Bilan individuel de compétences

Projet GL Équipe 16
Youness Lehdili
27 janvier 2021

Contents

1 Introduction **1**

2 Mettre en œuvre des processus de validation/Agir en professionnel responsable **1**

1 Introduction

Au début, J'ai essayé de passer le temps à faire une documentation afin de d'avoir une vision globale du projet, mon objectif était de comprendre le fonctionnement d'un compilateur dans sa globalité. Le projet est trop vaste pour etre compris au premier abord, mais cette stratégie s'est avéré payante sur le long terme. Avec le recul, je pense que la spécialisation rapide des membres de l'équipe est une bonne méthodologie pour la suite du projet, ca m'a permis d'optimiser le temps de travail et paralléliser les taches.À mon avis, le sujet était intéressant,il m'a permis de comprendre les différentes étapes de vérification du code. Aprés 2 semaines de travail sur l'etape A, J'ai décidé de rejoindre Amine afin d'avancer plus rapidement sur l'extension, cette partie du projet nécessite une documentation générale des algorithmes usuelles de la classe math,on doit vraiment se débrouiller tout seul. et réfléchir à comprendre des méthodes qui sont vaguement expliquées dans les documents, on est obligé de tester plusieurs algorithmes pour voir ce qu'il marche le mieux.Enfin, le fait de ne pas nous donner de doc dessus nous donne une première impression de ce que peut être le travail d'un ingénieur en entreprise.

2 Mettre en œuvre des processus de validation/Agir en professionnel responsable

J'ai commencé au début à travaillé sur l'étape A , j'ai implémenté une partie du Lexer et du Parser, ce travail nécessite une communication avec les autre membres afin de se mettre d'accord sur les structures de donnée à utiliser, leur montrer brièvement le travail accompli est répondrez à leur questions.
Les tests du lexer ne comportent généralement qu'un caractère ou un mot, reconnu par le langage comme un token. Le but est de couvrir l'ensemble des caractères et de vérifier la globalité des règles du lexer.
Les tests syntaxiques sont fait de sorte à couvrir la majorité des cas d'erreur syntaxique : mauvais usage des opérandes (opérande manquant ou mal placé),

oubli d'un caractère entraînant une erreur de syntaxe (parenthèse, point-virgule, virgule ...), assignation incomplète, mauvais usage des conditions et des instructions dans les boucles (else avec une condition, while sans instruction ...).La base de tests est également construite de manière à vérifier le plus de programmes syntaxiquement corrects possibles.
le projet GL m'a permis d'acquérir les compétence nécessaire pour élaborer une infrastructure d'automatisation des tests,écrire des codes shell afin de vérifier tout les tests d'un seul coup.
par exemple ,j'ai implémenter les fichier shell $synt-test-inv.sh$ et $lex-test-inv.sh$,et ajouté leurs blocks dans le fichier pom.xml afin de faire des tests lexicales et syntaxiques des fichier qui existent dans le répertoire valid et invalid en meme temps.
J'ai ensuite travaillé sur l'extension dans laquel j'ai pu renforcer plusieurs techniques liés à la validation des tests, ainsi que l'implementation des tests validant la précision des algorithmes que j'ai étéb amener a programmer afin d'approximer les fonctions de la bibliothèque Math.
par exemple :
Afin de s'assurer de la précision de l'approximation qu'on a choisi, Nous avons confronter les résultats avec celles donnés par Java,l'objectif que nous avons fixé est d'atteindre un niveau de précision de quelques ulps ($\leq$ pour la fonction ulp(sans erreur).
Une grande partie de notre travail s'est donc portée sur la validation des tests. Pour chaque étape, des scripts de validation automatique ont été mis en place, ce qui nous a permis d'éviter d'introduire de nouveaux bugs. Cette démarche a été l'un des points forts de notre projet, puisque cela nous a permis d'avancer par incréments, et donc de gagner en agilité. C'était aussi un point positif pour le moral de l'équipe, puisqu'on voyait notre compilateur s'améliorer de jour en jour.