Université de Bretagne Occidentale

Faculté des Sciences & Techniques

Projet de compilation

Réalisation d'un interpréteur avec entiers et tableaux d'entiers

Présenté par:

Boulmaali Linda Imene.

2023/2024

Introduction

Dans ce projet, l'objectif est d'utiliser les outils de compilation JFlex et Java Cup pour construire un compilateur capable de comprendre et d'évaluer des expressions manipulant des entiers et des tableaux.

La calculatrice développée acceptera des expressions comprenant des opérateurs d'addition et de multiplication, des tableaux définis entre accolades, des opérations de concaténation sur les tableaux, et des affectations de variables, à la fois pour les entiers et les tableaux. De plus, une commande d'affichage (PRINT) permettra d'afficher l'ensemble des variables définies avec leurs valeurs.

Le langage s'appuiera sur une grammaire spécifique pour définir la structure syntaxique des expressions acceptées par le compilateur. La grammaire comprend des règles pour la manipulation des entiers, des opérations arithmétiques prioritaires, la déclaration de tableaux, la concaténation, ainsi que la gestion des erreurs pour renforcer la robustesse du compilateur.

Squelette de la grammaire

G = <T, N, P, S>
= < (OP_PLUS, OP_FOIS, PAR_OUV, PAR_FERM, ANTI_SLASH, SEP, IDENT, TABLEAU, VIRG, AFFECT, CONCAT, PRINT, ENTIER), (expression, initial, decla, affect, instruct, concat, declaration, tab, errors), P, expr>

P:
expr → instruct ANTI_SLASH expr | e ;

instruct →affect | expression | declaration | concat | PRINT | errors | e;

//===1 Les affectations
affect → IDENT AFFECT expression

Example:

//===2 Les opération arithmétiques
expression → expression OP_PLUS expression
expression → expression OP_FOIS expression
expression → AR_OUV expression PAR_FERM
expression → ENTIER
expression → IDENT
expression → tab

expression → error

//===3 La déclaration des tableaux
declaration → TABLEAU AFFECT concat

//===4 Traitement des cas de concaténations
concat → initial CONCAT initial
concat → initial CONCAT initial CONCAT concat
concat → initial CONCAT TABLEAU
concat → initial CONCAT TABLEAU CONCAT concat
concat → TABLEAU
concat → initial

initial → SEP decla SEP
decla → expression VIRG decla | expression
tab → TABLEAU SEP expression SEP | initial SEP expression SEP
//===5 Traitement des cas d'erreurs:
errors →TABLEAU OP_FOIS expression
errors →expression OP_FOIS TABLEAU
errors →TABLEAU OP_PLUS expression
errors →expression OP_PLUS TABLEAU
errors →TABLEAU OP_FOIS errors
errors →expression OP_FOIS errors
errors →TABLEAU OP_PLUS errors
errors →expression OP_PLUS errors

Description de la grammaire

- **Non-terminaux :**
  - expr: Représente une expression globale.
  - instruct: Représente une instruction, qui peut être une affectation, une expression, une déclaration, une concaténation, une instruction d'affichage (PRINT), ou une erreur.
  - affect: Représente une affectation d'une expression à une variable.
  - expression: Représente une expression arithmétique ou une variable.
  - declaration: Représente la déclaration d'un tableau.
  - concat: Représente une opération de concaténation de tableaux.
  - initial: Représente une initialisation de tableau dans la concaténation.
  - decla: Représente la déclaration d'une expression dans un tableau.
  - tab: Représente un tableau dans une expression ou une concaténation.
  - errors: Représente une expression incorrecte ou mal formée.
- **Terminaux :**

    - OP_PLUS, OP_FOIS: Opérateurs d'addition et de multiplication.
    - PAR_OUV, PAR_FERM: Parenthèses.
    - SEP: Séparateur (utilisé dans la déclaration des tableaux et l'initialisation des tableaux).
    - IDENT: Identificateur de variable.
    - TABLEAU: Mot-clé pour représenter un tableau.
    - VIRG: Virgule.
    - AFFECT: Symbole d'affectation.
    - CONCAT: Symbole de concaténation.
    - PRINT: Mot-clé pour l'instruction d'affichage.
    - ENTIER: Représente un entier.
    - ANTI_SLASH: Symbole antislash pour l'échappement.
- **Production :**
    - expr est construit à partir d'instructions suivies de symboles antislash ou une epsilon (e).
- **Les affectations :**
    - Une affectation se compose d'un identificateur suivi du symbole d'affectation et d'une expression.
- **Les opérations arithmétiques :**
    - Une expression peut être une addition, une multiplication, une parenthèse, un entier, un identificateur ou une erreur.
- **La déclaration des tableaux :**
    - Une déclaration de tableau se compose du mot-clé TABLEAU suivi d'une concaténation.
- **Traitement des cas de concaténations :**
    - La concaténation peut être entre deux éléments initiaux, entre deux éléments initiaux concaténés à d'autres, entre un élément initial et un tableau, ou entre un élément initial et un tableau concaténé à d'autres.
- **Traitement des cas d'erreurs :**
    - La grammaire prend en compte plusieurs cas d'erreurs où des opérations incorrectes ou mal formées sont détectées.

