PEREZ LEFEUVRE Vincent
QUACH Kevin

Université Gustave Eiffel 2021-2022
L3 Informatique

Rapport Projet de Compilation

Sommaire

Manuel d'utilisateur

- Pour compiler le programme, tapez **make** sur le terminal.
- Pour lancer le script de tests, tapez **bash tests.sh**, cela va produire le fichier rapport.txt contenant les résultats de tous nos tests.
- Pour lancer le programme sur un des tests présent, tapez
  ./bin/tpcc < test/good/[Nom du fichier].tpc (TPC correct)
  ./bin/tpcc < test/syn-err/[Nom du fichier.tpc (TPC incorrect)
  ./bin/tpcc < test/sem-err/[Nom du fichier].tpc (TPC avec erreur sémantique)
  ./bin/tpcc < test/warn/[Nom du fichier].tpc (TPC qui cause des warnings)
- Pour afficher l'arbre abstrait il faut rajouter -t ou --tree :
  ./bin/tpcc -t < test/good/[Nom du fichier].tpc
- Pour afficher la table des symboles il faut rajouter -s ou --symtabs :
  ./bin/tpcc -s < test/good/[Nom du fichier].tpc
- Pour afficher une description de l'interface utilisateur il faut rajouter -h ou --help: **./bin/tpcc -h**

Nos Choix

```
================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
              a |             int |               0 |             glb
         lettre |            char |               4 |             glb
         repeat |            char |               5 |             fun
            sum |             int |               6 |             fun
         equals |             int |              10 |             fun
            add |            char |              14 |             fun
           main |             int |              15 |             fun
================================================================


================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
         letter |            char |              -1 |             arg
================================================================


================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
              x |             int |              -4 |             arg
              y |             int |              -8 |             arg
================================================================


================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
              q |             int |              -4 |             arg
              s |             int |              -8 |             arg
================================================================


================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
              x |             int |              -4 |             arg
              y |             int |              -8 |             arg
================================================================


================================================================
====================== Table des symboles ======================
================================================================
 Symbole        |  Type           | Adresse         | Classe
          count |            char |              -1 |             var
================================================================
```

Un exemple des tables de symbole d’un fichier tpc

On crée une liste de table pour chaque programme TPC dans laquelle se trouve la table des symboles globaux et une table pour chaque fonction contenant les variables locales et leurs arguments. La table globale est la première table de la liste et contient les variables globales et également les symboles des fonctions du fichier. Les tables sont des listes chaînées.

Pour remplir les tables, on effectue une première exploration de l'arbre. Lorsqu'on rencontre des nœuds décrivant le type d'une variable, on sait que c'est une déclaration de variable, car on ne peut pas assigner une valeur à la déclaration en tpc. On ajoute alors cette dernière dans la table des symboles correspondante.

Durant cette première exploration, on vérifie également la présence ou non d'erreurs sémantiques ou la nécessité d'afficher un message d'avertissement. Il existe plusieurs fonctions pour vérifier ces erreurs. Par exemple, lorsqu'on ajoute des symboles dans les tables dans la fonction add_all_symbol, si le symbole est déjà présent dans la table, on affiche un message d'erreur pour redéclaration. Chacune de ces fonctions sont appelées lorsque les situations correspondantes sont rencontrées. Nous avons choisi de laisser le programme continuer après avoir rencontré une erreur sémantique. Si aucune erreur syntaxique ou sémantique n'est rencontrée durant la première exploration, on passe à la traduction du programme.

Pour la traduction, on effectue une deuxième exploration de l'arbre. Nous avons des fonctions pour traduire chaque opération (+, -, *, /, =, !, And, Or) et une fonction qui s'occupe de traduire les comparaisons. Une fonction evaluate nous permet d'évaluer des expressions de façon récursive. Elle ajoute dans la pile les variables ou valeurs rencontrées durant le parcours suffixe de l'arbre, puis les dépiles pour évaluer les expressions lorsqu'on rencontre un opérateur.

Nous avons pour chacune des instructions if (else) et while une fonction contenant sa traduction. Ces fonctions utilisent des labels, les labels ne peuvent pas avoir le même nom on ajoute donc à "label" un numéro que l'on incrémente lors de l'utilisation de if ou while. Ainsi

chacune de ses instructions à ses propres labels. Par exemple, si on utilise un while on a besoin de trois label on va donc utiliser les labels : label1, label2 et label3 puis augmenter le nombre de label de trois pour que la prochaine instruction utilise le label4, label5, etc… .

Le type énuméré Type décrit le type d'un symbole, elle peut être une variable locale (VAR), une variable globale (GLB), un argument (ARG) et une fonction (FUN). Ce type nous a facilité la tâche lors de la traduction, pour vérifier que le nombre d'argument que prend une fonction est le bon. Pour la traduction des variable globales on vérifie que c'est une globale avec GLB dans la table et on initialise les globales avec une étiquette glb dans un section .data que l'on utilise lors de l'utilisation de globale, par exemple la première variable sera dword [glb] et si la première variable globale est un entier alors la deuxième sera dword[glb + 4], etc… .

Difficultés rencontrés

- Mauvaise exploration de l'arbre au début

Notre première fonction pour l'exploration de l'arbre n'était pas bonne. On ne descendait que dans le nœud de déclaration des variables pour les ajouter dans la table des symboles ce qui ne nous permettait pas de faire les vérifications sur les erreurs sémantiques. Nous avons alors refait une fonction d'exploration, cette fois explorant l'arbre en entier.

- Table vide problème realloc

Nous avions rencontré des problèmes liés au realloc lors de la création des tables des symboles, résultant en des tables vides alors qu'il devait avoir des lignes. Certain appel récursif était de trop nous avons donc rajouté les conditions nécessaire afin d'empêcher le problème

- Switch dans le .y

Nous avons dû modifier la syntaxe du switch dans l'analyseur syntaxique car il n'acceptait pas certaine syntaxe comme la succession de 'default' par exemple)

- Traduction des conditions

Lorsque les conditions ont besoins de plusieurs comparaisons la traduction ne marche pas (par exemple:

var1 && !var2 || var3 fonctionne,
var1 > var2 fonctionne mais
var1 < var2 && var3 >= var4 ne fonctionne pas)

- Traduction du switch

Pas fait.

- Récupérer le nom du fichier

Récupérer le nom du fichier pour nommer le fichier cible s'est avéré difficile dû à la redirection. Finalement, nous n'avons pas implémenté ça. Le fichier cible s'appelle _anonymous.asm pour tous les tests.