NEILA

Presentation + sommaire +De quoi s'agit-il ?

il s'agit d'un projet à réaliser en binôme dans le cadre de l'unité d'enseignement Compilation. Le but du projet est de créer un compilateur afin de traduire un langage source qui est un sous-langage du langage C appelé miniC en une représentation intermédiaire de sa syntaxe en arbre syntaxique codé en DOT .

Outils et ressources :

Nous nous sommes appuyés principalement sur ce que nous avons pu apprendre au cours du semestre en terme d'analyse lexicale, syntaxique et sémantique.
Les différentes ressources mises à notre disposition sur moodle à savoir les cours, corrigés des td et manuels sur lex et yacc nous ont beaucoup aidé.

ZIED

Les étapes du projet :

Avant de commencer toute analyse, il fallait d'abord comprendre ce qui était demandé et ce qu'il y avait à faire.
Nous avons commencé par lire le pdf du projet et comprendre ce que contenait le langage source, suite à quoi nous avons procédé à l'analyse lexicale avec l'outil lex.
Nous avons complété le support fourni en supprimant les tokens inutiles à notre langage et en rajoutant les tokens manquants.

Une fois l'analyse lexicale complétée, on passe au parsing afin de vérifier que notre grammaire respecte les règles syntaxiques de notre langage.

Nous avons défini les tokens et rectifié la grammaire afin qu'elle analyse notre langage comme on le souhaite en déclarant des erreurs de syntaxe quand celles-ci se produisait par exemple. Yyparse est appelée afin de réaliser l'analyse syntaxique dans le main du fichier yacc.

NEILA

Vient ensuite le tour de l'analyse sémantique qui s'assure que les éléments de la syntaxe du code source ont un sens dans notre langage.
Nous avons d'abord créer deux tables de symboles nécessaires à l'analyse sémantique, l'une pour les variables de notre langage et l'autre pour les fonctions. Nous avons choisi les tables de hachage comme structure de données d'abord parce que nous avons pensé que c'était l'idéal dans le cadre de notre projet, ça nous permet une recherche plus facile des éléments et ensuite parce que nous sommes plus familier avec, nous les avons vues pendant les travaux dirigés. Notre analyse sémantique prend en compte la vérification des types et des redéclarations de fonction et des variables et gère les erreurs de ce côté-là.

Concernant la génération du code DOT, nous avons formé un arbre abstrait de nœuds, chaque nœuds est créé grâce à une fonction create_node et sont stockés sous la forme d'une structure qui contient la forme, le label, la couleur etc conformes à leurs descriptions dans le fichier pdf du projet. Une fois que la liste de nœuds est formée, la fonction print_dot génère le fichier représentatif de l'arbre syntaxique correspondant au code source de notre compilateur en transformant la structure des noeuds en code DOT.

ZIED

NOTRE ORGANISATION :

L'intégralité du projet, c'est-à-dire le code, le rapport ainsi que les slides de la présentation, ont été réalisés par nous deux en présentiel.
Au début, on se voyait 2 à 3h par semaine, mais ces dernières semaines nous avons augmenté notre rythme de travail jusqu'à se voir 7 voir 8h par jour en applicant la méthode pomo doro pour ne pas se surcharger.
Nous nous sommes partagé le travail équitablement, par exemple dans le fichier lex, la moitié des tokens ont été fait par zied et l'autre moitié par moi, idem pour les tables de symboles, je me suis chargé de faire celle des variables et lui celle des fonctions, et ainsi de suite pour le reste.
Si l'un bloquait sur quelque chose l'autre était là pour l'aider.

NEILA

DIFIICULTÉS :

Les principales difficultés rencontrées était les parties de l'analyse sémantique et de la génération de code DOT, nous avons mis du temps à comprendre comment procéder et à mettre un plan pour cela, nous avons bien évidemment susciter l'aide de nos camarades pour une meilleure compréhension des spécifications du PDF par rapport aux arbre représentative.
Nous avons également rencontré plusieurs erreurs de typage , de syntaxe et d'inclusion lors du test de notre code, nous avons réussi à toutes les rectifier même si ce n'était forcément pas évident pour certaines d'entre-elles.

ZIED

CE QUI A ETET FAIT ET RESTE 0 AMELIORER :

Nous n’avons aucune erreur lors de l’exécution de notre code, nous avons également un fichier dot qui se crée et du code DOT qui est généré mais ce code est malheureusement incomplet, il faut donc l’améliorer et le rectifier.
Nous avons quelques erreurs syntaxiques qu’on n’a pas pu réglé, on a essayé, mais vu qu’on a testé notre code en dernier après avoir fini la totalité des fichiers, on manquait un peu de temps pour pouvoir les gérer, nous n’avons pas réussi à tester tous les tests fournis dans le projet car dans certains il y a l’erreur « syntaxe error » qui s’affiche, mais malgré ceci nous avons quand même pu générer du code dot.

NEILA

CONCLUSION :

Ce projet nous a permis en premier lieu de développer nos compétences techniques, à comprendre plus précisément ce que fait un compilateur et les différentes étapes nécessaire à sa création, ainsi que l’amélioration de notre compréhension de la gestion mémoire en langage C ce qui n’est vraiment pas négligeable.

ça nous a également permis d’avoir une idée plus précise sur les compilateur et surtout développer un intérêt pour la compilation.

Nous avons fait de notre mieux pour ce projet et nous sommes assez contents de ce que nous avons pu rendre comme produit final par rapport à ce qu’on passait pouvoir rendre et malgré que plusieurs améliorations sont encore à prévoir.