POLYTECH NICE-SOPHIA

SI 3 - 2015-2016

Programmation Procédurale

Feuille 2

Encore des exercices simples en C

Les exercices de cette feuille sont simples. Leur but est de vous familiariser avec le langage C et le compilateur. La plupart de ces exercices prennent leurs données sur le fichier standard d'entrée et placent leurs résultats sur le fichier standard de sortie. Bien sûr, vous pouvez utiliser les redirections Unix si vous voulez travailler sur des vrais *fichiers.*

Exercice 1: Comptage

Ecrire un programme qui affiche le nombre d'occurences pour chacun des chiffres et des lettres qu'il a lu sur le fichier standard d'entrée.

Exercice 2: Une version simplifiée de wc

Ecrire un programme qui affiche le nombre de caractères, lignes et mots lus sur le fichier standard d'entrée. On considère ici, comme dans la commande `wc`, qu'un mot est une suite de caractères délimitée par les caractères SPACE, TAB, ou NEWLINE.

Exercice 3: Insertion dans un tableau

Écrire une fonction `insertion` qui insère une valeur dans un tableau d'entiers triés. Cette fonction prend en paramètres un tableau, le nombre d'éléments présents dans ce tableau et la valeur `v` à insérer. On suppose ici que les éléments du tableau sont triés par ordre croissant. La fonction `insertion` insère alors la valeur `v` à sa place.

Votre programme de test saisira une suite d'entiers sur le fichier standard d'entrée et les insérera successivement dans un tableau. A la fin de la saisie, le contenu du tableau sera affiché. Si tout se passe bien, les valeurs affichées seront triées, même si elle ont été saisies *dans le désordre*.

Exercice 4: Dump hexadécimal

Ecrire un programme permettant d'afficher le contenu du fichier standard d'entrée sur le fichier standard de sortie en hexadécimal et en caractères. Ce programme travaillera par lignes de 16 caractères. Pour chacune de ces lignes, on aura à gauche le code hexadécimal des caractères et à droite leur équivalent sous forme caractère (les caractères non imprimables étant représentés par le caractère '.').

Note: Le PC utilisent le jeu ASCII pour coder les caractères. Les caractères imprimables dans ce jeu sont ceux compris entre ' ' et '~'.

Exemple:

```
23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e #include <stdio.
68 3e 0a 0a 23 64 65 66 69 6e 65 20 4d 41 58 4c h>..#define MAXL
49 47 4e 45 20 38 0a 0a 76 6f 69 64 20 69 6d 70 IGNE 8..void imp
72 69 6d 65 72 5f 6c 69 67 6e 65 28 63 68 61 72 rimer_ligne(char
```

Exercice 5: Chaînes de caractères

Ecrire les fonctions suivantes:

```
void strcpy(char s1[], char s2[]);
int  strcmp(char s1[], char s2[]);
void strupper(char s[]);
```

où

strcpy recopie les caractères de la chaîne `s2` dans la chaîne `s1`.

strcmp compare les chaînes de caractères s1 et s2. Cette fonction renvoie:

0 si les deux chaînes sont identiques.

une valeur négative si $s1 < s2$

une valeur positive si $s1 > s2$

strupper convertit la chaîne de caractères en majuscules (Utiliser pour cela votre propre fonction de conversion).

Ecrire un programme permettant de tester ces fonctions.

Exercice 6: Une version simplifiée de fgrep

En tapant "`fgrep` *chaîne* `<`*fichier*" on obtient l'ensemble des lignes de *fichier* qui contiennent la séquence *chaîne*.

- Coder la fonction `int Strstr(char a[], char b[])` qui retourne 1 si la sous-chaîne `b` apparaît à l'intérieur de la chaîne `a`, ou 0 si `b` n'est pas une sous-chaîne de `a`.
- Utiliser cette fonction pour écrire un programme qui, étant donné une chaine en argument, affiche les lignes de son entrée qui la contiennent. Pour accéder au premier argument passé sur la ligne de commande, déclarez votre fonction `main` de la façon suivante:

  ```
  void main(int argc, char *argv[])
  ```

  Le premier paramètre de la ligne de commande se trouve dans `argv[1]`.