Langace C - Travaux Dirigés n° 9

— Licence Informatique —

Le but de cette séance de Td est d'améliorer le programme de la séance précédente en utilisant du hachage. On va utiliser du hachage ouvert où les éléments qui ont même valeur de hachage sont stockés dans des listes. On reprend donc les types `Cellule` et `Liste` définis de la façon suivante.

```
// Type des cellules pour les listes chaînées
struct Cellule {
  char* str;                    // Pointeur sur la chaîne
  int nbr;                      // Nombre d'occurrences
  Cellule *next;                // Pointeur sur la cellule suivante
};
typedef Cellule* Liste;
```

On définit alors un type pour une table de hachage.

```
// Type pour les tables de hachage
struct HashTable {
  int mod;                      // Nombre d'éléments de la table
  Liste *table;                 // Pointeur sur la table allouée
};
```

▶ **Exercice 1 (Fonction de hachage)**
Une chaîne de caractères est vue comme un entier écrit en base 256. Chaque caractère est donc un chiffre de cet entier. Le premier caractère de la chaîne est le chiffre de poids fort. La valeur de hachage de la chaîne est la valeur de cet entier modulo la taille de la table. Écrire une fonction `hash` qui calcule cette valeur de hachage. Cette fonction prendra en paramètre la chaîne et la taille de la table.

▶ **Exercice 2** Écrire une fonction `hashTableInit` qui initialise une table de hachage. Cette fonction prend en paramètre la table de hachage à initialiser et la taille.

▶ **Exercice 3** Écrire une fonction `hashTablePrint` qui affiche une table de hachage. Pour chaque valeur de hachage h pour laquelle la liste correspondante n'est pas vide, cette fonction affiche h puis la liste.

▶ **Exercice 4** Écrire un programme global qui accepte des noms de fichiers sur la ligne de commande et qui lit les fichiers les uns après les autres. Si aucun nom de fichier n'est présent sur la ligne de commande, le programme lit sur l'entrée standard. À la fin, le programme affiche la table de hachage qui contient toutes les lignes qui apparaissent dans le fichiers avec leur nombre d'occurrences.

Une table de hachage est intéressante lorsque les listes associées à chaque valeur de hachage soient courtes. La recherche d'un élément ne parcourt qu'une courte liste et se fait donc rapidement. Si la table contient trop d'éléments, les listes s'allongent et la recherche perd de son efficacité.

▶ **Exercice 5** Écrire une fonction `hashTableResize` qui redimensionne une table de hachage. Cette fonction prendra comme paramètre la nouvelle taille de la table.

Au départ, les tables de hachage sont de taille 701. Lorsque la table contient trop d'éléments, la table devient de taille 5007. Ensuite elle peut devenir de taille 50021. Ces valeur seront mises dans un tableau afin de pouvoir être configurées facilement.

▶ **Exercice 6** Écrire une fonction `hashTableUpdate` qui augmente la taille d'une table de hachage si elle contient trop d'éléments. La table contient alors le nombre totale de chaînes qu'elle contient. Il est décidé de changer la taille d'une table lorsqu'en moyenne, elle contient plus de 10 éléments par valeur de hachage.