EI6PG116 Atelier algorithmique et programmation

Tableau de bord / Mes cours / ENSEIRB-MATMECA / INFORMATIQUE / Semestre 6 / EI6PG116



Session BBB pour le G2

Liens d'accès aux ressources

Semaine 1

Semaine 2

Semaine 3

Semaine 4

Dépôts git et organisation du code

Projet sur thor

Semaine 1

L'objectif est de réaliser d'abord la mise en œuvre du type set avec sentinelle ainsi que ses tests structurels, puis la mise en œuvre par tableau dynamique ainsi que les tests structurels correspondants. Et enfin, écrire les tests fonctionnels et les appliquer aux deux mises en œuvres. Il est essentiel de répondre aux questions de cours au fur et à mesure.



Feuille TD semaine 1

Prototypes des 3 fonctions utilitaires pour la mise en œuvre avec sentinelle

```
// returns the first position p in s such that s[p]>=c if any, or
// the position of SET__BOUND otherwise
// we assume that s is sorted and terminated by SET__BOUND
int find(int const s[], int c);
// moves all elements in s starting from position begin and up to
// SET__BOUND included, one position to the right.
// The value in s[begin] is left unchanged
// we assume that s is terminated by SET__BOUND, and that shifting from
// begin to SET__BOUND to the right is valid in s
void shift_right(int s[], int begin);
// moves all elements in s starting from position begin and up to
// SET__BOUND included, one position to the left.
// we assume that s is terminated by SET__BOUND, and that shifting from
// begin to SET__BOUND to the left is valid in s
void shift_left(int s[], int begin);
```

Prototypes des 3 fonctions utilitaires pour la mise en œuvre dynamique

1 sur 6 10/02/2022, 12:20

```
// returns the first position p in s such that begin<=p<=end and s[p]>=c
// if any, or end otherwise
// we assume that s is sorted and that all indices in begin..end are valid in s
int find(int const s[], int c, int begin, int end);
// moves all elements in s from indices begin..end (included) to
// indices begin+1..end+1
// the indices in begin..end+1 are assumed to be valid positions in s
void shift_right(int s[], int begin, int end);
// moves all elements in s from indiced begin..end (included) to
// indices begin-1..end-1
// the indices in begin-1..end are assumed to be valid positions in s
void shift_left(int s[], int begin, int end);
```

Changement de prototype de set size

Attention! On remplace:

```
int set__size(struct set const * se);
```

par:

```
size_t set__size(struct set const * se);
```

Et il est nécessaire d'inclure stdlib.h dans le fichier set.h

Test Fonctionnels.

Dans les tests fonctionnels fichier set/test_set_func.c remplacer la ligne #include "set_sentinel.h" //FIX ME par les lignes suivantes

```
#if defined SENTINEL
#include "set_sentinel.h"
#elif defined DYNAMIC
#include "set_dynamic.h"
#error "A set implementation has to be defined" // -Wfatal-errors
#endif
```

La compilation du fichier test_set_func.c doit maintenant définir la macro-constante (SENTINEL OU DYNAMIC) avec l'option -D (comme exemple voir la cible test sentinel func.o: dans le fichier Makefile)

Spécification du comportement de set dynamic

L'implémentation d'un tableau trié à capacité variable set_dynamic doit vérifier les propriétés suivantes :

- l'ensemble vide est tel que .size=0, .capacity=0, et .s=NULL
- si, avant d'ajouter un élément dans le tableau, celui-ci était tel que (size == capacity), alors on réalloue le tableau de manière à doubler sa taille si elle était non nulle, sinon de fixer cette taille à 1.
- si, après avoir retiré un élément dans le tableau, celui-ci était tel que (size <= capacity/2), alors on réalloue le tableau de manière à ce que sa taille devienne capacity/2 (et donc vaut 0 si capacity valait 1).
- Invariant : le champ capacity est toujours représentatif de la taille du tableau pointé par s

Dans l'exemple suivant, on affiche les tailles, capacités, et contenus du tableau lors de l'ajout successif de 5 éléments, puis le retrait successif de ces 5 éléments :

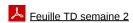
2 sur 6 10/02/2022, 12:20

```
\{.size = 0, .capacity = 0, .s = \{\}\}
                                               // s is the NULL pointer
\{.size = 1, .capacity = 1, .s = \{1\}\}
\{.size = 2, .capacity = 2, .s = \{1,2\}\}
\{.size = 3, .capacity = 4, .s = \{1,2,3\}\}
\{.size = 4, .capacity = 4, .s = \{1,2,3,4\}\}
\{.size = 5, .capacity = 8, .s = \{1,2,3,4,5\}\}
\{.size = 4, .capacity = 4, .s = \{2,3,4,5\}\}
\{.size = 3, .capacity = 4, .s = \{3,4,5\}\}
{.size = 2, .capacity = 2, .s = \{4,5\})
{.size = 1, .capacity = 1, .s = \{5\})
{.size = 0, .capacity = 0, .s = \{\}}
                                                // s is the NULL pointer
```

10/02/2022, 12:20 3 sur 6

Semaine 2

L'objectif est de mettre en œuvre la struct set avec une liste chaînée, ainsi que les tests structurels associés. Vérifiez que vos tests fonctionnels s'exécutent sans problème sur votre nouvelle mise en œuvre.



