TP numéro 1
Dictionnaires

Programmation impérative avancée, ENSIIE

Semestre 2, 2015–16

Exercice 1 : Interface

1. Écrire le fichier d'interface des dictionnaires associations à des entiers des chaînes de caractères (fichier `dictionnaire.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.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.o` dépend de `dictionnaire.h`.

Exercice 2 : Listes d'association

4. Dans un fichier `listeassoc.c`, implémenter les dictionnaires à l'aide de listes d'association.
5. Modifier le `Makefile` pour permettre la compilation du programme de test utilisant cette implémentation.
6. Compiler, tester.

Exercice 3 : Mesures de complexité

7. Modifier le fichier `test.c` pour effectuer les opérations suivantes, où n est un entier passer en paramètre au programme :
   - Créer un dictionnaire de taille $\frac{n}{10}$.

— Insérer n associations entre un entier choisi aléatoirement entre 0 et $n-1$ et une chaîne contenant sa valeur. On pourra utiliser `sprintf` pour créer la chaîne, on pensera à allouer un nouvel espace mémoire de la bonne taille pour la chaîne.

— Supprimer $\frac{n}{2}$ éléments associés à un entier choisi aléatoirement entre 0 et $n-1$. On libérera la mémoire allouée pour la chaîne de caractère.

— Afficher, pour chaque entier entre 0 et $n-1$, soit la valeur associée dans le dictionnaire si elle existe, soit un message disant qu'elle n'existe pas.

8. Instrumenter le code de `test.c` avec la fonction `clock()` (cf. man) pour afficher le temps moyen en μs nécessaire pour réaliser chacune des opérations d'insertion, de recherche et de suppression. (On enlèvera les affichages.)
9. Trouver la relation liant ces temps avec la taille de l'entrée.