Université de Bretagne Occidentale

Master Informatique

2019/2020

Compilation

Rapport de Projet

Réalisation d'un calculateur entiers/chaines

Auteur :
William Pensec

12 novembre 2019

Table des matières

1 Introduction

Ce rapport concerne la réalisation d'un calculateur de chaînes et d'entiers à l'aide des outils JavaCup et JFLex dans le cadre d'un projet Informatique.

Le calculateur réalisé est capable de reconnaître des entiers, chaînes de caractères, parenthèses ouvrantes et fermantes, règles de calcul (priorité dans les calculs), variables contenant des chaînes de caractère et variables contenants des entiers. Le calculateur reconnaît également la chaîne "PRINT" comme étant un signal permettant d'afficher à l'écran toutes les variables stockées et leurs contenus. Aussi, il est possible de quitter le calculateur à tout moment à l'aide de la commande "ctrl + d". Le calculateur est capable de reconnaître des symboles n'appartenant pas au langage de ce dernier, mais également de signaler des erreurs éventuelles de syntaxe et reprendre son exécution par la suite.

Ci-dessous seront expliqués plus en détails les choix d'implémentation de ce calculateur, la grammaire mais également la table de symboles qui a été utilisée ainsi que les erreurs et améliorations possibles.

2 Définition de la grammaire

```
liste : := expr
        | { : System.out.print("Analyse finie. "); System.out.println("Au
revoir !"); :}
        ;

expr : := entier :res NL liste
        | chaine :val NL liste
        | aff_entier NL liste
        | aff_chaine NL liste
        | print NL liste
        | error NL liste
        | erreur NL liste
        ;
```

3 Description de la table des symboles

Symbole	**Éléments reconnus**
PRINT	PRINT
NB	[0-9]+
STR	\" [^\"]* \"
PLUS	"+"
FOIS	"*"
EQUAL	"="
PO	"("
PF	")"
VARE	[a-zA-Z0-9]+
VARC	"$"[a-zA-Z][a-zA-Z]*

TABLE 1 – Table des Symboles

4 Explication des choix réalisés

La grammaire réalisée a été implémentée à l'aide de 2 non terminaux de base : LISTE et EXPR. Le premier (liste) permet de faire boucler le programme jusqu'à la fin de l'entrée utilisateur (commande : "ctrl + d") et attend un nombre infini de ligne.

Le deuxième terminal (expr) permet de différencier un entier, une chaine, une affectation de chaîne ou une affectation d'entier mais également une erreur (déclarée ou non déclarée) et enfin la commande PRINT permettant d'afficher le contenu de la HashMap permettant de sauvegarder en mémoire les différentes variables.

Il y a par la suite 6 non terminaux permettant de chercher une erreur (erreur), d'afficher la HashMap (print), d'effectuer un calcul (entier), de faire une affectation d'entier (aff_entier), de faire une affectation de chaîne (aff_chaine), de faire un calcul de chaîne (chaine). Chacun de ces non terminaux appelle soit un autre non terminal soit un terminal afin de chercher les symboles reconnus par le langage ainsi que le retour à la ligne.

Concernant la persistance des calculs on utilise une HashMap afin d'ajouter/modifier d'éventuelles variables. L'analyseur syntaxique prendra soin de vérifier la présence ou pas d'une variable dans cette HashMap avant d'effectuer un calcul qui implique une variable. Afin d'afficher le contenu de la HashMap, j'ai créé 2 fonctions d'affichage qui parcourent la HashMap et affiche le contenu clé par clé.

La gestion des erreurs est faite de telle sorte que l'analyseur lexical peut détecter la présence d'un caractère ne faisant pas partie du langage et signaler sa présence. Aussi, le programme détecte les erreurs syntaxique de 2 manières différentes, soit il manque une opérande dans l'expression et on le précise, soit il s'agit d'une autre erreur syntaxique et on traite les autres cas comme un cas général d'erreur syntaxique.

5 Justification du fichier de test

Les fichiers de tests couvrent tout les cas analysés par l'interpréteur ainsi que quelques erreurs et symboles non reconnus.