Limitations eventuelles

La grammaire ne peut pas traiter des cas comme ceci:

- @p = {1, {3,4,5}[{0,1}[0]]}

Le traitement de plusieurs séparateurs imbriqués, constituant le tableau et l'indice, provoquent un conflit. En effet cette instruction devrait donner @p = {1, 3} mais elle donne

seulement {3}. Le traitement sémantique de cette instruction provoque plusieurs erreurs de compilation.

Contrairement à l'instruction @p = {{3,4,5}[{0,1}[0]], 5, 8} qui donnera @p = {3, 5, 8} .

- On ne pourra pas initialiser des tableaux vide, donc l'instruction @b = {} est fausse. un tableau contient au moins un valeur.

Les erreurs du programmes

Le code répond à toutes les instructions demandées. Toutes les erreurs de compilation, rencontrées lors de la phase de test, ont été traitées.
Le cas particulier @p = {1, {3,4,5}[{0,1}[0]]} ne donnera pas @p = {1, 3} mais donnera {3}.
Les conflits décalage/réduction ont été évités.

Les améliorations éventuelles

1. Répondre aux instructions de type @p = {1, {3,4,5}[{0,1}[0]]}, il faudra rajouter une table de symbole temporaire pour les déclarations sans stockage et leur attribuer des clés afin de pouvoir les utiliser après dans l'instruction, ou une éventuelle pile.

2. **Division**
    - L'Ajout des règles de grammaire pour prendre en charge l'opération de division dans les expressions arithmétiques.
3. **Opérateurs de comparaison :**
    - Étendre la grammaire pour inclure des opérateurs de comparaison et des expressions conditionnelles.
4. **Instructions conditionnelles et boucles :**
    - L'intégration des règles de grammaire pour les instructions conditionnelles (if, else) et les boucles (for, while).
5. **Fonctions :**
    - L'ajout des règles pour définir et appeler des fonctions, y compris la gestion des paramètres et des valeurs de retour.
6. **Gestion des erreurs améliorée :**
    - L'amélioration des règles de gestion des erreurs pour couvrir un éventail plus large de scénarios d'erreurs potentiels.
7. **Optimisations :**
    - Amélioration et optimisation des initialisations et expressions imbriquées.

Comment exécuter le fichier

Structure du code:

/Projet
- **Dossier :** /calculatrice
  - **fichier :** Calculatrice.java
- **Dossier :** /compilationFiles
  - **Dossier :** calculatrice
    - **fichier :** Calculatrice.class
  - // Tous les fichiers générés par lex et cup.
- **Dossier :** /images
- **Dossier :** /tests
  - **fichier :** testDifferents.txt
  - **fichier :** testGeneral.txt
- **fichier :** calculatrice.cup
- **fichier :** calculatrice.lex
- **fichier :** makefile.sh
- **fichier :** makefileTerminal.sh
- java-cup-11b-runtime.jar

Dans la classe Calculatrice, on retrouve les tables de symboles ainsi que les méthodes:

- getSymboleTable()
- getSymboleTableTab()
- print()
- toStringFromSymbolVar(String key)
- toStringFromSymbolTab(String key)
- toStringTab(ArrayList<Integer> tableau)

Exécution par instruction:

Placez vous dans le projet. Sur votre terminal exécuter la commande:

```
./makefileTerminal.sh calculatrice
```

Ainsi, vous serez capable d'exécuter le programme instruction par instruction.
votre terminal devra afficher le message suivant:

```
Faites entrer vos instructions :
--------------------------------
```

Exécution des tests:

Si vous voulez exécuter directement les tests déjà proposé, placez vous dans le projet exécuter la commande suivante:

