Chapitre 10: Les chaines de caractères

Introduction

- En langage C, il n'existe pas de véritable type chaîne, dans la mesure où l'on ne peut pas y déclarer des variables d'un tel type. En revanche, il existe une convention de représentation des chaînes. Celle-ci est utilisée à la fois :

✓ par le compilateur pour représenter les chaînes constantes (notées entre doubles quotes) ;

✓ par un certain nombre de fonctions qui permettent de réaliser :

  - les lectures ou écritures de chaînes ;
  - les traitements classiques tels que concaténation recopie, comparaison, extraction de sous-chaîne, conversions…

La convention adoptée

- En C, une chaîne de caractères est représentée par une suite d'octets correspondant à chacun de ses caractères (plus précisément à chacun de leurs codes), le tout étant terminé par un octet supplémentaire de code nul. Cela signifie que, d'une manière générale, une chaîne de n caractères occupe en mémoire un emplacement de n+1 octets.
- Cas des chaînes constantes : Une notation de la forme:

  "bonjour"

  est traduite par le compilateur en un pointeur de type

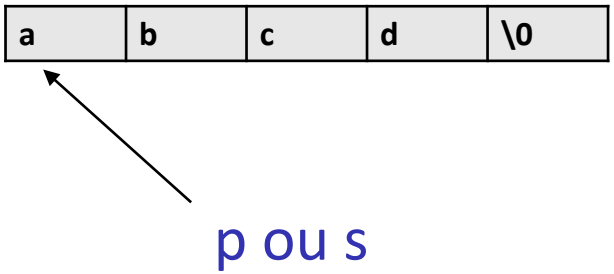
  char*

  sur la zone mémoire correspondante.

Déclaration des chaînes

- Deux manières de les définir :
  - ✓ par un tableau : `char s[ ] = "abcd";`
  - ✓ par un pointeur : `char* p = "abcd";`

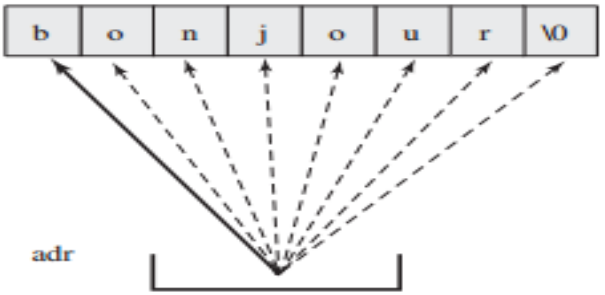
  Dans les deux cas le 0 final est rajouté automatiquement



- Attention : sizeof(s) = 5 : nombre de caractères + 0 final

Exemple : Calcul de la longueur

```
int longueur(const char* s)
   {
    char* adr = s;
    while (*adr)  adr++;
    return adr- s;   // nombre de char entre s et adr
    }
```



Initialisation de chaînes de caractères

- C autorise à initialiser un tableau de caractères à l'aide d'une chaîne constante. Ainsi, vous pourrez écrire :
  `char ch[20] = "bonjour" ;`
  ce qui sera équivalent à
  `char ch[20] = { 'b','o','n','j','o','u','r','\0' };`

- De plus nous pouvons écrire
  `char message[]= "bonjour " ;`

- Le langage C offre plusieurs possibilités de lecture ou d'écriture de chaînes :
  - ✓l'utilisation du code de format %s dans les fonctions printf et scanf ;
  - ✓les fonctions spécifiques de lecture (gets) ou d'affichage (puts) d'une chaîne (une seule à la fois).

Exemple

```
#include <stdio.h>
main()
{ char nom[20], prenom[20], ville[25] ;
  printf ("quelle est votre ville : ") ;
  gets (ville) ;
  printf ("donnez votre nom et votre prénom : ") ;
  scanf ("%s %s", nom, prenom) ;
  printf ("bonjour cher %s %s qui habitez à ", prenom, nom) ;
  puts (ville) ;
 }
```

- la délimitation de la chaîne lue ne s'effectue pas de la même façon avec scanf et gets. Plus précisément :
  - ✓avec le code %s de scanf, on utilise les délimiteurs habituels (l'espace ou la fin de ligne). Cela interdit donc la lecture d'une chaîne contenant des espaces. De plus, le caractère délimiteur n'est pas consommé : il reste disponible pour une prochaine lecture ;
  - ✓avec gets, seule la fin de ligne sert de délimiteur. De plus, contrairement à ce qui se produit avec scanf, ce caractère est effectivement consommé : il ne risque pas d'être pris en compte lors d'une nouvelle lecture.
- La fonction puts réalise un changement de ligne à la fin de l'affichage de la chaîne, ce qui n'est pas le cas de printf avec le code de format %s.

La fonction strlen

- La fonction strlen fournit en résultat la longueur d'une chaîne dont on lui a transmis l'adresse en argument:
- Syntaxe: int strlen(char *s1); (string.h)
- Le caractère de code nul n'étant pas pris en compte dans la longueur.
- Par exemple, l'expression :
  strlen ("bonjour"); vaudra 7 ;
- de même, avec :
  char * adr = "salut" ;
  l'expression :
  strlen (adr); vaudra 5

Les fonctions de concaténation: strcat

- L'appel de strcat se présente ainsi:
  strcat (but, source) (string.h)
  Cette fonction recopie la seconde chaîne (source) à la suite de la première (but), après en avoir effacé le caractère de fin.
- Il est nécessaire que l'emplacement réservé pour la première chaîne soit suffisant pour y recevoir la partie à lui concaténer.

