TP numéro 3
Tables de hachage

Programmation avancée, ENSIIE

Semestre 2, 2014–15

Vous pouvez réaliser ce TP au choix en OCaml ou en C.

Exercice 1 : Interface

1. Écrire le fichier d'interface des dictionnaires (fichier `dictionnaire.[mli|h]`). Il contient le type abstrait des dictionnaires, et les quatre prototypes des fonctions de manipulation des dictionnaires `creer`, `rechercher`, `inserer` et `supprimer`.
2. Écrire un fichier `test.[ml|c]` qui déroulera le scénario suivant :
   - créer un dictionnaire vide de taille 2 ;
   - ajouter l'association `3` $\mapsto$ `"coucou"` dans le dictionnaire ;
   - vérifier que la recherche de `3` dans le dictionnaire renvoie bien `"coucou"` ;
   - ajouter l'association `1` $\mapsto$ `"toto"` dans le dictionnaire ;
   - ajouter l'association `2` $\mapsto$ `"titi"` dans le dictionnaire ;
   - vérifier que la recherche de `1` dans le dictionnaire renvoie bien `"toto"` ;
   - supprimer `1` dans le dictionnaire ;
   - vérifier que la recherche de `3` dans le dictionnaire renvoie bien `"coucou"`.
3. Écrire le `Makefile` correspondant. Indication : `test.cmo` (resp. `test.o`) dépend de `dictionnaire.cmi` (resp. `dictionnaire.h`).

Exercice 2 : Tables de hachage

1. Dans un fichier `hashtable.[ml|c]`, implémenter une fonction `hash` qui prend un entier `k` et une taille de table `m` et qui applique la méthode de la multiplication.
2. Dans ce même fichier, implémenter les dictionnaires à l'aide de tables de hachage. On codera tout d'abord une version sans redimensionnement dynamique.
3. Modifier le `Makefile` pour permettre la compilation du programme de test utilisant cette implémentation.

   Indication : en OCaml, il faut copier `dictionnaire.mli` en `hashtable.mli`, et utiliser `Hashtable` au lieu de `Dictionnaire` dans le fichier `test.ml`.
4. Compiler, tester.
5. Modifier le fichier `hashtable.[ml|c]` pour ajouter du redimensionnement dynamique lors d'une insertion. Il faudra donc que la table de hachage conserve l'information du nombre d'éléments insérés.