TestGenerale: contenant les instruction données dans l'énoncé

```
./makefile.sh calculatrice testGeneral.txt
```

Votre terminal devra afficher le message suivant:

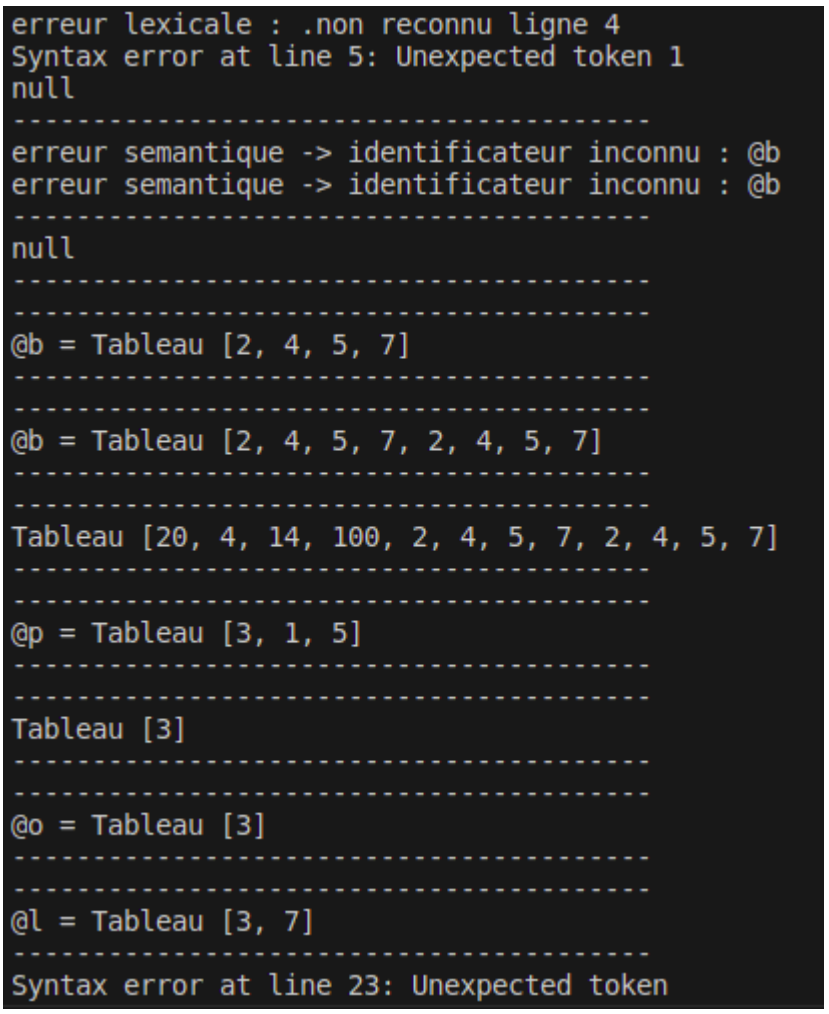
```
----------------------------------------
Entier 18
----------------------------------------
----------------------------------------
Tableau [3, 4, 5]
----------------------------------------
a = Entier 123
----------------------------------------
----------------------------------------
@b = Tableau [2, 4, 5, 7]
----------------------------------------
----------------------------------------
Entier 127
----------------------------------------
----------------------------------------
@montab = Tableau [127]
----------------------------------------
val2 = Entier 56
----------------------------------------
----------------------------------------
Tableau [20, 4, 14, 100, 2, 4, 5, 7]
----------------------------------------
Affichage :
----------------------------------------
les valeurs des variables sont :
a = Entier 123
val2 = Entier 56
@b = Tableau [2, 4, 5, 7]
@montab = Tableau [127]
----------------------------------------
erreur de type sur l'operateur +
----------------------------------------
Erreur ligne: 10
erreur semantique -> identificateur inconnu : d
----------------------------------------
----------------------------------------

Au revoir!
```

TestDifferents: contenant des instructions testé aléatoirement

```
./makefile.sh calculatrice testDifferents.txt
```

Votre terminal devra afficher le message suivant:



Conclusion

En conclusion, la création d'un compilateur de calculatrice fut un challenge intéressant, plusieurs opportunités pour étendre et améliorer les fonctionnalités du langage sont envisageables. Avec des fonctionnalités robustes et une gestion appropriée des erreurs, le compilateur peut devenir un outil puissant pour manipuler des expressions arithmétiques complexes et des opérations sur les tableaux.