Mise à jour du Makefile pour link

Liste des modifications à apporter:

Définir la variable LNK_DIR=link

Ajouter la cible test_link à la variable BIN

Ajouter la directive de compilation - I \${LNK_DIR} à la variable CPPFLAGS

Ajouter les règles suivantes pour link (attention aux tabulations) :

```
#link
link.o: ${LNK_DIR}/link.c ${LNK_DIR}/link.h
        ${CC} ${CPPFLAGS} ${CFLAGS} ${LNK_DIR}/link.c -c
test_link.o : ${LNK_DIR}/test_link.c ${LNK_DIR}/link.h
       ${CC} ${CPPFLAGS} ${CFLAGS} ${LNK_DIR}/test_link.c -c
test_link: test_link.o link.o
        ${CC} test_link.o link.o -o $@ ${LDFLAGS}
```

Les règles ci-dessus supposent que les fichiers link/link.h, link/link.c et link/test_link.c existent. Créez ces fichiers vides, puis débutez par la mise en œuvre de link en développant les tests structurels associés.

Enfin, implémentez et testez set_link.

Créer les fichiers link/set_link.h, link/set_link.c et link/test_link_struc.c.

Ajoutez les règles suivantes pour set link (attention aux tabulations) :

```
set_link.o: ${LNK_DIR}/set_link.c ${LNK_DIR}/set_link.h
        ${CC} ${CPPFLAGS} ${CFLAGS} ${LNK_DIR}/set_link.c -c
test_link_struc.o : ${LNK_DIR}/test_link_struc.c ${LNK_DIR}/set_link.h ${LNK_DIR}/link.h
        ${CC} ${CPPFLAGS} ${CFLAGS} ${LNK_DIR}/test_link_struc.c -c
link_struc: test_link_struc.o set_link.o link.o
        ${CC} test_link_struc.o set_link.o link.o -o $@ ${LDFLAGS}
```

Pour exécuter les tests fonctionnels de set (set/test_set_fonc.c) sur cette réalisation, définir sur le modèle des tests fonctionnels de sentinel les deux cibles test_link_fonc.o (compilation de test_set_fonc.c avec -DLINK) et link_func dans

Sur thor, nos tests fonctionnels et structurels ne sont déclenchés qui si la cible link_func est définie et produit un exécutable.

- </> link.h
- </ ink.c
- test link.c
- set link.h

Semaine 3

Feuille TD semaine 3

</> affectation.c

Semaine 4

Feuille TD semaine 4



problem2.c



problem4.c



Tests unitaires sur de grosses instances de set. Permet de tester les allocations mémoires. Attention, les dépassements de capcité sur le tableau à sentinelle ne sont pas détectées. Les tests doivent échouer si votre code détecte les dépassements de capacité. Vous pouvez également utiliser valgrind pour le vérifier.

Il faut lier le fichier data-int.c, bigtests-int.c et l'une des mises en œuve de set pour obtenir un exécutable.

Ces tests peuvent également être utilisés pour comparer les performances des mises en œuve. À titre indicatif:

- set-dynamic: 2.24s - set-link: 31.83s - set-sentinel: 16.30s

(compilation avec -O3, sur MacBook Pro Late 2013, clang-700.1.81)

Pour découvrir d'où viennent ces différences, utilisez l'outil gprof qui permet de savoir quelles fonctions consomment le plus de temps de calcul.

Dépôts git et organisation du code

Vous disposez d'un dépôt git individuel hébergé sur la machine thor, qui lance régulièrement des tests automatiques pour vérifier vos mises en œuvre et vous aider à y découvrir des bugs que vos propres tests ne parviendraient pas à détecter.

Organisation du code

Votre dépôt doit impérativement suivre l'organisation ci-dessous pour le bon fonctionnement des tests automatiques. Vos Makefiles doivent proposer une règle build qui compile votre code, et une règle test qui lance les tests (vous pouvez bien sûr ajouter d'autres règles).

```
Makefile
set/
-- set.h
-- test_set_func.c
```

Connecté sous le nom « Mouad Boumour » (Déconnexion)

<u>Accueil</u>

Bordeaux INP

www.bordeaux-inp.fr











Accueil (vers composantes)

Tutos

<u>étudiants</u>

enseignants

Français (fr)

English (en)

Español - España (es es)

Español - Internacional (es)

Français (fr)

Résumé de conservation de données

Obtenir l'app mobile

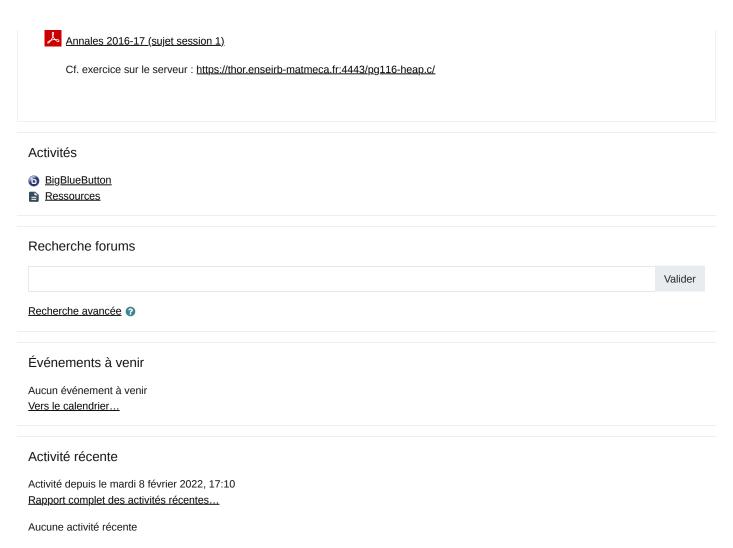
Politiques



Sujet examen 2018/2019 session 1



annales-pg116-2018-19-session1



6 sur 6 10/02/2022, 12:20