Travaux Pratiques de programmation n°2

Cours de Programmation C

—L2.1—

Le but de ce TP est d'écrire un programme `lexique` permettant d'établir la liste des mots d'un texte contenu dans un fichier, chaque mot étant associé à son nombre d'occurences. Pour cela, on se servira des tables de hachages vues en cours. Les sources seront organisées en trois fichiers.

Dans toute la suite, on désigne par *mot* une chaîne de caractères lue grâce au format %s de scanf.

Les mots seront rangés dans une table de listes de mots de taille MAX. On pourra tester différentes valeurs de MAX.

On utilisera ainsi les types

```
typedef struct cellule{
    char *mot;
    int occurence;
    struct cellule *suivant;
} CelluleMot, *ListeMot;

typedef ListeMot TableHachage[MAX];
```

▸ **Exercice 1.** *Gestion des listes*

Créer les fichiers `liste.c` *et* `liste.h` *qui seront dédiés à la gestion de listes ordonnées de mots avec comptage des occurences.* `liste.h` *contiendra la déclaration des types* `CelluleMot` *et* `ListeMot`.

Écrire les fonctions d'allocation, d'affichage et d'ajout ordonné d'un mot dans une liste. Pour l'ajout, si une cellule est déjà présente, on incrémentera le nombre d'occurences (on utilisera la fonction strcmp pour comparer deux mots).

▸ **Exercice 2.** *Hachage*

Créer les fichier `hachage.c` *et* `hachage.h` *qui serviront à la gestion de la table de hachage.* `hachage.h` *contiendra la déclaration du type* `TableHachage`.

1. *Écrire deux fonctions de hachage pour les chaines de caractères :*
   - *Somme des codes ASCII des lettres*
   - *Somme des produits du code ASCII de la lettre par sa position dans le mot (i.e.* $\sum_{i=0}^{l-1}(i+1) * m[i]$ *où* l *est la longueur du mot* m*).*

2. *Écrire une fonction* `int AjouteMot(char* mot, TableHachage T)` *qui ajoute le mot* `mot` *à la table* `T`.
3. *Écrire une fonction* `InfoTable` *qui affiche le nombre de listes non vides, la taille maximum atteinte et la moyenne des tailles des listes non vides.*
4. *Écrire une fonction* `VideTable` *qui vide la table de hachage et retourne la liste des mots de la tables (la liste n'a pas besoin d'être ordonnée).*
5. *La fonction suivante est souvent utilisée pour le hachage des chaines caractères, elle utilise des notions qui seront abordées plus tard dans le cours. Copiez-la et Ajoutez-la à votre programme :*

```
unsigned  hash(char *p) {

        unsigned  h=0,g;

        for(;*p;p++)
        {
            h = (h<<4)+*p;
            if(g=h&0xf0000000) /*il y a des valeurs
            dans les 4 bits de poids fort*/
            {
                h = h^(g>>24); /*Elles vont influencer le haut
                 de l'octet de poids faible */
                h = h^g; /* on les supprime du haut de h */
            }
        }

        return h;
}
```

▸ **Exercice 3.** *Lecture / Écriture en fichier*

Créer un fichier `main.c` *qui contiendra la fonction main de votre programme.*

1. *Écrire une fonction* `ListeMot LitFichier(FILE *f)` *qui prend en paramètre un pointeur sur un fichier et utilise une table de hachage pour créer la liste ordonée avec nombre d'occurence de ses mots. Utilisez la fonction d'affichage de liste écrite dans* `liste.c` *pour tester votre fonction.*
2. *Dans le fichier* `liste.c`*, écrivez une fonction* `void EcritListe(ListeMot liste, FILE *f)` *qui écrit dans un fichier la liste des mots avec leur nombre d'occurences.*
3. *Ecrivez une fonction* `main` *qui lit dans les arguments de la ligne de commande un nom de fichier* `source` *et un nom de fichier* `cible` *et qui enregistre dans le fichier cible la liste des mots du fichier source, si aucun fichier* `cible` *n'est donné, on affichera le résultat sur la sortie standard.*

   Exemple :

Affichage sur la sortie standard : `lexique cleve`

Affichage dans un fichier `cleve.l` *:* `lexique cleve cleve.l`

Le fichier de test `cleve` *peut-être téléchargé à cette adresse : http ://www-igm.univ-mlv.fr/ pons/ens/2011-L2/*

4. *Que fait la commande :* `lexique cleve |sort > cleve.l` *?*
5. *Modifier votre programme pour afficher les informations de* `InfoTable` *et effectuer des tests avec vos différentes fonctions de hachage.*