TD3: Chaînes de caractères

Novembre 2015

En C, il est possible d'initialiser un tableau de caractères par une chaîne de caractère de la manière suivante :

```
char c[] = "salut"
```

Le tableau se termine alors par le caractère de fin de chaîne `'\0'`. Sur l'exemple, il s'agit donc un tableau de taille 6. Cela permet l'usage de fonctions adaptées aux chaînes de carctères, fournies dans la librairie `string.h`, telle que la fonction `strlen()` qui donne la taille du tableau sans le caractère de fin de chaîne.

Exercice 1. Arguments de la ligne de commande

Une utilisation des chaînes de caractères est le passage d'argument au programme dans la ligne de commande. Pour cela, le prototype de la fonction `main()` doit être adapté de la manière suivante :

```
int main(int argc, char** argv)
```

où `argv` est un tableau de chaînes de caractères, et `argc est la taille du tableau`.

a. Tester et comprendre le programme suivant :

```
#include <stdio.h>
int main( int argc, char** argv){
  int i;
  for(i=0; i<argc; i++)
    printf("%s\n", argv[i]);
  return 0;
}
```

en l'appelant avec une ligne de commandes qui contient n'importe quel nombre d'arguments, par exemple :

```
./a.out 6 toto salut2016 ?!:;
```

b. Pourquoi la variable `argv` est-elle de type `char**` ?

c. En utilisant la fonction de conversion `atof()` (voir le manuel), écrire un programme qui affiche la somme des nombres passés en arguments. Par exemple la commande

```
./a.out 6.2 -8 2.0
```

doit afficher 0.2.

d. En utilisant la fonction `isdigit()` (voir le manuel), vérifier au préalable que les arguments passés sont bien des nombres flottants éventuellement négatifs.

Exercice 2. Tableau et liste de caractères

L'objectif est d'écrire des fonctions simples de manipulation des chaînes de caractères. Nous travaillerons avec deux chaînes de caractères qui seront passées en argument de l'appel au programme de la manière suivante :

```
./programme chaine1 chaine2
```

La première chaîne sera stockée de manière classique sous forme de tableau de caractère terminé par le caractère `'\0'`. La deuxième sera implémentée par une liste chaînée écrite dans la mémoire dynamique. Vous pouvez réutiliser/adapter les fonctions de manipulation des chaînes que vous avez vues.

a. Définir la structure de données `ListeC` permettant de stocker la chaîne de caractères sous forme de liste chaînée, et écrire une fonction d'ajout d'un caractère dans la liste. Ecrire la fonction `afficheChaine` qui affiche la liste chaînée, et la fonction

```
ListeC ajoutChaine(char* t, ListeC l)
```

qui écrit la chaîne de caractères `t` dans la liste `l` de manière *récursive*.

b. Écrire la fonction

```
void remplitMot(char** argv, int argc, char** t, ListeC* l)
```

qui affecte la première chaîne passée en argument du programme dans `t` et la deuxième dans `l`. Vous vérifierez au prélable que le nombre d'arguments du programme est correct, sinon le programme doit se terminer avec un message d'erreur.

c. Ecrivez une fonction *récursive* qui compare les deux chaînes (resp. `t` et `l`) selon l'ordre lexicographique : le comportement de la fonction imite celui de la fonction `strcmp()` (voir le manuel) pour la valeur de retour.

d. Ecrivez une fonction *recursive* qui teste si les deux chaînes (resp. `t` et `l`) sont symétriques l'une de l'autre.

e. Ecrire une fonction qui renvoie une variable de type `ListeC` qui contient le plus grand préfixe commun entre les deux chaînes.