
Ce TP se déroule sur deux séances (quatre heures). Les étudiants qui se sentent en difficulté devraient réaliser la section 1, uniquement. Les autres peuvent réaliser la section 2 aussi. Des structures de données introduites dans les premiers TP sont réutilisées.

1 Petits exercices sur les tables de hachage

1.1 Double hachage

Question 1. Implanter un module de table de hachage d'entiers, à partir des déclarations suivantes. La technique utilisée est le double hachage. Prendre comme fonction de hachage :

$$h(a) = (h_1(a)\,,\ h_2(a)) = (a \bmod N\,,\quad 1 + (a \bmod (N-1)))\,.$$

Écrire un programme principal, comportant une boucle (boucle A), qui lise une suite d'entiers terminée par -1 et qui les enregistre dans la table. À la fin de la boucle, imprimer la table.

```
#define N 11
struct valeur_hachage {
    int h1;
    int h2;
};
typedef struct valeur_hachage fonction_hachage (int);
enum etat_alveole { libre, occupe };
struct alveole {
    enum etat_alveole etat;
    int valeur;
};
struct table_hachage {
    struct alveole tab [N];
    fonction_hachage* h;
    int n;
};
```

Question 2. Compléter le module en lui ajoutant une fonction qui recherche si un élément est présent dans la table. Dans le programme principal, ajouter une deuxième boucle (boucle B), qui lise une suite d'entiers terminée par -1, et qui indique, pour chacun d'eux, s'il est ou non présent dans la table.

Question 3. Compléter le module en lui ajoutant une fonction qui supprime un élément de la table. Adapter éventuellement les types proposés plus haut. Dans le programme principal, insérer une troisième boucle, entre les boucles A et B, qui lise une suite d'entiers terminée par -1, et qui supprime chacun d'eux de la table.

1.2 Simple hachage

Dans cette sous-section, réutiliser les listes chaînées introduites au premier TP.

Question 4. Adapter le module précédent pour qu'il s'appuie sur une technique de simple hachage. Ranger les entiers ayant même valeur de hachage dans des listes chaînées. Supprimer les champs et les déclarations inutiles. Utiliser la fonction de hachage suivante :

$$h(a) = a \bmod N\,.$$

Question 5. Écrire une fonction qui permette de visualiser la table de hachage de la question précédente, en utilisant le logiciel `dot`. Par exemple, les commandes

```
$ dot -Tpdf hachage.dot -Grankdir=LR -o hachage.pdf
$ evince hachage.pdf
```

produisent l'affichage de la Figure 1 à partir du fichier `hachage.dot` suivant :

```
digraph G {
    tableau [shape=record,label="<f0> 0 | <f1> 1 | <f2> 2 | <f3> 3 | <f4> 4 |
<f5> 5 | <f6> 6 | <f7> 7 | <f8> 8 | <f9> 9 | <f10> 10"];
    tableau:f1 -> 45;
    45 -> 56;
    tableau:f6 -> 17;
    17 -> 6;
    tableau:f8 -> 8;
    tableau:f9 -> 9;
};
```

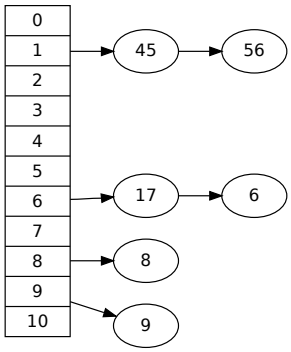

FIGURE 1 – Une table de hachage

Question 6. Reprendre la section 1, en réalisant une table de hachage de chaînes de caractères. Réutiliser le type `chaine` introduit au TP 1.

Toute la difficulté tient au fait que le type `chaine` est muni d'un constructeur et d'un destructeur (ce qui n'est pas le cas des types `int`, `double` et `char`).

Vérifier avec `valgrind` que vos programmes ne perdent pas de mémoire.