Structure de données

TP7 : Listes associatives et tables de hachage

On appelle structure de données associative une structure de données qui permet d'enregistrer et rechercher des données en fonction d'une clé identifiant de manière unique une donnée. Un exemple classique d'une telle structure de données est le dictionnaire. La clé est un mot et la donnée sa définition. Un autre exemple est un annuaire où la clé est le nom d'un abonné et la donnée les informations le concernant (numéro de téléphone, adresse, etc.).

L'objectif de ces exercices est tout d'abord de spécifier les opérations attendues sur une structure de données associative (exercice 1). Ensuite nous étudierons une première structure de données associative simple s'appuyant sur des éléments chaînés (exercice 2). Enfin, nous envisagerons une solution pour améliorer la recherche d'un élément dans la structure en introduisant la notion de table de hachage (exercice 3).

Remarque : Pour éviter de manipuler des noms un peu longs, il est possible (et conseillé) d'utiliser les abréviations introduites dans le sujet.

Exercice 1 : Spécification des opérations sur une structure de données associative

Nous allons commencer par spécifier les opérations d'une structure de données associative indépendamment de sa représentation. On suppose qu'il existe un type appelé SDA (Structure de Données Associative).

1.1. Pour simplifier nous considérerons que la clé est une chaîne de caractères de longueur variable et la donnée est un entier.

Définir les types Clé et Donnée dans un fichier d'entête `types.h`.

1.2. Une structure de données associative propose les opérations suivantes :

- initialiser une SDA. La SDA ainsi initialisée est vide ;
- indiquer si une SDA est vide ou non ;
- enregistrer une donnée et la clé correspondante. La clé ne doit pas avoir déjà été utilisée pour enregistrer une donnée ;
- indiquer si une clé est utilisée dans une SDA. Une clé est utilisée par une SDA si une donnée a été enregistrée avec cette clé ;
- récupérer la donnée associée à une clé. On considère que cette clé doit nécessairement être une clé utilisée dans la SDA ;
- vider une SDA, c'est-à-dire supprimer toutes les informations enregistrées dans la SDA.

Écrire la spécification des opérations d'une SDA. On fera apparaître la signature des sous-programmes correspondants en précisant le mode de passage des paramètres (in, out ou in out), le descriptif de l'opération ainsi que ses préconditions et postconditions.

On écrira ces opérations sous la forme d'un module d'entête.

Exercice 2 : Liste chaînée associative

Nous nous intéressons ici à une première implantation d'une structure de données associative : la liste chaînée associative (LCA). Elle s'appuie sur des éléments chaînés appelés cellules. Une cellule est composée d'une clé, d'une donnée et d'un accès vers une autre cellule. La liste chaînée associative est alors définie comme un pointeur sur une cellule correspondant à la première cellule de la structure de données.

2.1. Donner les types nécessaires pour définir une LCA et en particulier le type LCA.

2.2. Donner le code en langage C des opérations de la LCA. Les opérations sont celles identifiées dans l'exercice 1 pour lesquelles, il suffit de remplacer le type SDA par LCA. Pour les différencier, on utilisera le suffixe `_lca` pour chaque opération.

Remarque : On impose que le temps d'enregistrement d'une information soit constant, c'est-à-dire que quel que soit le couple clé/donnée, le temps d'exécution doit être identique.

2.3. Écrire un programme de test minimal qui insère successivement dans une même LCA les couples clé/donnée ("un", 1) et ("deux", 2). On dessinera la représentation de la LCA après l'exécution de chacune des opérations qui la manipule.

Exercice 3 : Table de hachage

Savoir si une clé est utilisée dans une liste chaînée associative ou rechercher un élément sont des opérations coûteuses en temps d'exécution car, dans le pire des cas, il faut parcourir toutes les cellules de la LCA. Ceci est en particulier le cas si une clé n'est pas utilisée dans la LCA. Nous allons nous intéresser à une autre implantation de la SDA qui améliore le temps d'exécution de ces deux opérations. Il s'agit des tables de hachage (TH).

