Cours de programmation séquentielle

Table de hachage

1 Buts

Dans ce travail vous verrez les concepts suivants:

- Créer et utiliser une librairie de table de hachage.
- Utilisation des pointeurs pour la liste chaînée.
- Utilisation de tableaux de pointeurs.
- Création de fonction de hachage simple.

Vous avez vu la théorie sur les tables de hachages au cours d'algorithmique.

2 Énoncé

Dans ce travail pratique il s'agit d'implémenter une table de hachage en utilisant la méthode du chaînage.

La structure `hm_t` (pour *hashmap*) sera composée comme suit

```
struct hm_t {
    struct entry_t **entries;
    int length;
};
```

Les entrées de la table seront un tableau de liste chaînée dont chaque noeud sera une paire *clé-valeur*, de type `entry_t` (la clé et la valeur sont des chaînes de caractères)

```
struct entry_t {
    char *key;
    char *value;
    struct entry_t *next;
};
```

Les fonctionnalités de votre table de hachage devront être les suivantes:

```
// création d'un pointeur vers une hm
hm_t *hm_create(unsigned int length);
// destruction de la hm et libération de la mémoire
void hm_destroy(hm_t **hm);

// insère la paire key-value dans la hm. si key est déjà présente
// écraser value dans la hm.
```

```
hm_t *hm_set(hm_t *hm, const char *const key, const char *const value);
// retourne la valeur associé à la clé, key
char *hm_get(const hm_t *const hm, const char *const key);
// retire key de la hm et retourne la valeur
char *hm_rm(hm_t *hm, const char *const key);

// convertit la hm en chaîne de caractères
// dans le but de l'afficher ensuite
char *hm_to_pretty_str(const hm_t *const hm);
// affiche le contenu de la hm
void hm_pretty_print(const hm_t *const hm);
// sauvegarde les pairs clé-valeur contenues de la table de hachage dans un fichier texte
// au format <key>;<value>
void hm_save(const hm_t *const hm, const char *const filename);
// charge les pairs clé-valeur dans une table de hachage depuis les
// lignes d'un fichier texte dont le format est <key>;<value>
hm_t * hm_load(const char *const filename);
```

Remarque 1 *(Utilitaires)*

Vous devrez probablement écrire d'autres fonctions "utilitaires" dans votre librairie mais les fonctions ci-dessus sont celles que vous exposerez à l'utilisatrice (p.ex. toutes les fonctions pour créer des `entry_t` et les détruire, ou encore vérifier si la table de hachage possède déjà une clé donnée).

En particulier, il y a la fonction de hachage. Nous vous suggérons d'utiliser la fonction

```
val = 0;
for (int i = 0; i < strlen(c); ++i) {
    val = (43 * val + c[i]) % length;
}
return (val % length);
```

Mais vous êtes libres d'en utiliser une autre si vous voulez (n'importe laquelle de celles vues en cours par exemple, mais évitez les solutions trop compliquées dans un premier temps).

Remarque 2 *(Collisions)*

Dans cette implémentation les collisions sont gérées par chaînage: lorsqu'une clé génère un hash déjà présent dans la table, on insère la clé dans l'alvéole à l'aide d'une liste chaînée. Vous n'utiliserez pas ici d'adressage tel que le sondage linéaire ou le double hachage.

2.1 Programme

Une fois votre librairie de table de hachage écrite et testée, vous devrez également écrire un programme d'annuaire qui permet de stocker des noms d'étudiants ainsi que des numéros de téléphone (tout cela sous forme de chaînes de caractères).

2.1.1 Cahier des charges

- L'annuaire est prévu pour 66 étudiants au départ, mais il faut pouvoir adapter cette valeur en cours d'exécution. Si le nombre d'étudiants augmente, la taille de la table doit pouvoir augmenter. (En général au double la taille de la table quand on a un taux de remplissage de 75%). Pour ce faire, on détruit l'ancienne table et on en recrée une nouvelle (il faut transférer tout l'annuaire et donc tout re-hacher).
- Afin de vous aider dans votre tâche, un fichier students.txt vous est fourni (il est également accessible sur Cyberlearn). Les numéros de téléphone sont sans indicatifs/espaces. Le format est `<prénom-nom>;<numéro de téléphone>`.
- Il faut prévoir une fonction qui permet de charger le fichier `students.txt` et qui permet de sauver la table dans un fichier texte au même format.
- Il faut pouvoir ajouter et supprimer manuellement des étudiants.
- Il faut pouvoir interroger l'annuaire par nom ou par numéro de téléphone (il faut donc deux tables).