Projet de compilation

Licence d’informatique

—2020-2021—

Sommaire :

I Présentation

Dans ce projet, l'objectif est de réaliser un compilateur à l'aide de flex et bison pour le langage TPC .Ici nous traiterons plus particulièrement le typage et la traduction.

Nous avons laissé un fichier script.sh qui permet de lancer tout les tests des dossiers correct, error et warnings, il ne lance cependant pas l'éxécution du fichier asm.

Pour lancer le programme sur votre fichier, placer vous dans le répertoire bin et lancer la commande :

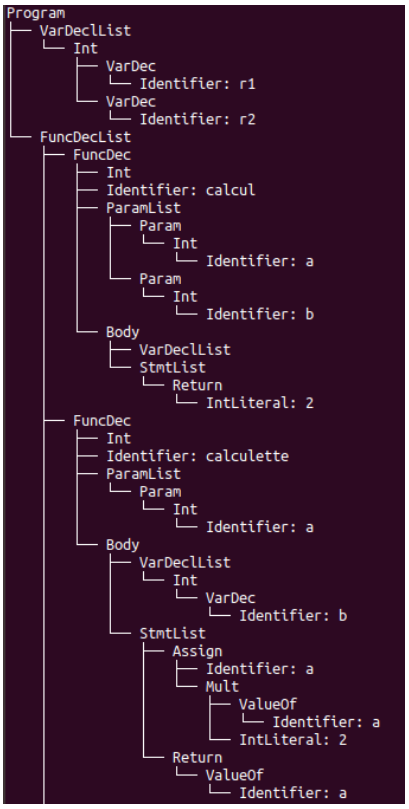
./compilation [-o prog.asm] < prog.tpc

où prog.tpc est le fichier que vous souhaitez analyser.

II Décomposition de projet

I. Arbre abstrait

Lors de l'étude lexicale du fichier étudié, nous avons créé un arbre contenant toutes les instructions du fichiers TPC analysé permettant de stocker les variables, identifiant et autres. Cette arbre va nous permettre ensuite de construire la table des symboles



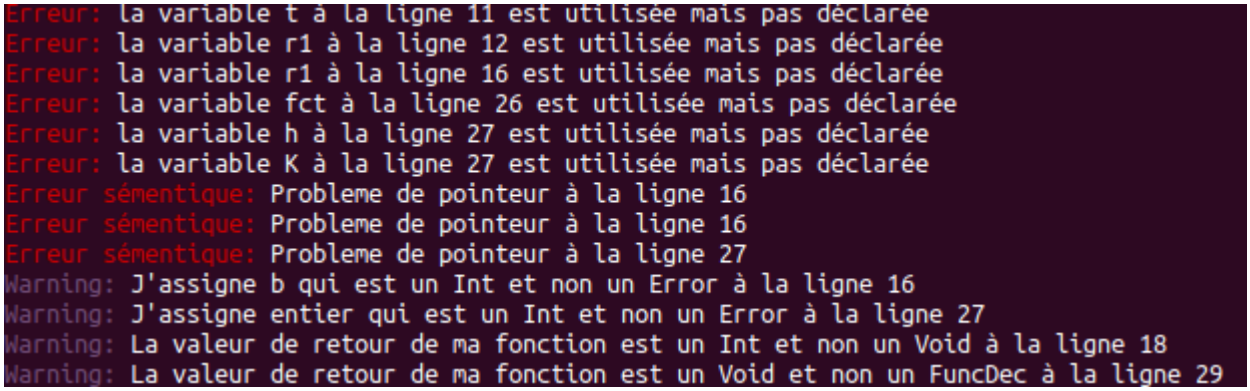
II. Table ses symboles

Nous avons dans un second temps construit la tables des symboles lors d'un premier parcours de notre arbre. Cela permet de donner des adresses à chaques variables déclarés et aussi de retenir leur type, leur portée, leur nom.. Mais également de vérifier que toutes les variables utilisées sont déclaré. La redéfinition de variables est aussi détecté grâce à cette table.

Niveau	Classe	name	lineno	type	pointeur	nbr arg	adresse
0							
	Global	r1	2	Int	0	0	8
	Global	r2	2	Int	0	0	16
1							
	Fonction	calcul	4	Int	0	2	0
	Argument	a	4	Int	0	0	-8
	Argument	b	4	Int	0	0	-16
2							
	Fonction	calculette	11	Int	0	1	0
	Argument	a	11	Int	0	0	-8
	Local	b	12	Int	0	0	-16
3							
	Fonction	test	17	Int	0	1	0
	Argument	q	17	Int	0	0	-8
	Local	t	18	Int	0	0	-16
	Local	p	19	Int	0	0	-24
4							
	Fonction	main	27	Int	0	0	0
	Local	a	28	Int	0	0	-8
	Local	b	28	Int	0	0	-16
	Local	d	28	Int	0	0	-24
	Local	c	29	Int	0	0	-32

III. Typage

Une fois la table terminée, les seuls erreurs non vérifiée sont les erreurs de type. Dans cette partie nous avons vérifié que les types lors des assignations, des retours ou appel de fonctions, ect sont corrects. Dans le cas inverse, nous avons levé des erreurs ou des warnings. Les adresses et les pointeurs sont gérés au niveau du typage. Nous avons également profité de cette partie pour vérifier qu'une fonction main existe dans le programme.



IV. Traduction

Nous avons terminé ce projet par la traduction. Nous avons réussi à gérer les appels de fonction, les déclarations de variables gloabales et locales, les readc et reade, les print. Cependant nous n'avons pas eu le temps de traiter les adresses et pointeurs qui est la seule partie du sujet non traité.

Cette partie est lancé uniquement si le programme ne comporte pas d'erreur. Cependant nous avons laissé le programme traduire le fichier tpc s'il y a des warning mais nous ne garantissons que le fonctionnement soit celui attendu.

V Test

Nous avons également préparé trois dossiers de test différents : correct, error et warning. Dans le dossier correct, il y a que des fichiers tpc correct, le code nasm sera donc créé. Dans le fichier error, nous avons mis dedans des fichiers tpc contenant des erreurs, la traduction n'aura donc pas lieu. Enfin dans le dossier warning, les fichiers seront traduit en nasm mais n 'auront pas forcément le comportement souhaité.

III Difficultés rencontrées

Dans ce projet, nous n'avons pas pu le traiter entièrement par manque de temps car nous avons passé beaucoup de temps sur les étapes précédentes. La partie non traitée est la partie traduction des adresse et pointeurs.