

Etudiant : Théo Dubus 22008507

TP2 C++

Ajouts dans cette version

- Support des fonctions unitaires
- Support des fonctions n-aires
- Support des fonction à nombre variable d'arguments

Difficultés rencontrés

L'ajout des fonction à nombre variable d'arguments, notamment pour le passage des arguments à la fonction lambda stockée en mémoire. La mémoire des fonction à aussi été complexe, au final cette mémoire est gérée par une classe dédiée `funStorage` avec toutes les méthodes nécessaire au management de cette mémoire, dont la vérification du nombre d'arguments.

Choix d'implémentation

La classe `Function` qui hérite de `Token` à été ajoutée pour gérer les fonctions. Ces fonctions sont calculées dans la méthode `eval`, ce qui permet de respecter la hiérarchie lexer/parser. On utilise `funStorage` pour stocker des fonction lambda associées à leur nom et leur nombre d'argument (mis à -1 si la fonction possède un nombre variable d'arguments).

Les tests unitaires ont aussi été ajoutés dans `test/programTest.cpp`. Une modification de la classe `Expression`, notamment de `tokensFromString()` simplifie le code et limite les passages d'arguments.

Compilation du projet

Le projet utilise le système de build CMake. Un script `buildrun.sh` permet de compiler, lancer les test et la calculatrice en 1 commande.

Le projet compile sous :

- `clang 10.0.0` sur Windows/Ubuntu
- `gcc 7.5.0` sur Ubuntu ou WSL (sous-système Windows pour Linux)

Tests

Les tests sont regroupés dans le répertoire `\test`. J'ai choisi d'utiliser la librairie de test Googletest qui est l'une des plus populaire (Catch, Boost.Test et Ctest sont aussi de bons candidats).

Les test sont séparés en 2 fichiers `.cpp` :

- `expressionTest.cpp` : Les tests d'analyse de tokens, créés au début du projet pour vérifier le fonctionnement de `tokensFromString`
- `programTest.cpp` : Les tests du fonctionnement du programme complet

- programme en une ligne (addition, priorités opératoires, parenthèses)
- programme multi lignes (variables, affichage ou non avec `;` , calcul du volume d'un cylindre...)

Fuites mémoire

Le projet a été testé avec `valgrind` pour détecter la présence ou non de 'memleaks'. Un script `testValgrind.sh` permet de tester les fuites sur un programme type composé de toutes les possibilités du langage.

Continuous integration

Le repo de ce projet est hébergé sur github.com (en privé pour éviter la copie), j'en ai donc profité pour expérimenter la continuous intégration avec [Github Actions](#) qui permet d'effectuer une série de test pour valider mes commit à chaque push.

