TD PG106
tests et couverture

Créer une branche pour le travail sur les tests, vous finirez avec un *merge* sur *master* une fois l'environnement de tests opérationnel.

▸**Exercice 1.** *Environnement de test.*
Dans le CMakeLists.txt *principal (à la racine du dépôt) ajouter un appel à la commande* `include (CTest)` *pour ajouter le support des tests dans* cmake.

Ajouter un répertoire appelé `tests`. *Dans ce répertoire, créer un premier fichier* `hash_unit_tests.c` *qui comportera le code :*

```
#include <assert.h>

int main(){
assert(false && "mon premier test!");
}
```

Ajouter un fichier `CMakeLists.txt` *qui permet la création de l'exécutable* `hash_unit_tests` *à partir de ce fichier source. Enfin, utiliser la commande* `add_test` *dans le fichier* `CMakeLists.txt` *pour automatiser le lancement de votre programme lors de l'appel à la commande* `make test` *depuis le répertoire de compilation.*

Vérifier bien que tout fonctionne avant de passer à la suite. Pour rappel, `make VERBOSE=1` *permet de voir les commandes de compilation utilisées.*

Depuis votre répertoire de build, *utiliser la commande* `ctest -T memcheck`. *Cela permet de lancer les tests sous valgrind.*

▸**Exercice 2.** *Tests unitaires*
Ecrire des tests unitaires pour chacunes des fonctions de la bibliothèque : init, ajout et recherche.

Vérifier que vos tests fonctionnent puis fusionner vos modifications dans master.

▸**Exercice 3.** *Nouveaux développements*
Nous allons maintenant ajouter une nouvelle fonction dans la bibilothèque : `hash_remove`. *Ajouter cette fonction et le test unitaire correspondant dans une branche que vous fusionnerez dans* master *une fois le travail fait.*

1 Couverture

La couverture est un élément important des tests. Cela correspond au nombre de lignes des fichiers sources qui ont été utilisées lors de l'exécution d'un test. Avant d'automatiser la génération de rapport de couverture, nous allons regarder comment cela fonctionne.

▸**Exercice 4.** *A la main.*
En utilisant `gcc`, *compiler à la main en une fois votre programme de test avec le fichier* hash.c *et cela avec l'option* `--coverage` *(cela doit donner une commande ressemblant à* `gcc tests/hash_unit.c src/hash.c -I. --coverage`*). Vous verrez qu'en plus de votre binaire de nouveaux fichiers ont été créés. Lancer votre programme de test, cela crée des fichiers de traces d'exécution.*

Utiliser le programme `gcov` *sur le fichier* `hash.c`. *Si tout ce passe bien, un fichier* `hash.c.gcov` *a été créé. Ouvrez ce fichier et analysez son contenu.*

▸**Exercice 5.** *cmake*
Nous allons ajouter un nouveau type de compilation (en plus des types Release et Debug) qui positionne

les options de compilation nécessaires pour avoir les informations de couverture lors du lancement des tests.

Pour cela ajouter la ligne ci-dessous dans le fichier CMakeLists.txt *principal :*

```
set(CMAKE_C_FLAGS_COVERAGE "-g -O0 -fprofile-arcs -ftest-coverage")
```

Créer un nouveau répertoire de compilation (`build_coverage` *par exemple) et configurer la compilation pour utiliser ce nouveau mode :* `cmake -DCMAKE_BUILD_TYPE=Coverage ...` *Compiler et lancer les tests, à l'aide de la commande* `find . -name "*gcna"` *vérifier que les fichiers liés à la couverture ont bien été générés.*

L'analyse des fichiers de traces peut se faire avec lcov *qui permet le parcours récursif à la recherche de traces et génére un rapport. Utiliser la commande suivante pour générer le rapport :*

```
lcov  --directory  . -c -o coverage.report
```

La commande suivante permet de générer un rapport en html :

```
genhtml -o html coverage.report
```

Ouvrez le fichier html/index.html *dans votre navigateur.*

▸**Exercice 6.** *automatisation lcov/genhtml*
Dans le CMakeLists.txt *principal, ajouter les lignes suivantes :*

```
if (CMAKE_BUILD_TYPE STREQUAL "Coverage")
add_custom_target(coverage lcov  --directory  ${CMAKE_CURRENT_BINARY_DIR} -c -o coverage.report
COMMAND genhtml -o html coverage.report)
endif()
```

Cela permet de créer une nouvelle règle de compilation make coverage *qui automatisera l'execution de* lcov *de* genhtml. *Vérifier que cela fonctionne correctement.*

▸**Exercice 7.** *Compléter vos tests*
Utiliser la couverture pour compléter vos tests.