CS315 TP1 et 2
Hachage à adressage ouvert

Antoine Bouthors
Antoine.Bouthors@imag.fr

13 novembre 2006

Instructions

Lisez le sujet en entier avant de commencer !
Le but du jeu est de développer un programme permettant la gestion d'une base de données via un hachage à adressage ouvert, stockée dans un fichier, sur deux scéances de TP.
Respectez scrupuleusement les spécifications demandées. En effet, l'évaluation du TP se fera par des tests unitaires, et il faut pour cela non seulement que vos fonctions se comportent comme attendu, mais aussi qu'elles aient exactement les prototypes demandés.
N'oubliez pas non plus de tester votre programme à chaque étape, en essayant de couvrir le plus de cas possibles.
Vous devrez rendre, une semaine après la fin de la dernière séance, le fichier annuaire.c contenant le corps des fonctions développées, et optionellement un fichier `main.c` contenant votre programme de test. De plus, un compte-rendu vous est demandé, dans lequel vous expliquerez l'algorithme de chacune des fonctions, ainsi que les difficultés rencontrées et solutions trouvées, et les modifications apportées au code de base avec leurs justifications. Le tout à envoyer par email.

1 Présentation

On désire réaliser un annuaire, en utilisant les couples (nom,prénom) comme clés d'indexage. Toutes les données, y compris les information sur la fonction de hachage et la taille de la table, sont stockées dans un fichier appelé data.db.
Le programme devra pouvoir insérer, rechercher, afficher et supprimer les éléments de l'annuaire.

1.1 Code fourni

Les fichiers annuaire.h et annuaire.c contiennent les prototypes des fonctions que vous aurez à écrire, ainsi que le corps de quelques-unes. Utilisez-les, et modifiez-les au besoin.

1.2 Structure du fichier

Le fichier se présente comme ceci :

En-tête	Entrée 0	Entrée 1	...	Entrée $m-1$

L'en-tête est constitué de la structure `Params`. Chaque entrée est une structure `Entry`. Le fichier est stocké en binaire.

2 Fonction utilitaires

Les fonctions `init()`, `create_file()`, `print_entry()` et `get_params()` sont déjà écrites et peuvent vous être utiles. Servez-vous en et inspirez-vous en au besoin. Les variables `NullEntry`, `DeletedEntry`, et `EmptyEntry` correspondent respectivement à une entrée nulle, supprimée et vide. Vous aurez à vous servir de ces valeurs au long du TP.
La fonction `compare_keys()` permet de comparer deux clés. Implémentez-la.

3 Fonction de hachage

Afin d'avoir un adressage efficace, nous décidons d'utiliser une fonction de hachage double : la fonction h de hachage dépend de deux fonctions auxilliaires h_1 et h_2 :

$$h(k,i) = (h_1(k) + ih_2(k)) \mod m$$

Avec m la taille de la table de hachage. On choisit pour h_1 et h_2 :

$$h_1(k) = k \mod m$$
$$h_2(k) = 1 + (k \mod m')$$

Avec par exemple m premier et $m' = m - 1$.
Implémentez la fonction de hachage `hash_key()`. Vérifiez son fonctionnement en comparant vos résultats avec les autres binômes.
Question 1 : Pour quelles valeurs maximales de m et m' votre implémentation est-elle valide ?

4 Fonction d'insertion

La fonction d'insertion `insert_entry()` commence par ouvrir le fichier pour y lire les paramètres de hachage, puis recherche la position d'insertion en utilisant la fonction de hachage. Elle insère ensuite les données dans le fichier, si cela est possible, ou retourne une erreur dans le cas contraire.
Implémentez cette fonction d'insertion, en utilisant `hash_key()` et `compare_keys()`.

5 Affichage de la table

Afin de vérifier la validité de votre programme, il peut être utile d'afficher le contenu de la table. Implémentez pour cela la fonctions `dump_table()`, qui affichera le contenu de toutes les entrées non nulles en utilisant `print_entry()`.

6 Fonction de recherche

Implémentez maintenant la fonction de recherche `search_entry()`, qui prend en entrée la clé à chercher et retourne soit l'enregistrement trouvé, soit une erreur. De même que la fonction d'insertion, `search_entry()` commence par ouvrir le fichier et y lire les paramètres de hachage avant d'effectuer la recherche, et se sert de `hash_key()` et `compare_keys()`.

7 Fonction de délétion

La fonction de délétion `delete_entry()` doit rechercher l'entrée à supprimer à partir de sa clé, et remplacer l'entrée par la valeur spéciale `DeletedEntry`. Il faudra alors modifier la fonction d'insertion afin de gérer les entrées supprimées.

8 Programme final

Vous pouvez, afin de tester vos fonction, faire un programme interactif de manipulation de la base de données.
Vérifiez la validité de vos algorithme en échangeant vos fichiers de données entre binômes.