle qui n'est pas correcte ?

- ◯ char val[5]="KONAN" ;
- ◯ char val[]="KONAN" ;
- ◯ char val[6]="KONAN" ;

Exercice 2

De ces instructions, qu'elle est celle qui est correcte ?

- ◯ char val[8]='Bonjour' ;
- ◯ char val[]='Bonjour' ;
- ◯ char val[8]="Bonjour" ;

III LES FONCTIONS DE MANIPULATION DES CHAINES DE CARACTERES

Pour la manipulation des chaînes de caractères en langage C, il est indispensable de déclarer la bibliothèque <string.h> qui rend possible l'exploitation des fonctions liées aux chaînes.

LA FONCTION strlen

Rôle :

Cette fonction retourne la longueur (nombre de caractères excepté le caractère final '\0')d'une chaîne définit en paramètre.

Syntaxe :

strlen(chaîne)

Exemple :

```
1
2 #include<stdio.h>
3 #include<conio.h>
4 #include<string.h>
5 main()
6 {
7   int l;  // déclaration de la variable l
8   char test[6]="titre";  //Déclaration et initialisation de la variable test
9   l=strlen(test);   //la variable l reçoit le nombre de caractères de la chaîne
  assignée à la variable test
10  printf("%d",l);   //Affichage du nombre de caractères à travers la variable l
  qui est de type entier.
11  getch();
12 }
```

LA FONCTION DE COMPARAISON strcmp ET SES DERIVEES

1. *LA FONCTION strcmp*
   Cette fonction compare deux chaînes de caractères à partir du code ASCII de chaque caractère des deux chaînes en commençant de part et d'autre par les caractères de la gauche.
   Syntaxe : l=strcmp(chaine1,chaine2)
   - Si chaine1=chaine2 alors l vaut 0.
   - Si chaine1<chaine2 alors l vaut -1

- Si chaine1>chain2 alors l vaut 1

Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int m;
char test[6]="titre";
char test2[6]="Titre";
m=strcmp(test,test2);
printf("%d",m);
getch();
}
// m retourne 1 car test>test2.
```

REMARQUE : Dans cet exemple test est supérieure à test2 car, en comparant le premier caractère de chaque variable, le code ASCII de t (minuscule)est supérieur à celui de T(majuscule) et par conséquent "titre">"Titre".

2. *LA FONCTION stricmp*
   Elle est une fonction dérivée de strcmp ; elle compare deux chaînes de caractères sans tenir compte de la casse. Ainsi, t(minuscule)=T(majuscule).
   Syntaxe : Elle garde la même syntaxe que strcmp
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int m;
char test[6]="titre";
char test2[6]="Titre";
m=strcmp(test,test2);
printf("%d",m);
getch();
}
// m retourne 0 car test=test2.
```

3. *LA FONCTION strncmp*
   Elle effectue la comparaison de deux chaînes sur un nombre défini de caractères.
   Syntaxe : strncmp(chaîne1,chaîne2,n) //Elle compare les n premiers caractères de chaîne1 avec les n premiers caractères de chaîne2.
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int m;
char test[9]="patricia";
char test2[9]="patrick";
m=strncmp(test,test2,6);
printf("%d",m);
getch();
}
// Dans cet exemple, m retourne 0 car les 6 premiers caractères de test donnent "patric" et ceux de test2 donnent également "patric"
```

4. *LA FONCTION strnicmp*
   elle compare également les n premiers caractères de chacune des variables définie en paramètre sans tenir compte de la casse.
   Syntaxe : strnicmp(chaîne1,chaîne2
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
```

```
main()
{
int m;
char test[9]="Patricia";
char test2[9]="patrick";
m=strnicmp(test,test2,6);
printf("%d",m);
getch();
}
// Dans cet exemple, m retourne -1 car les 6 premiers caractères de test donnent "Patric" et ceux de test2 donnent également "patric"
```

LA FONCTION strcpy et sa dérivée strncpy

1. *LA FONCTION strcpy*
   Cette fonction permet de copier une chaîne de caractères dans une autre.
   Syntaxe : strcpy(chaîne1,chaîne2) ;
   Ici, elle copie le contenu de chaîne2 dans chaîne1
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[9]="Contenu1";
char test2[9] ;
strcpy(test2,test);
puts(test2);
getch();
}
```

   //Dans cet exemple le contenu de test est reproduit dans test2, ainsi test2 retourne "Contenu1".
2. *LA FONCTION strncpy*
   La fonction strncpy copie dans le premier paramètre, le n premiers caractères du second paramètre.
   Syntaxe : strncpy(chaîne1,chaîne2,n) //les n premiers caractères de chaîne2 seront copiés dans chaîne1.
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[9]="Contenu1";
char test2[9] ;
strncpy(test2,test,4);
puts(test2);
getch();
}
```

   //Dans cet exemple les 4 premiers caractères de test sont reproduits dans test2, ainsi test2 retourne "Cont".