Exemple

```
#include <stdio.h>
#include <string.h>
main()
{
   char ch1[50] = "bonjour" ;
   char * ch2 = " monsieur" ;
   printf ("avant : %s\n", ch1) ;
   strcat (ch1, ch2) ;
   printf ("après : %s", ch1) ;
}
```

```
Avant: bonjour
Apres : bonjour monsieur
```

Les fonctions de concaténation : strncat

- Syntaxe:
  strncat (but, source, lgmax) (string.h) travaille de façon semblable à strcat en offrant en outre un contrôle sur le nombre de caractères qui seront concaténés à la chaîne d'arrivée (but).

Exemple

```
#include <stdio.h>
#include <string.h>
main()
{
    char ch1[50] = "bonjour" ;
    char * ch2 = " monsieur" ;
    printf ("avant : %s\n", ch1) ;
    strncat (ch1, ch2, 6) ;
    printf ("après : %s", ch1) ;
}
```

Avant: bonjour
Apres : bonjour monsi

Les fonctions de comparaison : strcmp

- Syntaxe:

```
int strcmp(char *s1, char *s2);
```

- Compare lexico graphiquement les chaîne s1 et s2, et retourne une valeur:
  - ✓ =0 si s1 et s2 sont identique.
  - ✓ <0 si s1 précède s2.
  - ✓ >0 si s1 suit s2.

Exemple

- ❖ strcmp ("bonjour", "monsieur") est négatif
- ❖ strcmp ("paris2", "paris10") est positif

Les fonctions de comparaison: strcnmp

- Syntaxe:

  strncmp (chaîne1, chaîne2, lgmax);

  travaille comme strcmp mais elle limite la comparaison au nombre maximal de caractères indiqués par l'entier lgmax.

Exemple

- strncmp("bonjour", "bon", 4); est positif
- strncmp("bonjour", "bon", 2); vaut zéro.

Les fonctions de comparaison: stricmp et strnicmp

- Enfin, deux fonctions :

✓ stricmp (chaîne1, chaîne2) (string.h)

✓ strnicmp (chaîne1, chaîne2, lgmax) (string.h)

travaillent respectivement comme strcmp et strncmp, mais sans tenir compte de la différence entre majuscules et minuscules (pour les seuls caractères alphabétiques).

Les fonctions de copie : strcpy et strncpy

- La fonction :
  strcpy (but, source) (string.h)
  recopie la chaîne située à l'adresse source dans l'emplacement d'adresse but. Cette fonction fournit comme résultat l'adresse de la chaîne but.
- La fonction :
  strncpy (but, source, lgmax) (string.h)
  procède de manière analogue à strcpy, en limitant la recopie au nombre de caractères précisés par l'expression entière lgmax.

Les fonctions de copie: strcpy ey strncpy

Exemple

```
#include <stdio.h>
#include <string.h>
main()
{
   char ch1[20] = "xxxxxxxxxxxxxxxxxxx" ;
   char ch2[20] ;
   printf ("donnez un mot : ") ;
   gets (ch2) ;
   strncpy (ch1, ch2, 6) ;
   printf ("%s", ch1) ;
}
```

```
donnez un mot : bon
bon

_____
donnez un mot : bonjour
bonjouxxxxxxxxxxx
```

Les fonctions de recherche

- strchr (chaîne, caractère) (string.h)
  recherche, dans chaîne, la première position où apparaît le caractère mentionné.
- strrchr (chaîne, caractère) (string.h)
  réalise le même traitement que strchr, mais en explorant la chaîne concernée à partir de la fin.
- strstr (chaîne, sous-chaîne) (string.h)
  recherche, dans chaîne, la première occurrence complète de la sous-chaîne mentionnée.

N.B: Ces fonctions fournissent comme résultat un pointeur de type char * sur l'information cherchée en cas de succès, et le pointeur NULL dans le cas contraire.

Les fonctions de recherche

Exemple

```
int main(int argc, char *argv[])
{
char chaine[] = "Texte de test " ; char *suiteChaine=NULL;
suiteChaine = strchr(chaine, 'd');
if (suiteChaine != NULL)
{ printf("Voici la fin de la chaine a partir du premier d : %s",
                suiteChaine);}
   return 0;
 }
```

```
Voici la fin de la chaine a partir du premier d : de test
```

Conversion de chaines en valeur numérique

- Il existe trois fonctions permettant de convertir une chaîne de caractères en une valeur numérique de type int, long ou double :
  - ✓ atoi (chaîne) (stdlib.h) :
    fournit un résultat de type int.
  - ✓ atol (chaîne) (stdlib.h) :
    fournit un résultat de type long.
  - ✓ atof (chaîne) (stdlib.h)
    fournit un résultat de type double.

Conversion de valeur numérique en chaines

- Les fonctions

✓char *itoa(int n, char *s, int b);

✓char *ltoa(long n, char *s, int b);

✓char *ultoa(unisigned n, char *s, int b);

Convertissent l'entier n, représenté en base de numération b, dans la chaîne s.