

Langage C

Les chaines de caractères

L2 Mathématique et Informatique

Université de Marne-la-Vallée

Caractères en C, le type char

Le type char désigne les entiers écrits sur 1 octet.

- ▶ Le type `unsigned char` désigne les entiers de 0 à 255
- ▶ Le type `signed char` désigne les entiers de -128 à 127
- ▶ Suivant le compilateur, le type char peut désigner `unsigned char` ou `signed char`.
- ▶ Le code `ascii` est la partie de 0 à 128. Il ne permet pas de gérer les lettres accentuées.
- ▶ Les fonctions `scanf` et `printf` utilisent le spécificateur de format `%c`.
- ▶ Pour désigner une constante de type char on écrit la lettre entre quote 'z' ou on donne la valeur numérique de son code.

```
1 printf ("%c", 'a' );  
2 printf ("%c", 97); /* mais pourquoi faire !*
```

affiche tous les deux a

Lecture, écriture par caractère

Deux fonctions déclarées dans `stdio.h` permettent une manipulation caractère par caractère.

- ▶ Lecture: la fonction de prototype `int getchar(void);` renvoie la valeur d'un octet lu au clavier. En cas d'erreur ou d'arrêt d'écriture (Ctrl D sous Linux) elle renvoie la constante symbolique `EOF` (définie dans `stdio.h`) par `#define EOF (-1)`. Pour pouvoir prendre cette valeur, il faut se placer dans un ensemble de valeur plus grand que l'ensemble des `char`: les `int`.
- ▶ Écriture: la fonction de prototype `int putchar(int c);` écrit l'entier `c`, considéré comme un `unsigned char`. Elle renvoie le caractère écrit ou `EOF`

Opérations sur les caractères

Ce sont des *petits* entiers. On peut donc leur appliquer toutes les opérations sur les entiers.

Eviter de mélanger dans le code valeur numérique et notation entre quote. Les minuscules ont des codes successifs, ordonnés de 'a' à 'z'. Les majuscules ont des codes successifs, ordonnés de 'A' à 'Z'. Les chiffres ont des codes successifs, ordonnés de '0' à '9'.

Exemples d'utilisation

- ▶ tester si une lettre est une minuscule.

```
1  if ( 'a' <= lettre && lettre <= 'z' )
```

- ▶ transformer une minuscule en majuscule

```
1      if ( estMin ( lettre ) )  
2          lettre = lettre - 'a' + 'A' ;
```

De nombreuses fonctions (islower, toupper, ...) sont disponibles avec l'inclusion de <ctype.h>.

Les chaines de caractères

Pas de type défini, mais une convention.

Une chaîne de caractères est un tableau de caractères utilisant le marqueur de fin de chaîne `'\0'` (caractère de code 0, non affichable). Toutes les manipulations de chaîne de caractères utilisent cette convention.

Son absence entraîne des erreurs.

Chaîne constante : suite de caractères entre guillemets

`"bonjour tout le monde"`.

Définie à la compilation elle est placée dans la zone texte, elle n'est pas modifiable.

La chaîne constante vide `""` est un tableau constitué de l'unique caractère `'\0'`

Déclaration de variables

Une chaîne se déclare comme un tableau

- ▶ déclaration simple sans affectation

```
1      char phrase[TAILLE];
```

- ▶ déclaration-affectation
 - comme un tableau:

```
1      char hi[TAILLE]={ 'b', 'o', 'n', 'j', 'o', 'u', 'r', '\0' };
```

- avec une chaîne constante

```
1      char hello[TAILLE]="bonjour";  
2      char salut[]="bonsoir";
```

dans le premier cas il faut que $TAILLE \geq 8$; dans le deuxième le tableau salut aura la taille minimum nécessaire (8).

Dans les deux cas, la chaîne "bonsoir" est recopiée dans le tableau. C'est le seul endroit où un tableau peut être affecté directement.

Fonctions sur les chaînes

C'est au programmeur de s'assurer que les tableaux manipulés sont suffisamment grand.

- ▶ écriture avec `printf`: le spécificateur de format `"%s"` indique un chaîne de caractère. Les caractères sont affichés (jusqu'au premier `'\0'`).
- ▶ lecture avec `scanf`:
avec le spécificateur de format `"%s"` `scanf` de placer, à l'adresse fournies, tous les caractères lus jusqu'au premier séparateur. Le caractère `'\0'` est ensuite placé dans le tableau.

De nombreuses fonctions sont disponibles avec l'inclusion de `<string.h>`.

- ▶ Copie: `char* strcpy(char* dest, const char* src);`
- ▶ Concatène: `char* strcat(char* dest, const char* src);`
- ▶ Mesure: `size_t strlen(const char* s);`
- ▶ ...

Attention, aucun test sur la validité des chaînes:

- ▶ `src` doit contenir `'\0'`
- ▶ `dest` doit être assez grand.

La fonction `strlen` renvoie le nombre de caractère avant `'\0'`.
`size_t` est compatible avec le type `int`.

Arguments du main

On peut ajouter des informations à la ligne de commande au moment du lancement de l'exécutable `$> monProg -l toto 45`.

Ces informations seront alors transmises à la fonction `main` sous la forme de 2 paramètres (traditionnellement appelés `argc` et `argv`). Le premier est le nombre d'arguments de la ligne de commande, le second est un tableau de chaînes de caractères.

Lors de la définition de la fonction `main`, on déclare ces deux arguments `int main(int argc, char *argv[])`.

La fonction reçoit alors toujours au moins une chaîne de caractères : le nom de l'exécutable.

Dans l'exemple précédent:

