UNIVERSITÉ DE BORDEAUX

INF466

Rapport du Projet de Compilation

Auteurs :
- Henri Frank ANABA
- Mohamed LAOUBARDY
- Yassine BENZINEB
- Yazid HASSAINE

Responsables :
M. Alain GRIFFAULT
M. Lionel CLÉMENT

11 décembre 2014

Table des matières

INTRODUCTION

Le but de ce projet était de mettre en place un compilateur pour un langage procédural déclaratif. Pour bien commencer, on a créé un dépôt svn qui a permis de travailler en équipe. Après cela nous nous sommes réunis pour implémenter les analyseurs lexical et syntaxique, afin de spécifier et de mettre en place une base solide qui soit comprise par tous les membres de l'équipe.

1 BILAN

1.1 BILAN COLLECTIF

Au départ, nous avons établi des règles afin de limiter les erreurs de compatibilité entre les différentes versions de chaque membre. Ainsi, nous avons désigné un membre «chef» qui va valider le travail des membres avant de faire un commit. Nous avons aussi implémenté ensemble les interfaces pour être d'accord sur la signature des fonctions et mettre en place le principe de polymorphisme. Durant ce projet plusieurs outils ont été utilisés comme TeamViewer, Skype, ou Dropbox (pour le partage de fichier non validé).
Le sujet prévoyait la répartition des tâches entre les 4 participants. Toutefois, étant donnée la dépendance entre celles-ci, nous avons préféré travailler ensembles. Nous avons mis en place une grammaire minimale pour pouvoir réaliser le sujet dans son intégralité.

1.2 BILAN INDIVIDUEL

1.2.1 Henri Frank ANABA

Ayant rejoint le groupe une semaine après le début du projet, mon travail se devait d' être précédé d'un état de l'art. Ainsi, après avoir fait le point sur ce qui avait déjà été mis en place par mes collègues, Il fallait comprendre l'implémentation pour intégrer plus aisement ma contribution. Cette étape de compréhension a été accélérée par le fait que l'architecture du projet étant quasiment la même que celle des TP.
Au cours de mon étape de compréhension, je me suis occupé des questions indépendantes du sujet qui n'avaient pas encore été faites par mes collègues. Ainsi j'ai mis en place le dépot pour le projet (création du dépot svn avec savane). Pendant cette même étape, j'ai aussi participé à l'élaboration du contrôle de compatiblilité des types lors des opérations sur les types et les affectations. En parallèle je participais à la conception du projet (réponse à certaines question, et élaboration de la grammaire) et à la réalisation de ceraines tâches répétitives.
Une fois mis dans le bain, je me suis occupé des questions d'optimisation car mes collègues étaient déjà partis pour répondre aux questions minimales et je trouvais que c'était la partie vraiment nouvelle entre le module d'analyse syntaxique de l'année dernière et le module Compilation de cette année. Ainsi, j'ai permis que tous les appels de fonction dont le résultat n'était pas stoqué dans une variable, soit mis dans une variable temporaire La séquence ESEQ(MOVE(TEMP t), CALL (f, el)). Cette question d'optimisation étant

liée à la suppression de tous les noeuds ESEQ , j'ai aussi intégré le code qui permet de faire cette suppression mais je l'ai mis en commmentaire pour que l'on puisse bien voir le résultat de la première optimisation.
J' ai aussi passé beaucoup de temps à faire une implémentation de la fonction d'unification avant qu'un collègue ne me fasse remarquer qu'elle était déjà implémentée par le professeur. Puis à réfléchir sur la réalisation du package pour le code à trois addresses, et la réalisation de la classe pour le graphe de flot de Contrôle avant que mon collègue ne me dise tour à tour qu'il a déjà fait ces questions.
Globalement, j'ai pû tirer des enseignements outre en compilation et en programmation, sur la conduite de projet : Pour un travail efficace et efficient, l' état de l'art doit être réalisé rigoureusement et la collaboration passe par une communication effective.

