Travaux dirigés #6 — Hash Map

L'objectif de ce TP est de fournir un `template` pour l'implémentation des tables de hachage en C++ [1]. Les tables de hachage sont des structures de type *tableau associatif*, permettant de stocker des correspondances `clé-valeur`. Une table de hachage utilise une fonction de hachage pour calculer un indice parmi un ensemble d'emplacements possibles, à partir duquel la valeur correspondante peut être associée.

Dans le meilleur des mondes, la fonction de hachage calcule un *unique* indice pour chaque `clé`. La conception de telles fonctions de hachage étant particulièrement difficile, la majorité des tables de hachage autorise des **collisions**. Il est demandé d'implémenter une méthode de résolution des conflits simple, appelée **chaînage** et décrite ci-après.

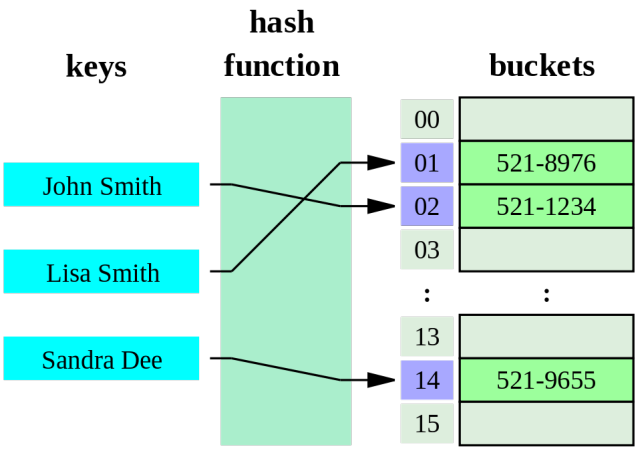

FIGURE 1 – Illustration d'une table de hachage. © Wikipedia.

Les tables de hachage

Les tables de hachage sont en fait une implémentation du type abstrait *tableau associatif* (association `clé-valeur`). Pour ce TP, il n'est cependant pas imposé d'implémenter une classe abstraite représentant les tableaux associatifs.

La structure demandée est la suivante : implémenter un modèle de classe permettant de gérer une table de hachage. Les **paramètres de modèle** doivent comprendre le type de la `clé` ainsi que le type de la `valeur`.

Et ?

Une autre propriété des tables de hachage mériterait-elle de figurer en paramètre de modèle du `template` ?

Dans un premier temps, votre modèle de classe contiendra un constructeur sans arguments, un destructeur, et les méthodes `bool acces(const &clé, ?)`, `inserer(const &clé, const &value)` et `supprimer(const &clé)`.

La fonction de hachage. Par souci de simplicité, l'implémentation de la fonction de hachage sera unique et dépendra du nombre d'emplacements disponibles. Concrètement, les emplacements sont représentés par un tableau de n éléments, et pour une `clé` donnée, la fonction de hachage retourne :

$$cle \ mod(n)$$

Les éléments contenus dans le tableau seront eux de type `Emplacement`. Ce modèle de classe devra contenir une `clé`, une `valeur` ainsi qu'un pointeur sur l'élément suivant, utilisé en cas de collision.

1. Le type `map` de la `STL` est-il une table de hachage ? La réponse est par ici.

L'adressage par chaînage. Pour résoudre les conflits lorsque deux clés sont hachées sur le même emplacement, le pointeur sur l'élément suivant de la classe `Emplacement` sera utilisé. Au moment de l'ajout d'une paire `clé-valeur` dans la table de hachage, si la valeur retournée par la fonction de hachage contient déjà des éléments, il faut parcourir ces derniers pour se mettre à la bonne position. Deux situations sont possibles :

— La `clé` est déjà présente dans la table de hachage : il faut mettre sa `valeur` à jour.
— La `clé` n'est pas encore présente : il faut l'ajouter à la table de hachage.

La Figure ci-après montre comment l'adressage par chaînage permet de résoudre les collisions au moment de l'insertion.

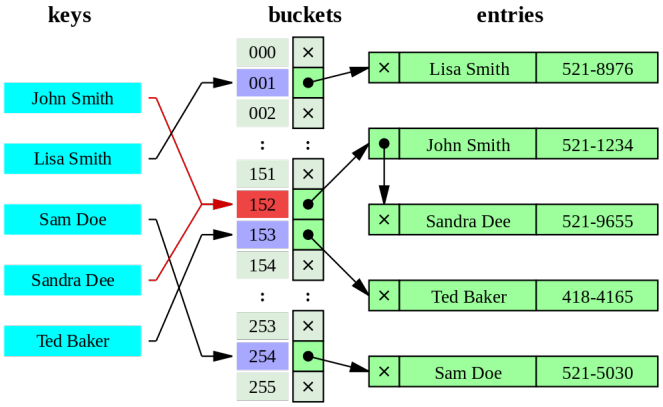

FIGURE 2 – Résolution d'une collision par chaînage. © Wikipedia.

Aller plus loin

Méthodes supplémentaires. La gestion décrite est fonctionnelle mais minimale. Afin d'obtenir une implémentation plus complète des tables de hachage, il est possible de rajouter quelques fonctionnalités :

— Itérateurs (pour parcourir la table notamment)
— Un critère de comparaison pour les clés
— Une gestion de la charge (avec un *facteur* de charge défini par $\frac{p}{n}$, où p est le nombre d'emplacements occupés).
— ...

Adressage ouvert. L'adressage ouvert est une gestion différente des collisions dans une table de hachage. Dans ce modèle d'adressage, chaque emplacement contient une unique valeur. Lorsqu'une clé est hachée sur un emplacement déjà occupé, un emplacement libre est cherché dans la suite de la séquence (par exemple en parcourant séquentiellement le tableau d'`Emplacement`).

Il existe différentes manières de parcourir la table à la recherche d'un `Emplacement` libre (ou de la clé, si elle appartient déjà à la table), la plus simple étant de parcourir les éléments un par un.

L'implémentation de cette seconde méthode de résolution peut amener à repenser la modélisation précédente. Il est peut-être nécessaire d'avoir une classe abstraite pour les tables de hachage, avec des méthodes d'insertion, de recherche et de suppression virtuelles pures.