Devoir surveillé

25 octobre 2018 — Durée 1h15

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

On considère le paquetage **recherche**, qui fournit une fonction `recherche_indice`. Cette fonction prend en argument un entier `x` et une séquence `t` (de type `tableau_entiers`) *triée par ordre croissant*, et renvoie l'indice de la première occurrence de `x` dans `t`. La fonction renvoie la valeur -1 si `t` ne contient pas la valeur `x`.

L'implémentation de ce paquetage est donnée ci-dessous :

```c
#include "recherche.h"

/* Données : un entier x, un tableau d'entiers trié t
   Renvoie :
   - l'indice dans le tableau de la première occurrence de x
   - -1 si le tableau ne contient pas x
   Précondition : le tableau t est trié par ordre croissant
 */
int recherche_indice(int x, tableau_entiers t) {
  int ib, ih, im;

  /* Recherche dichotomique */
  ib = 1;
  ih = t.taille-1;
  while (ib < ih) {
    im = (ib + ih) / 2;
    if (t.tab[im] < x) {
      ib = im+1;
    } else {
      ih = im;
    }
  }
  if (t.tab[ib] == x) {
    return ib;
  } else {
    return -1;
  }
}
```

Les paquetages **type_tableau** et **es_tableau** sont donnés en annexe.

Exercice 1. (2 pt)

Écrire le fichier *spécification* du paquetage **recherche**.

```c
#ifndef _RECHERCHE_H_
#define _RECHERCHE_H_

#include <stdbool.h>
#include "type_tableau.h"

/* Données : un entier x, un tableau d'entiers trié t
   Renvoie :
   - l'indice dans le tableau de la première occurrence de x
   - -1 si le tableau ne contient pas x
   Précondition : le tableau t est trié par ordre croissant
 */
int recherche_indice(int x, tableau_entiers t);

#endif
```

Exercice 2. (4 pt)

Écrire un programme de test du paquetage **recherche**. Ce programme doit utiliser les paquetages **type_tableau** et **es_tableau** fournis en annexe, pour lire un tableau dans un fichier dont le nom est donné en argument de la ligne de commande, lire un entier `x` sur l'entrée standard, et afficher le résultat de l'appel de la fonction `recherche_indice`.

```c
#include <stdio.h>

#include "type_tableau.h"
#include "es_tableau.h"
#include "recherche.h"

int main(int argc, char ** argv) {
  FILE * fentree;
  tableau_entiers t;
  int x, res;

  fentree = fopen(argv[1], "r");

  /* Lecture du tableau dans le fichier d'entrée */
  lire_tableau(fentree, &t);

  /* Lecture de l'entier à chercher dans le tableau */
  fscanf(fentree, "%d", &x);

  fclose(fentree);

  res = recherche_indice(x, t);

  printf("Résultat : %d\n", res);

  return 0;
}
```

Exercice 3. (5 pt)

Décrire un jeu de tests fonctionnels permettant de tester la fonction `recherche_indice` à l'aide du programme écrit à l'exercice précédent.

NB : il est demandé de *décrire* ce jeu de tests, et donc de *justifier* sa construction, sans nécessairement écrire explicitement les valeurs de chaque test.

Un test est composé d'une séquence s et d'un entier x. On note n la taille de la séquence s, et par commodité $s[i]$ la valeur `s.tab[i]`.
On décompose les cas de tests en fonction :

1. *de la taille de la séquence,*
2. *de la répétition ou non de valeurs dans s,*
3. *de l'appartenance ou non de x à s,*
4. *de la position de x dans s.*

Pour différentes tailles de séquences :

- *cas limite : $t = [], x = 0$*
- *cas limite : $t = [1]$*
  - *trois valeurs de x testées : 0, 1, 2*
- *cas général : plusieurs séquences de longueurs arbitraires (longueurs paires, impaires), en répétant ou non des valeurs. Cas limite : séquence d'une seule valeur répétée n fois.*
  Pour chaque séquence :
  - *cas limites :*
    - *un test avec un entier $x < s[0]$*
    - *avec un entier $x > s[n-1]$*
    - *avec $x = s[0]$*
    - *avec $x = s[n-1]$*
  - *cas généraux :*
    - *un test avec un entier x égal à une des valeurs de la séquence*
    - *un test avec un entier x égal à une des valeurs répétées plusieurs fois dans la séquence*
    - *un test avec un entier x compris strictement entre deux valeurs consécutives de la séquence*

Étant donné qu'il s'agit de tests fonctionnels, *toutes les séquences du jeu de tests doivent être triées par ordre croissant.*

Exercice 4. (2 pt)

La fonction `recherche_indice` comporte au moins une erreur.

