Projet Compilation

Binôme : Nicolas Gourrin et Tom Agry

November 2015

1 Détail du projet

Le projet consiste à implémenter un compilateur. Le langage reconnu par ce dernier sera le langage C Map/Reduce simplifié.

Les types que nous avons choisi d'implémenter, sont les "int", les "float", et les pthread_t pour l'utilisation des threads.

Les tableaux peuvent être copié de l'un à l'autre directement, avec la condition que ce soit d'un tableau plus grand dans un plus petit.

Les mauvaises utilisations des fonctions avec une incohérence dans les types ne compileront pas non plus. Les déclarations des prototypes de fonction avec le mot clef extern devant, indique que la fonction est implémentée dans une autre fichier.

Une fonctionnalité ajoutée est la compilation en parallèle avec l'utilisation des threads.

Dans ce rapport, les fichiers de tests qui permettront d'assurer le bon fonctionnement du compilateur sont détaillés. Ainsi les tests valides sont des fichiers censés ne créer aucune erreur une fois compiler, et les tests non valides sont censés créer une erreur et exactement celle détaillée ci-dessous.

2 Tests valides

Test 1

Teste si, lorsque l'on copie un tableau plus grand dans un tableau plus petit, le code compile.

Test 2

Teste toutes les affections de variables possibles, ainsi que la déclaration des variables et leur initialisation.

Test 3

Teste la compilation de la fonction "reduce" et son bon fonctionnement.

Test 4

Teste la compilation d'une boucle "for" avec une déclaration de la variable en dehors de la boucle.

Test 5

Teste la compilation du mot clef "if" avec et sans les "else". Ce fichier teste aussi la conditionnelle "if else".

Test 6

Teste la compilation de la boucle "while" .

Test 7

Teste la déclaration des fonctions externs qui sont dans d'autres fichiers. Le fichier compile.

Test 8

Teste la compilation pour la fonction "map".

Test 9

Teste le retour de tableau par une fonction.

Test 10

Teste la compilation de l'expression d'égalité.

Test 11

Teste la compilation de l'expression d'inégalité.

Test 12

Teste la compilation de l'expression de comparaison "ET".

Test 13

Teste la compilation de l'expression de comparaison "OU".

Test 14

Teste la compilation pour créer des threads, donc l'utilisation de la fonction "pthread_create", en utilisant la bibliothèque "pthread".

Test 15

Teste la compilation pour l'affectation d'un pointeur de fonction sur un autre.

3 Tests non valides

Test 1

On essaie de copier un tableau plus petit dans un tableau plus grand. Le test ne compile pas pour le compilateur gcc.

Test 2

On initialise un pointeur de fonction sur une fonction existante et l'on essaie de modifier la valeur du pointeur. Le code ne compile pas avec le compilateur gcc car la modification du pointeur est interdite.

Test 3

On essaie de définir une fonction dans une autre fonction (la fonction main). Le code ne compile pas car la fonction est déclarée à l'intérieur d'une autre fonction.

Test 4

On essaie d'utiliser une variable qui n'est pas déclarée. Le code ne compile pas, la variable n'est pas déclarée.

Test 5

On essaie de déclarer deux fois une variable. Le code ne compile pas car il est impossible de déclarer plusieurs fois la même variable.

Test 6

On essaie de déclarer deux fois un nom de variable avec un type différent. Le code ne compile pas, du au fait qu'il y ait une déclaration multiple de la variable.

Test 7

On essaie d'appeler une fonction mais les types des paramètres ne sont pas ceux attendus par la fonction. Le code en c compilerait avec des warnings pour le compilateur gcc, mais pas pour notre compilateur.

Test 8

On essaie de récupérer le retour d'une fonction avec une variable ayant le mauvais type. En utilisant gcc, le code en c compilerai avec des warning, mais il ne compilera pas avec notre compilateur.

Test 9

On essaie de mettre un mot clef "else" sans mettre le mot clef "if" plus tôt. Le code ne compile pas car l'utilisation de "else" sans "if" est non autorisée.

Test 10

On essaie d'utiliser la fonction "map" avec un tableau vide. Le code ne compilera pas car on ne peut pas passer un tableau vide en argument de la fonction map.

Test 11

On essaie d'utiliser la fonction "reduce" avec un tableau vide. Le code ne compilera pas car on ne peut pas passer un tableau vide comme argument de la fonction reduce.

Test 12

On essaie d'utiliser les threads sans inclure la bibliothèque "pthread". Le code ne compile pas, la référence vers thread est non définie.

Test 13

On essaie de compiler un fichier qui ne contient pas de fonction main à l'intérieur.

Tests 14 à 17

On essaie de faire des opérations entre tableaux autre que l'affectation. C'est-à-dire, on teste l'addition, la soustraction, la multiplication et la division.

Tests 18 à 22

On essaie de compiler alors qu'il manque un ";", une parenthèse ou des crochets au début ou à la fin d'une fonction.

Tests 23 à 26

Pour un boucle for, on essaie de compiler alors que l'affectation propre à la boucle est mal déclarer. Il manque un ";" dans les deux premiers tests et il y a une mauvaise incrémentassions sur la valeur dans les deux autres.

4 Conclusion

Il nous a semblé utile de faire des fichiers tests pour les erreurs d'inattention tel que l'oublie de parenthèses, ou d'accolades, comme de ";". Ces fichiers nous permettront de vérifier l'exactitude du résultat de notre compilateur. Des fichiers permettant de tester les opérations ont aussi été implémentés. Ils permettent de garantir le résultat, suite à un calcul.

Une fois les tests de bases effectués, il est possible de faire des tests plus globaux. Ainsi les boucles, les conditionnelles et les fonctions vont être testées pour que le compilateur détecte toutes les erreurs qu'elles peuvent contenir.

Pour finir, les tests des fonctions spéciales et de la correspondance entre les bibliothèques sont implémentés. Il a été décidé d'optimiser le compilateur pour lui permettre de s'exécuter en parallèle sur deux parties différentes. Cette partie sera possible grâce à l'utilisation des threads.