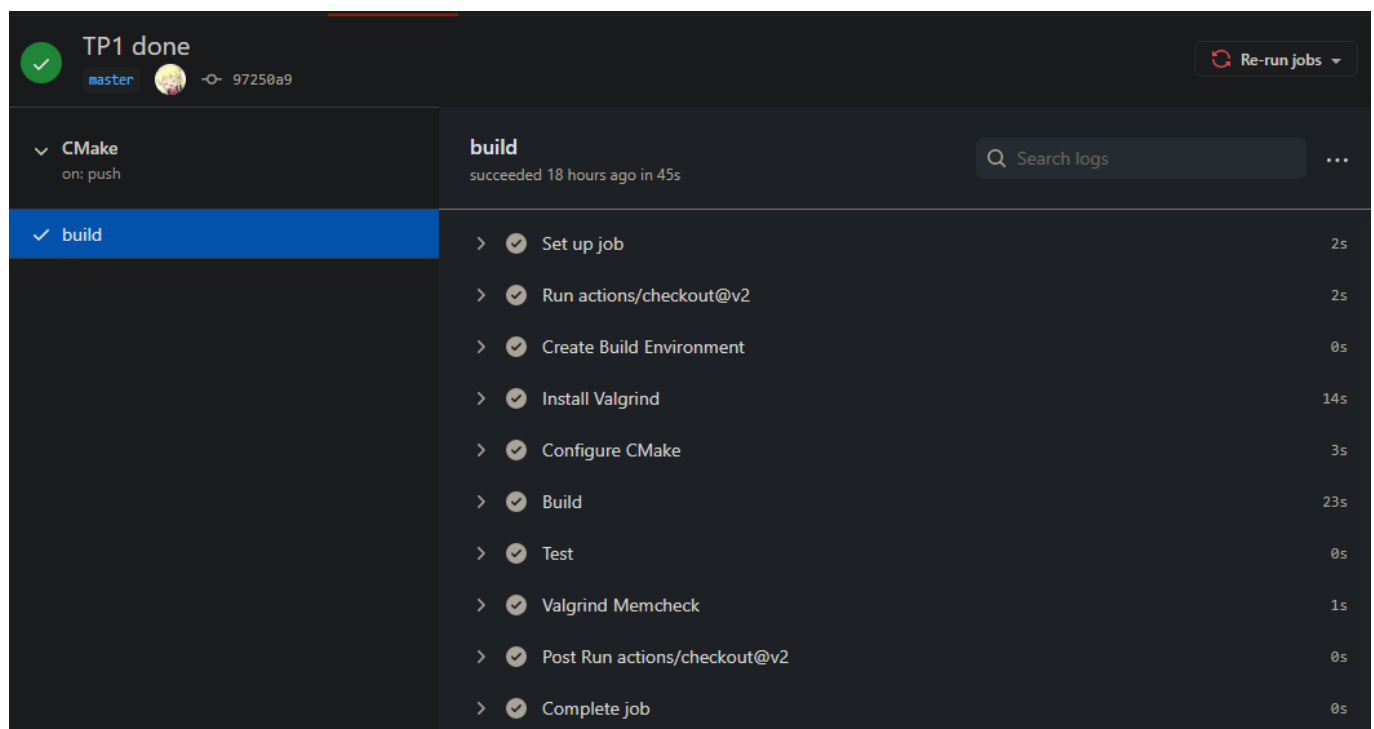
`.github/workflow/cmake.yml` contient le fichier yaml qui s'occupe de cette continuous integration.

Il y est effectué en autre:

- la compilation du projet avec CMake.
- Le lancement des tests Googletest.
- La vérification de présence des fuites mémoires avec Valgrind.

Exemples d'execution de CI : \

- Execution du "workflow" avec succès :



The screenshot shows a GitHub Actions workflow run titled "TP1 done" on the master branch. The workflow is named "CMake" and was triggered by a push. The "build" job is highlighted and shows a list of steps that all completed successfully:

Step	Duration
Set up job	2s
Run actions/checkout@v2	2s
Create Build Environment	0s
Install Valgrind	14s
Configure CMake	3s
Build	23s
Test	0s
Valgrind Memcheck	1s
Post Run actions/checkout@v2	0s
Complete job	0s

- Echec de la tache Valgrind Memcheck :

```

23 ==3448== by 0x10A068: main (in /home/runner/work/TP_CPP_M1/build/src/calculator_run)
24 ==3448== If you believe this happened as a result of a stack
25 ==3448== overflow in your program's main thread (unlikely but
26 ==3448== possible), you can try to increase the size of the
27 ==3448== main thread stack using the --main-stacksize= flag.
28 ==3448== The main thread stack size used in this run was 8388608.
29 ==3448==
30 ==3448== HEAP SUMMARY:
31 ==3448==    in use at exit: 1,168 bytes in 25 blocks
32 ==3448== total heap usage: 28 allocs, 3 frees, 82,064 bytes allocated
33 ==3448==
34 ==3448== LEAK SUMMARY:
35 ==3448==    definitely lost: 0 bytes in 0 blocks
36 ==3448==    indirectly lost: 0 bytes in 0 blocks
37 ==3448==    possibly lost: 0 bytes in 0 blocks
38 ==3448==    still reachable: 1,168 bytes in 25 blocks
39 ==3448==    suppressed: 0 bytes in 0 blocks
40 ==3448== Rerun with --leak-check=full to see details of leaked memory
41 ==3448==
42 ==3448== For counts of detected and suppressed errors, rerun with: -v
43 ==3448== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
44 /home/runner/work/_temp/b7bc3157-c4ba-4987-a3ba-9656cecb6c82.sh: line 1: 3448 Segmentation fault
(core dumped) valgrind ./src/calculator_run 100/2+33-7
45 Eval:
46 Error: Process completed with exit code 139.

```