Identifier une de ces erreurs. Donner un test permettant de mettre cette erreur en évidence.

NB : il n'est pas demandé de *corriger* l'erreur trouvée !

Deux erreurs possibles :

- `ib` *est initialisé à 1 (au lieu de 0). Le résultat n'est pas correct si le premier élément de la séquence est égal à* `x`*. Test : $s = [1, 2], x = 1$.*
- *Accès à* `t.tab[ib]`*, avec* `ib` *pouvant être strictement supérieur à* `t.taille`*. Test : $s = [], x = 0$ (en fonction de l'état de la mémoire et des valeurs précédentes de s...).*

Exercice 5. (4 pt)

On souhaite maintenant utiliser un *oracle* pour vérifier automatiquement que le résultat fourni par la fonction `recherche_indice` est correct. Quelles sont les propriétés à vérifier par cet oracle ? Écrire une fonction oracle vérifiant ces propriétés.

```
1  bool oracle_recherche(int x, tableau_entiers t, int resultat) {
2    int i, res_attendu;
3
4    /* Recherche de la première valeur x dans t */
5    i = 0;
6    while ((i < t.taille) && (x > t.tab[i])) {
7      i = i + 1;
8    }
9    if ((i < t.taille) && (t.tab[i] == x)) {
10     res_attendu = i;
11   } else {
12     res_attendu = -1;
13   }
14   return (resultat == res_attendu);
15 }
```

Exercice 6. (3 pt)

Écrire un Makefile permettant de compiler le programme écrit à l'exercice 2.

```
1  CC=clang
2
3  all: test_recherche
4
5  %.o: %.c
6        $(CC) -c $<
7
8  es_tableau.o: es_tableau.c es_tableau.c es_tableau.h type_tableau.h
9
10 recherche.o: recherche.c recherche.h type_tableau.h
11
12 test_recherche.o: test_recherche.c recherche.h es_tableau.h type_tableau.h
13
14 test_recherche: test_recherche.o recherche.o es_tableau.o
15       $(CC) $^ -o $@
```

Annexes : paquetages utilisés

Paquetage type_tableau

```c
#ifndef _TYPE_TABLEAU_H_
#define _TYPE_TABLEAU_H_

#define TAILLE_MAX 10000

/* Définition du type vecteur_entiers :
   tableau d'entiers de taille TAILLE_MAX */
typedef int vecteur_entiers[TAILLE_MAX];

/* Structure contenant un tableau (de taille TAILLE_MAX) et un entier
   taille : le nombre d'entiers du tableau */
typedef struct {
  int taille;
  vecteur_entiers tab;
} tableau_entiers;

#endif /* _TYPE_TABLEAU_H_ */
```

Paquetage es_tableau

```c
#ifndef _ES_TABLEAU_H_
#define _ES_TABLEAU_H_

#include <stdio.h>
#include "type_tableau.h"

void lire_tableau(FILE * fichier, tableau_entiers * t);

void ecrire_tableau(FILE * fichier, tableau_entiers t);

#endif /* _ES_TABLEAU_H_ */
```

```c
#include <stdio.h>
#include "es_tableau.h"

void lire_tableau(FILE * fichier, tableau_entiers * t) {
  int i;

  /* Lecture de la taille du tableau */
  fscanf(fichier, "%d", &(t->taille));

  /* Lecture des valeurs du tableau */
  for (i = 0; i < t->taille; i++) {
    fscanf(fichier, "%d", &(t->tab[i]));
  }
}

void ecrire_tableau(FILE * fichier, tableau_entiers t) {
  int i;

  /* Écrire la taille du tableau dans le fichier */
  fprintf(fichier, "%d\n", t.taille);

  /* Écrire les valeurs du tableau */
  for (i = 0; i < t.taille; i++) {
    fprintf(fichier, "%d\n", t.tab[i]);
  }
}
```