LA FONCTION strcat ET SA DERIVEE strncat

1. *LA FONCTION strcat*
   Cette fonction effectue la concaténation de deux chaînes de caractères mises en paramètres.
   Syntaxe : strcat(chaîne1,chaîne2) ;
   Ici, elle concatène chaîne1 et chaîne2 et l'assigne à chaîne1
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
```

```
main()
{
char test[]="Filiere";
char test2[]=" Informatique";
strcat(test,test2);
puts(test);
getch();
}
```

//Dans cet exemple le contenu de test retourne la concaténation de test et test2, ainsi test affiche "Filiere Informatique"

2. *LA FONCTION strncat*
   La fonction strncat concatène les n caractères du second paramètre avec ceux du premier paramètre.
   Syntaxe : strncat(chaîne1,chaîne2,n) //les n premiers caractères de chaîne2 seront concaténés avec toute la chaîne de chaîne1.
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[]="Filiere";
char test2[]=" Informatique";
strncat(test,test2,5);
puts(test);
getch();
}
```

//Dans cet exemple, la variable test2 reçoit en plus de son contenu initial, les 5 premiers caractères de test2, ce qui nous donne comme résultat "Filiere Info"

LA FONCTION strchr ET SES DERIVEES

1. *LA FONCTION strchr*
   Cette fonction retourne l'adresse(espace mémoire alloué à une variable) de la première occurrence (la première fois où le caractère est trouvé, en allant de gauche vers la droite) d'un caractère recherché. si le caractère recherché n'est pas trouvé, la fonction retourne une valeur nulle.
   Syntaxe :strchr(chaîne,caractère recherché)
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[]="Filiere";
char test2[]=" Informatique";
char *rech; //déclaration d'une variable qui récupérera l'adresse mémoire d'un caractère, on l'appelle pointeur
rech=strchr(test,'i'); //ici rech récupère l'adresse mémoire de la première occurrence de i dans la variable test. (c'est à dire l'adresse du premier i trouvé en partant de la gauche)
printf("%c",*rech); // Ici, printf affiche la valeur contenue dans l'adresse mémoire définie en paramètre( c'est à dire i).
getch();
}
```

2. *LA FONCTION strrchr*
   Cette fonction est identique à la précédente sauf qu'elle retourne l'adresse de la dernière occurrence du caractère recherché dans la chaîne de caractères.
3. *LA FONCTION strstr*
   Cette fonction retourne l'adresse de la première occurrence d'une sous-chaîne de caractères recherchée dans une chaîne. si la chaîne de caractères recherchée n'est pas trouvé, la fonction retourne une valeur nulle.
   Syntaxe :strstr(chaîne,sous_chaine)
   Exemple :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[]="bonjour et bonsoir";
char *rech; //déclaration d'une variable qui récupérera l'adresse mémoire d'un caractère, on l'appelle pointeur
rech=strstr(test,"bon"); //ici rech récupère l'adresse mémoire de la première occurrence de "bon"c'est à dire l'adresse du "bon" de "bonjour"
printf("%c",*rech); // Ici, printf affiche la valeur contenue dans l'adresse mémoire définie en paramètre. Elle va afficher 'b' car c'est le premier caractère de la sous-chaine trouvée dans l'adresse mémoire rech.
getch();
}
```

IV

EVALUATION SUR LES FONCTIONS DE MANIPULATION DE CHAINES DE CARACTERES

Exercice 1

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int l;
char val1[]="titre essai";
l=strlen(val1);
printf("%d",l);
getch();
}
```

Que retourne l à la fin du programme ?

- ○ 12
- ○ 11
- ○ 10

Exercice 2

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int l;
char val1[5]="titre";
l=strlen(val1);
printf("%d",l);
getch();
}
```

Que retourne l à la fin du programme ?

- ◯ Une erreur
- ◯ 5
- ◯ 4

Exercice 3

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int m;
char test[8]="trompe";
char test2[8]="trampe";
m=strcmp(test,test2);
printf("%d",m);
getch();
}
```

Que retourne m à la fin du programme ?

- ◯ 1
- ◯ 0
- ◯ -1

Exercice 4

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int m;
char test[8]="trompe";
char test2[8]="Trampe";
m=strnicmp(test,test2,2);
printf("%d",m);
getch();
}
```

Que retourne m à la fin du programme ?

- ◯ -1
- ◯ 1
- ◯ 0

Exercice 5

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[]="Chapitre1";
char test2[] ;
strncpy(test2,test,4);
puts(test2);
getch();
}
```

Que retourne test2 à la fin du programme ?

- ◯ Chapi
- ◯ Cha
- ◯ Chap

Exercice 6

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char test[]="Chapitre1";
char test2[] ;
strncpy(test2,test,4);
puts(test);
getch();
}
```

Que retourne test à la fin du programme ?

- ◯ Chap
- ◯ Chapitre1
- ◯ Cha