L'idée est d'utiliser un tableau (donc un accès direct) pour stocker les données et d'utiliser une fonction, appelée fonction de hachage, qui à partir d'une clé calcule la position de la donnée dans le tableau. On peut par exemple prendre pour fonction de hachage, la fonction qui renvoie le nombre de caractères d'une chaîne de caractères. La valeur calculée par une fonction de hachage est appelée valeur de hachage.

La figure 1 présente une table de hachage dans laquelle nous avons utilisé pour valeur de hachage le nombre de caractères de la clé. Ainsi la donnée 1 ayant pour clé "un" sera mise à l'indice 2 de la table (donc en 3e position). De même 2 est à l'indice 4, 3 à l'indice 5 et 4 à

l’indice 6 car ils ont respectivement pour clé "deux", "trois" et "quatre", soit 4, 5 et 6 caractères. L’accès à une donnée est alors immédiat une fois la valeur de hachage calculée.

		1		2	3	4				

FIGURE 1 – Table de hachage dans laquelle ont été mis les couples ("un", 1), ("deux", 2), ("trois", 3) et ("quatre", 4), la fonction de hachage étant le nombre de caractères de la clé.

Cependant, deux problèmes peuvent se poser. Tout d’abord, il se peut que la valeur de hachage soit trop grande par rapport à la taille de la table de hachage. Dans ce cas, il suffit d’utiliser le modulo pour que cette valeur constitue un indice valide pour la table. Ainsi, pour enregistrer 99 avec la clé "quatre-vingt-dix-neuf", la valeur de hachage est 21 donc supérieure à 11, taille de notre table. On prendra donc l’indice 10.

Le deuxième problème est que deux clés différentes peuvent avoir la même valeur de hachage. Ceci signifie qu’il faut alors les ranger dans la même case du tableau ! C’est par exemple le cas du couple ("cinq", 5). La valeur de hachage est 4 mais la case correspondante de la table est déjà occupée par la donnée 2. Une solution consiste donc à considérer que chaque case du tableau n’est pas une simple donnée mais une liste chaînée associative (exercice 2). Ainsi les deux données 2 et 5 seront rangées dans cette liste.

3.1. *Structure de la table hachage*. Dessiner la représentation en mémoire de la table de hachage qui contient les couples clé/donnée ("un", 1), ("deux", 2), ("trois", 3), ("quatre", 4), ("cinq", 5), ("quatre-vingt-dix-neuf", 99) et ("vingt-et-un", 21) sachant que la taille de la table est 11 et la valeur de hachage est le nombre de caractères de la clé.

3.2. *Fonctions de hachage.* Nous commençons par nous intéresser à quelques fonctions de hachage particulières.

3.2.1. Écrire une fonction de hachage qui renvoie le nombre de caractères de la clé. On n’utilisera aucune fonction de la bibliothèque C mais on indiquera celle que l’on aurait pu utiliser.

3.2.2. Écrire une fonction de hachage qui renvoie le code ASCII de l’initiale de la clé.

On rappelle qu’en C, le type *char* est considéré comme un type entier et contient le code ASCII du caractère.

Remarque : C’est cette fonction de hachage qui est généralement utilisée sur un répertoire téléphonique « papier ».

3.2.3. Écrire une fonction de hachage qui renvoie la somme des codes ASCII des caractères de la clé.

3.3. Définir le type TH (table de hachage) en considérant que la taille de la table est arbitrairement fixée à 11.

3.4. Écrire le code des opérations qui consistent à :

- initialiser la table de hachage ;
- la vider ;
- enregistrer une donnée avec sa clé ;
- savoir si elle est vide ou non.

Les autres opérations de la spécification (exercice 1) ne seront pas implantées. On utilisera pour valeur de hachage, le nombre caractères de la clé (question 3.2).