Fiabilité logicielle – TP1 : Tests unitaires

Zakari IKHOU (responsable rendu)
Akim SAADI
Mohamed TAKHCHI
Reda CHANAA
Sohaib EL MOUHTADI

CUnit : environnement de test pour C

Nous avons commencé à coder le fichier triangle.c. Dans ce fichier, on peut retrouver deux fonctions : « readData » et « typeTriangle ».

La première fonction « readData » lit un document « .txt » pour extraire les longueurs des côtés du triangle et les attribuent à des variables (« coteA », « coteB » et « coteC »). Plusieurs cas de figure sont possibles. Le premier est le cas où le fichier n'existe pas : alors les côtés prendront tous la valeur « -1 ». Nous avons tout simplement réalisé cela avec un « if ».
Si le premier test est validé, on rentre dans une boucle « while » qui nous permet d'attribuer aux variables « coteA », « coteB » et « coteC » les valeurs présentes dans le fichier lu.
La dernière étape consiste à vérifier si le fichier lu possède exactement 3 valeurs ; si ce n'est pas le cas, la valeur « -1 » sera attribuée aux variables « coteA », « coteB » et « coteC ».

La fonction typeTriangle prend en entrée 3 variables de type « float » et renvoie un « int ». Nous avons codé plusieurs conditions avec des « if », « else if » et « else » dans le dernier cas. La première condition vérifie si les 3 valeurs en entrée sont strictement positives et peuvent former un triangle, c'est-à-dire que les 3 valeurs respectent l'inégalité triangulaire. Si ce n'est pas le cas, on renvoie la valeur « -1 ». La seconde condition vérifie tout simplement si les 3 cotés sont égaux : cela donnerait donc un triangle équilatéral ; on renvoie « 3 ». La 3ème condition vérifie si deux cotés sont égaux : cela donnerait un triangle isocèle ; on renvoie « 2 ». Et dans le dernier cas, si toutes les conditions n'ont pas été respectées, on obtient un triangle quelconque ; on renvoie donc « 1 ».

Les tests :

Les tests de « readData » :

Pour testReadNumbers : on crée un fichier avec les bonnes propriétés d'entrée, et on vérifie si les valeurs des côtés sont bien celles du fichier « .txt ».

Pour testReadLessNumbers : on crée un fichier avec 2 valeurs uniquement : on vérifie si la sortie nous renvoie bien « -1 » comme on le souhaite.

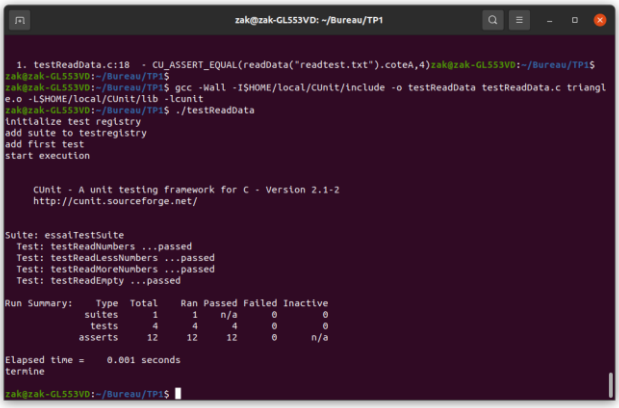
Pour testReadMoreNumbers : on crée un fichier avec exactement 4 valeurs : on vérifie si la sortie nous donne bien « -1 ».

