Examen

5 janvier 2021 — Durée 2h

Document autorisé : **Mémento C** vierge de toute annotation

On étudie dans ce sujet d'un paquetages d'*ensembles d'entiers*. Ce paquetage permet de créer des ensembles vides, d'ajouter ou de supprimer des éléments, de rechercher des éléments dans des ensembles. Le fichier d'en-tête `ensembles.h` contenant la spécification du paquetage se trouve en annexe.

On suppose l'existence d'une implémentation de ce paquetage, nommée `ensembles.c`. **Cette implémentation n'est pas demandée et n'est pas nécessaire pour les exercices de ce sujet.**

Exercice 1. (4 pt) Écrire un programme C nommé `test-ensembles-simple.c`, permettant de tester les fonctions d'ajout, de suppression et de recherche d'éléments.

Ce programme prend en paramètre le nom d'un fichier d'entrée, qui contient :

1. une séquence d'entiers S_1, sous la forme :
   (a) sur la première ligne, un entier N, le nombre d'entiers de la séquence
   (b) N entiers à ajouter dans l'ensemble ;
2. une deuxième séquence S_2, du même format ;
3. un entier qu'on souhaite rechercher dans l'ensemble créé.

Ce programme doit créer un ensemble vide, ajouter tous les éléments de la séquence S_1 dans cet ensemble, supprimer tous les éléments de S_2, puis afficher si l'élément à chercher appartient à l'ensemble final ou non.

Exercice 2. (3 pt) On cherche à tester si la propriété suivante est vérifiée : « *si on ajoute un élément x dans un ensemble, puis qu'on supprime cet élément, alors la recherche de cet élément x dans l'ensemble obtenu renvoie faux* ».

Décrire un jeu de tests fonctionnels pour le programme de l'exercice précédent, permettant de tester cette propriété ; donner en complément de cette description deux exemples de test de ce jeu de tests.

Exercice 3. (1 pt) Donner un exemple de test de robustesse pour le programme de l'exercice 1.

Exercice 4. (2 pt) Sur le même modèle qu'à l'exercice 2, donner une autre propriété que doivent vérifier les fonctions du paquetage **ensembles**. Donner un test, utilisant le même programme, permettant de vérifier cette propriété.

Exercice 5. (5 pt) On dispose maintenant d'un deuxième paquetage, **ensembles-hash**, de même spécification, mais d'implémentation différente. Le fichier d'en-tête contenant la spécification de ce paquetage est fourni en annexe.

On souhaite tester ce deuxième paquetage, en comparant son comportement avec celui du premier paquetage. Pour cela, on va effectuer des séries d'opérations d'ajouts, de suppressions et de recherches identiques sur deux ensembles avec les deux implémentations.

Une série d'opération sera décrite dans un fichier, sous la forme suivante :

1. sur la première ligne, un entier N, le nombre d'opérations de la série ;
2. sur les N lignes suivantes, une opération par ligne :
   — la lettre `A`, suivie d'un entier x, pour une opération d'ajout de l'entier x ;
   — la lettre `S`, suivie d'un entier, pour une opération de suppression ;
   — la lettre `R`, suivie d'un entier, pour une opération de recherche.

Écrire un programme qui lit une séquence d'opérations, réalise ces opérations en parallèle sur deux ensembles avec les deux implémentations différentes, et teste systématiquement si les opérations de recherches renvoient un résultat identique.

Exercice 6. (3 pt) Décrire un jeu de tests fonctionnels pour ce programme, en indiquant systématiquement les propriétés testées.

Exercice 7. (2 pt) Écrire un Makefile permettant de compiler les programme des différents exercices. L'exécution de la commande `make` sans argument doit générer tous les exécutables correspondants aux différents programmes.

Annexes

Paquetage ensembles : contenu du fichier ensembles.h

```
1  #ifndef _ENSEMBLES_H_
2  #define _ENSEMBLES_H_
3
4  #include <stdbool.h>
5
6  struct cellule;
7  typedef struct cellule Cellule;
8
9  typedef struct liste {
10   Cellule * tete;
11 } Liste;
12
13 /* Type Ensemble : ensembles d'entiers */
14 typedef Liste Ensemble;
15
16 /* Création d'un ensemble vide
17    Précondition : ens est un pointeur sur une structure allouée
18    Effet : aprè s l'appel de la fonction creer_vide(ens), ens pointe sur un ensemble vide
19  */
20 void creer_vide(Ensemble * ens);
21
22 /* Ajout d'un élément dans un ensemble
23    Précondition : ens est un pointeur sur une structure allouée
24    Effet : ajout_element(ens, x) ajoute l'élément x àl'ensemble ens.
25    Si x est déjà dans l'ensemble, ens n'est pas modifié.
26  */
27 void ajout_element(Ensemble * ens, int x);
28
29 /* Suppression d'un élément dans un ensemble
30    Préconditions :
31    - ens est un pointeur sur une structure allouée
32    - ens contient l'élément x
33    Effet : suppression_element(ens, x) supprime l'élément x de l'ensemble ens.
34  */
35 void suppression_element(Ensemble * ens, int x);
36
37 /* Recherche d'un élément dans un ensemble
38    Précondition : ens est un pointeur sur une structure allouée
39    Valeur de retour : recherche_element(ens, x) renvoie vrai ssi x appartient àens.
40  */
41 bool recherche_element(Ensemble * ens, int x);
42
43 #endif
```

Paquetage ensembles-hash : contenu du fichier ensembles-hash.h

```c
#ifndef _ENSEMBLES_HASH_H_
#define _ENSEMBLES_HASH_H_

#include <stdbool.h>

#define NMAX 1019

struct cellule;
typedef struct cellule Cellule;

typedef struct {
  Cellule * table[NMAX];
} Table;

/* Type Ensemble : ensembles d'entiers */
typedef Table Ensemble_Hash;

/* Création d'un ensemble vide
   Précondition : ens est un pointeur sur une structure allouée
   Effet : aprè s l'appel de la fonction creer_vide_hash(ens), ens pointe sur un ensemble vide
 */
void creer_vide_hash(Ensemble_Hash * ens);

/* Ajout d'un élément dans un ensemble
   Précondition : ens est un pointeur sur une structure allouée
   Effet : ajout_element_hash(ens, x) ajoute l'élément x àl'ensemble ens.
   Si x est déjà dans l'ensemble, ens n'est pas modifié.
 */
void ajout_element_hash(Ensemble_Hash * ens, int x);

/* Suppression d'un élément dans un ensemble
   Préconditions :
   - ens est un pointeur sur une structure allouée
   - ens contient l'élément x
   Effet : suppression_element_hash(ens, x) supprime l'élément x de l'ensemble ens.
 */
void suppression_element_hash(Ensemble_Hash * ens, int x);

/* Recherche d'un élément dans un ensemble
   Précondition : ens est un pointeur sur une structure allouée
   Valeur de retour : recherche_element(ens, x) renvoie vrai ssi x appartient àens.
 */
bool recherche_element_hash(Ensemble_Hash * ens, int x);

#endif
```