PROGRAMMATION C
TP4

LICENCE MATHS-INFO
6 FÉVRIER 2012

L'ensemble des fichiers de ce TP seront placés dans un même répertoire et avec un unique `makefile`.

Vous écrirez les fonctions dans un fichier `lib_string.c`. Vous utiliserez le fichier `lib_string.h` et le programme de test `test_lib_string.c` (ne contenant qu'une fonction `main`) disponibles à `http://www.lif.univ-mrs.fr/~vemiya/?page=L2C_2011_2012`.

Rappels :

- une chaîne de caractères est un tableau d'éléments de type `char` terminé par le caractère `'\0'`. Dans toute cette partie, nous utiliserons la notation avec des pointeurs : une chaîne de caractères sera représentée par une variable de type `char *`.
- n'hésitez pas à vous reporter au poly en cas de doute sur les caractères et les chaînes.
- le type `char` est un type entier : on peut en particulier comparer deux variables de type `char` en utilisant les opérateurs `<,>,==` : le résultat est celui de la comparaison des entiers correspondants (en général le code dit ASCII). En particulier, les lettres majuscules (de A à Z) correspondent, dans l'ordre, aux entiers de 65 à 90 et les minuscules correspondent, dans l'ordre, aux entiers de 97 à 122. Une recherche sur internet vous donnera facilement l'ensemble des correspondances au besoin.

Longueur d'une chaîne. Ecrivez la fonction `int string_length(char *s)` qui détermine et renvoie le nombre de caractères d'une chaîne de caractères.

Miroir d'une chaîne. Ecrivez la fonction `void string_flip(char *s)` qui modifie son argument en son miroir : le miroir de `"les pointeurs"` est `"sruetniop sel"`.

Utilisation de gdb. Compilez votre programme en ajoutant l'option "-g" pour l'appel à `gcc` dans le `makefile`. L'utilisation de `gdb` est alors possible : lancez-la via la commande `gdb nom_executable`. Essayez alors les commandes suivantes :

- `help break` puis `help run`
- `break main` puis `break miroir` : quelles informations sont affichées ?
- `run` : qu'observez-vous ? Où en est l'exécution ?
- `step` : qu'observez-vous ? Où en est l'exécution ?
- `continue` : qu'observez-vous ? Où en est l'exécution ?
- `print nom_de_variable` pour une variable existante, autre qu'un pointeur, lorsque vous vous trouvez dans `string_flip` : qu'affiche la commande ?
- `print nom_de_pointeur` : qu'affiche la commande ? Expliquez.
- `print *nom_de_pointeur` : qu'affiche la commande ? Expliquez.

Vous enverrez les réponses dans un fichier texte joint lors à votre compte-rendu.

Comparaison de deux chaînes. L'ordre sur les chaînes étant l'ordre lexicographique habituel, écrivez la fonction `int string_cmp(char *s1, char *s2)` définie par :

$$\text{string\_cmp}(s_1, s_2) = \begin{cases} -1 & \text{si } s_1 < s_2 \\ 0 & \text{si } s_1 == s_2 \\ 1 & \text{si } s_1 > s_2 \end{cases}$$

Tableau de chaînes de caractères. On souhaite utiliser un tableau pour stocker plusieurs chaînes de caractères. Les longueurs des différentes chaînes de caractères ne sont pas les mêmes : on ne peut donc pas stocker une chaine directement dans une cellule du tableau. On utilisera une architecture comme celle illustrée sur la figure 1. Chaque cellule du tableau contiendra un pointeur, et sera donc de type `char *`.

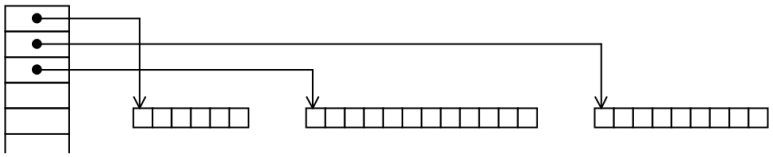

FIGURE 1. Tableau de chaînes de caractères.

Complétez la fonction `main` en déclarant le tableau avec le bon type et écrivez la fonction d'affichage du tableau `void print_tableau_strings(à-compléter t, int N)` qui affiche le tableau `t` de longueur `N` en question. L'appel à cette fonction doit donner :

```
Ramadane
Meriem
Charly
Mamadou
Thomas
Stephane
Aboubacar Sidy
etc.
```

Tri du tableau de chaînes de caractères. Ecrivez une fonction

`void tri_tableau_strings(à-compléter t, int N)`

qui modifie le tableau en triant les éléments par ordre lexicographique puis vérifiez que l'affichage du tableau trié donne :

```
Aboubacar Sidy
Brice
Charly
Clara
Dai Hung
Guillaume
Hadrien
etc.
```

Vous pouvez utiliser l'algorithme de tri de votre choix, par exemple le tri à bulle [1].

Une autre relation d'ordre. Définissez une fonction

```
int stringlength_cmp(char *s1, char *s2)
```

qui compare deux chaînes de caractères sur la base de leur longueur et non plus de leur ordre lexicographique. Si les deux chaînes sont de même longueur, alors on renverra le résultat de la comparaison par ordre lexicographique. Ecrivez ensuite une autre fonction de tri d'un tableau de chaînes de caractères via cette nouvelle relation d'ordre. Vérifiez que l'affichage du tableau trié donne :

```
Oleg
Yoan
Brice
Clara
Niass
Charly
Johnny
etc.
```

1. cf. la description `http://fr.wikipedia.org/wiki/Tri_a_bulles` et l'illustration par une vidéo `http://www.youtube.com/watch?v=gWkvvsJHbwY`.