RIBEIRO Ghyslaine
NEANG Thomas
L3 Info Groupe 1

SOMMAIRE

1 Mode d'emploi

2 Table des symboles

3 Attribut et fonctionnement du compilateur

4 Problèmes rencontrés

1 Mode d'emploi

-Se rendre dans le répertoire principale à partir du terminal et taper make. Un executable appelé tcompil sera créer dans ce répertoire ainsi que dans le repertoire EXEMPLE qui contient quelques codes sources afin de tester la compilation.

Des fichiers déjà compiler à partir de tcompil se trouve dans le répertoire principale et sont prêt à être donner à la vm.

- Le compilateur écrit directement sur le terminal donc le résultat est à rediriger vers un fichier manuellement.

Exemple : ./tcompil operation > resoperation
./vm686 resoperation

2 Table des symboles

Pour le projet nous avons définie une structure symbole qui contient un champ type et plusieurs champs de valeur qui sont remplie en fonction du type.

Pour le champ type la valeur 0 correspond au type entier et 1 correspond à un caractère.

Nous avons ensuite déclarer un tableau de symbole de taille fixe et fais quelques fonctions pour manipuler ce tableau (fonction d'ajout, recherche, mis à jour...).

3 Attribut et fonctionnement du compilateur

Pour l'analyseur syntaxique nous avons simplement décrit tous les motifs du langage et renvoyer le token correspondant comme dans le sujet.
Et nous renvoyons une erreur si aucun motif ne correspond.

Nous avons décider de mettre un champ opération dans l'union pour attribuer une valeur aux opérations afin que l'analyseur syntaxique puisse les différenciers.
ADDSUB :

- 0 pour +
- 1 pour -

DIVSTAR :

- 0 pour *
- 1 pour /
- 2 pour %

COMP :

- 0 pour ==
- 1 pour !=
- 2 pour <
- 3 pour >
- 4 pour <=
- 5 pour >=

BOPE

- 0 pour &&
- 1 pour | |

Pour les opérations « uniques » comme l'affectation ou la négation nous avons garder le token de base car cela suffit.

Nous avons ensuite définie les non terminaux Exp Litteral et Lvalue comme des symboles afin de connaître leur type et leur valeur pour mettre à jour la table des symboles.

La partie déclaration se contente de créer le symbole avec le nom de la valeur de Ident et le bon type dans la table des symboles en vérifiant qu'il n'existe pas déjà et la règle de l'affectation met à jour la table si le nom existe.

Pour la déclaration de vartiable global nous avons décidé de mettre un entier type_decl pour simuler l'attribut hérité.

Nous avons ensuite jouer sur les valeurs d'attribues des expressions pour gérer la concordence des types et l'affichage.

4 Problèmes rencontrés

Nous avons eu des soucis pour la gestions de la table des symboles car nous ne savions pas comment stocker les données et on s'est retrouvé bloquer pour récupérer la valeur du registre pour la mettre dans la table avec l'instruction READ ou même la gestion de la porté.

Il y aussi un problème pour la déclaration de variable global où nous ne pouvons pas déclarer 2 types différents à la suite car notre entier type_decl n'est pas mis à jour pendant l'analyse syntaxique. Ce qui fait que la deuxiéme déclaration est faite mais garde le type de la déclaration précédente.
Cependant les déclaration dans la fonction main marchent.

Nous avons aussi l'impression que l'affectation n'est pas considéré comme une instruction dans la grammaire. Dans un if else par exemple nous avons une instruction print et une affectation dans le if et le print est bien sauté contrairement à l'affectation qui est toujours faite.