1.2.2 Mohamed LAOUBARDY

Mon travaille a consisté tout d'abord à compléter le fichier build.xml pour avoir une compilation parfaite (ant all) ; puis un nettoyage à utiliser avant l'archivage (make clean) pour pouvoir enfin tester le projet (make test).
J'ai créé un fichier Makefile afin de convertir les fichiers .dot au format .pdf,et les ouvrir automatiquement.
J'ai implementé l'analyseur lexial (Flex) et l'analyseur syntaxique (CUP) en prenant les versions déja fournites dans les TP précendents et en ajoutant des nouveaux régles qui correspondent au langage déclaratif.
Ensuite,j'ai développé une classe Main.java,qui contient l'ensemble des appels qui permettent de produire le code intermédiaire ,de l'optimiser ,de lui faire produire le code à trois adresses. J'ai developpé l'ensemble des classes dans le package stree pour toutes les expressions et instructions du langage,j'ai aussi implementé une classe pour les arbres binaire de recherche et une autre pour la pile des environnements dans le package env.
Esuite,j'ai modifié le fichier Cup afin d'ajouter pour chaque déclaration de variable,de type,de fonction le nom et le type associé dans l'environnement courant. J'ai mis en place l'ensemble des classes dans le package itree pour toutes les expressions et instructions du code intémédiaire ,J'ai exploité les notions des conceptions pour fournir des classes coéhrentes (toutes les classes hérite de la classe Itree), pour cela j'ai implementé des interfaces qui fournissent les traitements que les autres membres de groupe doivent implementer J'ai aussi développé une classe FlowControl qui implémente un graphe de flot de contrôle où chaque sommet correspond à un ensemble d'instructions et chaque arc à un saut implicite ou explicite d'une séquence d'instructions à une autres,mais par manque de temps j'ai pas intégré

cette classe dans le projet,j'ai met en oeuvre un algorithme qui permet de détecter les boucles dans le graphe de flot de contrôle. Enfin,j'ai developpé une classe qui implemente les quadruplets qui contiendront le code à trois adresses ainsi que la méthode toString() permettant d'afficher ce code.

1.2.3 Yassine BENZINEB

Durant la période du projet, j'ai appris beaucoup des techniques concernant la compilation et qui m'a aider pour la participation au projet. Malgré tout, c'était un peu déficile pour moi parce que c'était la première fois que je faisait de la compilation (Analyses lexicale, syntaxique et sémantique entre autres).

Pour la première étape qui est l'analyse lexical,(Lexer) j'ai participé pour mettre en ouevre un analyseur lexical pour langage procédural comme demandé dans l' énoncé. Ensuite dans la partie analyse syntaxique(Cup), j'ai défini des régles de grammaire avec à chaque fois une phase de test pour que la fonctionnalité rajoutée soit validée et n' apporte pas des problèmes pour la suite du projet.

Pour gérer la validité des opérations binaire(BINOP), j'ai assuré le principe de compatibilité afin que chaque opération entre deux types différents lève une exception.

1.2.4 Yazid Hassaine

J'ai participé à la réalisation du projet par plusieurs optimisation pour perfectionner la gestion des types dans plusieurs classes de Stree, ceci pour, mieux gérer les erreurs et régler différents cas particuliers, j'ai participé à la conception de la classe Stree, on ajoutant par exemple, une fonction qui permettait d'utiliser les mêmes label dans StreeIF ou StreeWhile. J'ai par ailleurs, réglé quelques problèmes dans le toDot().

2 LIMITATIONS

Notre langage reste minimal. Nous nous sommes focalisés sur un groupe de fonctions et de traitements ; faute de temps , nous n'avons pas fait les autres. Ainsi, les instructions suivantes ont été implémentées :

- l' affectation
- les opérations arithmétiques et logiques
- la déclaration des variables et des fonctions
- l'appel de fonctions
- les branchements conditionnels
- les boucles
- les opérations sur les pointeurs.

3 TESTS

Pour tester les fonctionnalités de notre langage, on a créé le fichier Input.txt qui contient toutes les instructions fonctionelles. La compilation de ce fichier génère les fichiers suivants :

- AST.pdf : représenatant l'arbre de syntaxe abstraite du langage.
- IR.pdf : représentant l'arbre de code intermédiaire.
- 3_adress_code.txt : représentant le code à trois adresse.
- env.pdf : représentant l'environement.

Nous avons aussi des fichiers d'erreurs pour aider les utilisateurs du langage au débogage :

- error.txt : Contenant les erreurs (variable non déclarée, type incompatible).
- Output.txt : Contenant les erreurs de syntaxes.