Pour testReadEmpty : on crée un fichier vide on vérifie si la sortie est « -1 » comme on le souhaite.`

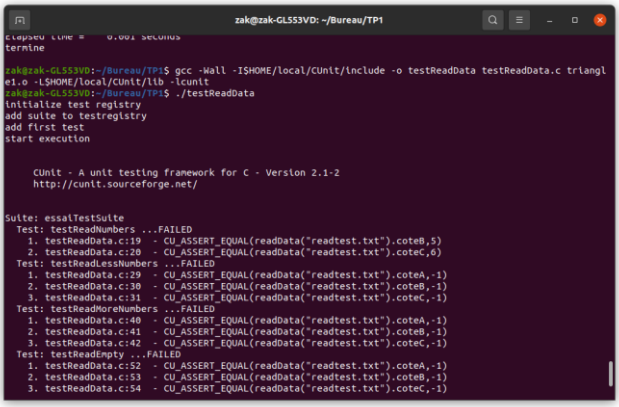
Pour la suite des tests, nous avons renommé le fichier triangle.o fourni sur Ametice par triangle1.o car à la création de notre point o avec les fonctions que nous avions créé, cela effaçait le fichier donner sur Ametice.

Nous avons obtenu pour les tests de « ReadData » les sorties suivantes :

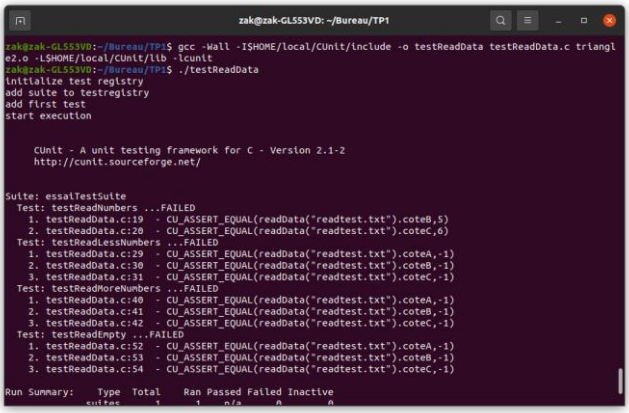
Pour triangle.o (le code que nous avons codé)



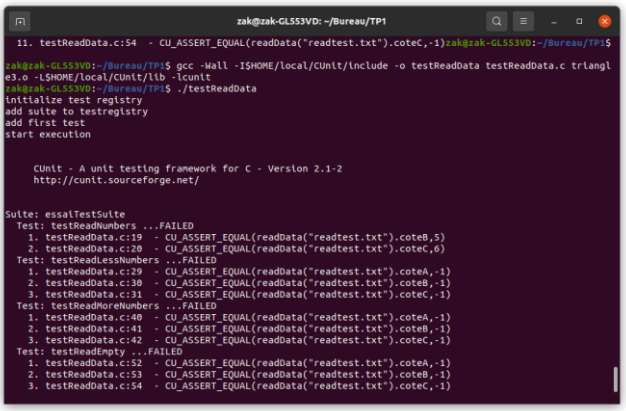
Pour triangle1.o



Pour triangle2.o



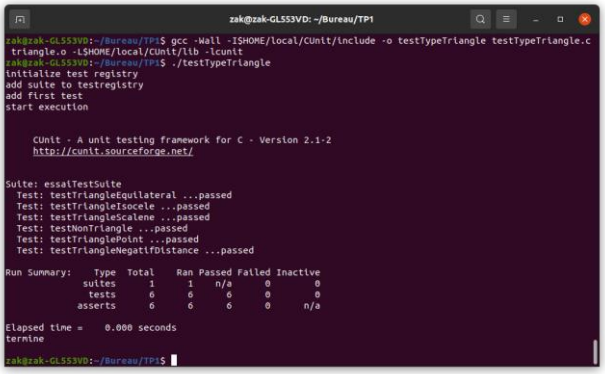
Pour triangle3.o



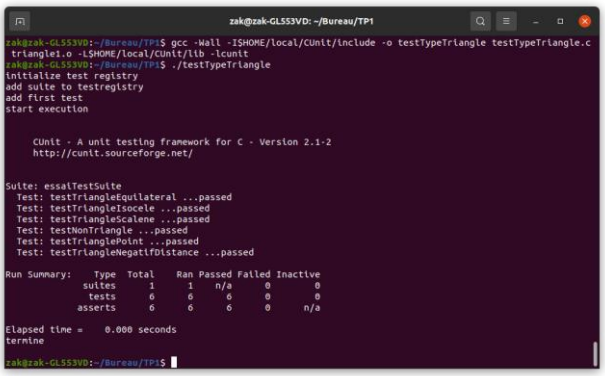
Pour triangle.o, tous les tests sont passés. Pour triangle1.o, triangle2.o et triangle3.o : aucun test n'est passé.