Le premier fichier de test permet de tester une série d'action complètement compatible avec le calculateur. Il en renvoi aucune erreur lors de l'exécution. Voici la trace d'exécution :

```
5+6
 - : entier 11
"abc"+2*"uv"
 - : chaine "abcuvuv"
$a="abc"
 $a: chaine "abc"
$a*3
 - : chaine "abcabcabc"
b=(4+2)*3
b: entier 18
$b="def"
 $b: chaine "def"
$a+$b
 - : chaine "abcdef"
a=5
a: entier 5
c=a+5
c : entier 10
d=a*b
d : entier 90
$a+$b*2
 - : chaine "abcdefdef"
PRINT
 Variables chaînes :
        $a :  chaine "abc"
        $b :  chaine "def"
 Variables entières :
        a :  entier 5
        b :  entier 18
        c :  entier 10
        d :  entier 90
Analyse finie. Au revoir !
```

FIGURE 1 – Trace d'exécution du fichier input

Le deuxième fichier de test permet de tester une série d'erreur déclarée et non déclarée (syntax error) dans le calculateur. Voici la trace :

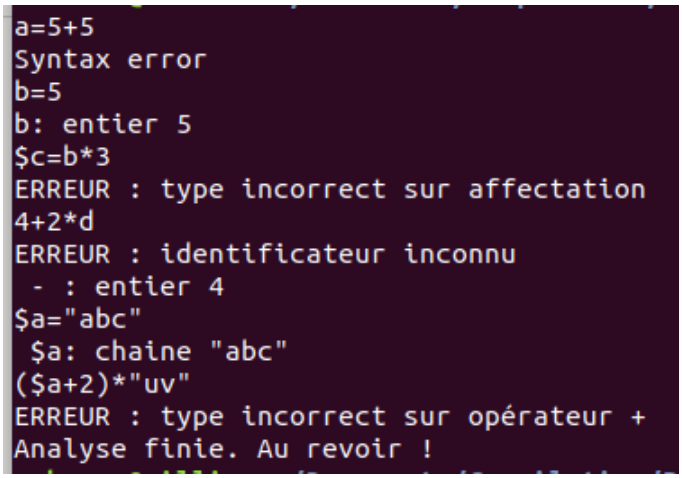

FIGURE 2 – Trace d'exécution du fichier input1

Le troisième fichier de test permet de tester une série de symboles non reconnus par le calculateur. Il y a différents positionnement des symboles ainsi que des doubles symboles non reconnus. Voici la trace :

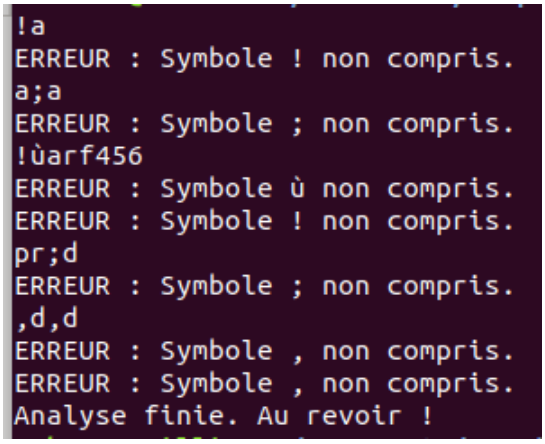

FIGURE 3 – Trace d'exécution du fichier input2

6 Présentation des erreurs et limitations éventuelles & Améliorations possibles

6.1 Erreurs

Le programme effectue les opérations demandées et des opérations supplémentaires comme par exemple addition de chaîne stockées dans des variables ou additions et multiplications de variables d'entiers.

Malheureusement, certaines fonctionnalités comme par exemple :

— a = 5+5 ne fonctionne pas pour des raisons que je ne comprend pas,

— quand j'insère un fichier en entrée directe du calculateur via la commande : "java parser < input" et bien les entrées écrites dans le fichier renvoient des erreurs mais s'exécutent normalement

Le calculateur reconnaît les symboles non valides ou une syntaxe incorrecte mais renvoi dans la plupart du temps "syntax error". J'ai mis des erreurs générales comme par exemple type incorrect sur l'addition, affectation, ou encore un identificateur inconnu. Je n'ai pas réussi à mettre exactement ce qu'était l'erreur mais au moins on a une idée de l'erreur sans savoir ce qui pose problème dans l'entrée.

6.2 Améliorations possibles

Dans l'état actuel du calculateur, on précise que certaines erreurs mais la plupart des erreurs sont des cas généraux et ne sont indiquées que par le "syntax error". Ainsi, il faudrait apporter plus de cas différents d'erreurs ainsi que mettre dans l'erreur ce qui pose problème.

Il faudrait, par ailleurs, mettre plus de cas de calcul.