I2010 : langage C (TP 5)

Rappel :

Quand la taille d'une zone mémoire est connue lors de l'écriture d'un programme, il est inutile de l'allouer dynamiquement. L'allocation dynamique ne se justifie que si la taille dépend d'une valeur qui ne sera connue qu'à l'exécution.

Astuces : Les fonctions de manipulation des chaînes de caractères

Pour lister les fonctions définies dans la librairie standard *string.h* accompagnées d'une courte description, entrez la commande : **`man 3 string`**

Pour lister l'ensemble des fonctions qui traitent les chaînes de caractères[1], entrez le commande : **`man 3 str [tab][tab]`**

Avant de résoudre les exercices de cette fiche, faites le petit test CodeRunner « **TP5 Les chaînes - Quiz** » sur Moodle.

1. Tableau des arguments du programme : `argv`

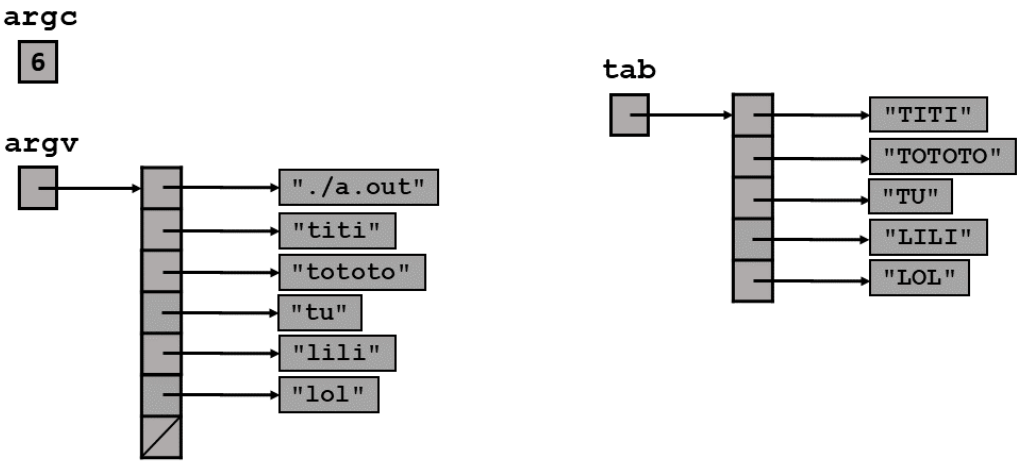
Ecrire un programme qui récupère les mots sur la ligne de commande et les copie dans un tableau de chaînes en majuscules. Une fois le traitement effectué, le programme affiche le contenu des deux tables en commençant par celle en majuscules.

Exemple d'appel:

```
./a.out titi tototo tu lili lol
TITI TOTOTO TU LILI LOL
titi tototo tu lili lol
```

Il s'agit bien ici d'utiliser dans un premier temps les paramètres `argc` et `argv` de la fonction `main` pour accéder aux arguments passés dans la ligne de commande et de les copier en majuscules dans un nouveau tableau `tab` qui aura le même type qu'`argv` (càd. un tableau de tableaux de caractères), comme l'illustre la figure suivante :

[1] Cette commande affiche toutes les pages de la section 3 du manuel `man` (consacrée aux librairies standards C) qui commencent par les lettres « str », abréviation de *string*.



2. Chaînes de caractères et fonction "fgets"

Ecrivez un programme qui recherche des mots parmi ceux passés en arguments au programme. Le traitement est le suivant :

- affiche les arguments du programme en les numérotant ;
- tant que la fin de fichier n'est pas rencontrée :
    - lit un mot sur *stdin* (pour info, "anticonstitutionnellement" [25 lettres] a cédé la place à "intergouvernementalisations" [27 lettres] en tant que mot le plus long de la langue française),
    - affiche le mot lu et sa longueur,
    - regarde si le mot est contenu dans `argv`,
    - affiche le mot lu suivi de « présent » ou « absent » selon le cas ;
- une fois l'exécution de la boucle terminée, affiche le nombre de fois qu'un mot lu n'a pas été trouvé dans `argv`.

Pour cet exercice, utilisez la fonction `fgets` pour lire un mot sur *stdin*.

Variante 1

Si la ligne entrée comporte plus de 27 caractères, prévenez l'utilisateur que son mot est trop long, videz le buffer de *stdin* (afin que les caractères de la ligne qui n'ont pas encore été lus soient effacés du buffer associé au clavier) et lisez un nouveau mot au clavier.

Variante 2

Modifiez votre programme afin que la recherche d'un mot soit insensible à la casse.

3. Chaînes de caractères: concaténation

Attention : *Ce programme est similaire au précédent. Cependant, il est suffisamment différent pour que cela vaille la peine de repartir d'un code vierge.*

Vous allez écrire un programme qui lit une suite de chaînes de caractères représentant des contenus de coffres forts. Elles sont structurées de la façon suivante :

1. les 5 premiers caractères représentent l'identifiant du coffre ;
2. les caractères suivants représentent le contenu du coffre.

Exemple:

```
AZ3ER Diamant 4K 1000 Euros Doudou d'enfance
```

L'application maintiendra une table de coffres qui sera gérée dynamiquement et à laquelle une taille logique et une taille physique seront associées. Chaque entrée de cette table contiendra donc une chaîne de caractères par coffre connu, composée de son **identifiant** suivi de son **contenu**.

Le programme fonctionnera de la façon suivante :

- tant que la fin de fichier n'est pas rencontrée :
  - lit une ligne d'au plus 254 caractères sur *stdin*,
  - parcourt la table des coffres afin de regarder si le coffre est déjà connu (cf. fonction `strncmp`) :
    - si non, il est inséré à la fin de la table en mettant à jour la taille logique et éventuellement la taille physique associée à celle-ci.
    - si oui, la partie de la nouvelle chaîne représentant le contenu est insérée dans le coffre existant à l'aide d'une simple concaténation (il peut donc y avoir plusieurs fois le même objet dans le coffre) (cf. function `strcat`).
- pour finir, la liste des coffres ainsi que leur contenu est affiché.

Exemple:

Input

```
ZRE43 pomme poire pèche
QS1SD boulon vis clous
az#fg copies d'examen
```

```
ZRE43 fraise yaourt pomme
az#fgPAS_MIS_D_ESPACES_EN_EXPRES
```

Output

```
ZRE43 pomme poire pèche fraise yaourt pomme
QS1SD boulon vis clous
az#fg copies d'examenPAS_MIS_D_ESPACES_EN_EXPRES
```

Notez que l'on ne se soucie pas de l'insertion d'espaces pour rendre le contenu plus agréable à lire : on effectue une « bête » concaténation. Par contre, le *RETURN* de fin de ligne doit être retiré de la chaîne lorsqu'un coffre est lu au clavier avec `fgets`.

Pour cet exercice, utilisez la fonction `strncmp` pour comparer uniquement les 5 premiers caractères de deux chaînes et la fonction `strcat` pour concaténer une *string* dans une autre.