Nous avons obtenu pour les tests de « typetriangle » les sorties suivantes :

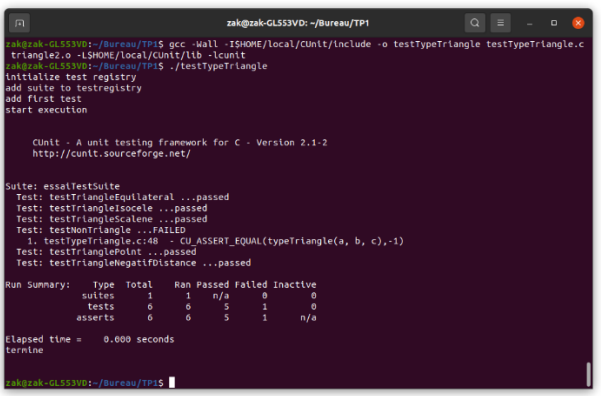
Pour triangle.o



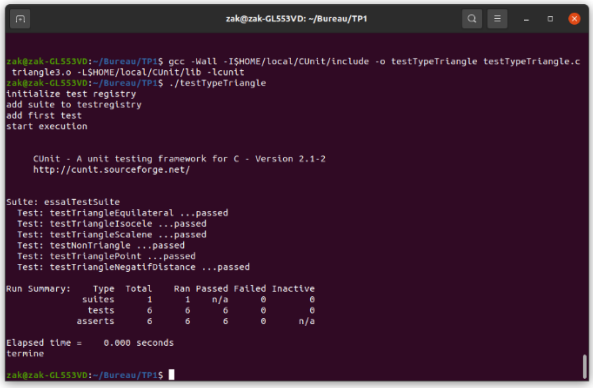
Pour triangle1.o



Pour triangle2.o



Pour triangle3.o



Pour triangle.o, triangle1.o et triangle3.o, tous les tests sont passés. Pour triangle2.o, le « testNonTriangle » n'est pas passé.

unittest : environnement de test en python

Nous commençons à coder la classe « ListeTriee » qui possède la méthode « chercherElt ».
La méthode « chercherElt » prend en entrée une valeur et une liste triée et renvoie le plus petit indice de l'emplacement de la valeur si elle existe dans la liste.

Pour cela, on procède pas dichotomie : cela nous permet d'aller plus vite. Nous utilisons une boucle « while » et des « if » pour nous permettre de trouver la valeur donnée en entrée. On se place au milieu de la liste et on vérifie si la valeur donnée en entrée se trouve dans la première moitié de la liste ou dans la seconde. Cela nous permet d'avoir un temps logarithmique.

Les tests :

Pour testEltIn : on vérifie si la valeur donnée en entrée est bien placée.

Pour testEltNotIn : on vérifie que l'on obtient bien la valeur « -1 » quand la valeur donnée en entrée ne fait pas partie de la liste.

Pour testEltMin : on vérifie que l'indice renvoyé est bien le plus petit des indices possibles.

```
In [2]: runfile('C:/Users/Zakari/Desktop/GROUPE10-Rendu-TP1/recherche/
testChercherElt.py', wdir='C:/Users/Zakari/Desktop/GROUPE10-Rendu-TP1/
recherche')
...
----------------------------------------------------------------------
Ran 3 tests in 0.005s

OK
```

Tous nos tests sont passés.