Bilal LATRACH

Compilation 1

Projet de Compilation

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

Introduction

Le projet vise à créer une "calculatrice" capable de reconnaître et évaluer des expressions manipulant des entiers ainsi que des tableaux, en utilisant des opérateurs d'addition, de multiplication, et de concaténation. Cette calculatrice permet également la définition de tableaux, l'accès indicé à ces tableaux, ainsi que l'affectation de variables, qu'elles soient entières ou tableaux. L'interpréteur doit garantir la gestion des erreurs lexicales, syntaxiques, et sémantiques. Dans cette perspective, cet interpréteur doit non seulement identifier des symboles inappropriés mais aussi traiter des expressions incorrectes, des erreurs de typage, et des accès indicées hors limites.

Grammaire utilisée

calculatrice → liste_operations

liste_operations → liste_operations operation

| liste_operations concatination

| liste_operations affectation

| liste_operations PRINT

| liste_operations NL

| ε

operation → operation ADD operation

| operation MULT operation

| NB | ID

| PARENTHES_O operation PARENTHES_F

| tableau ACROCHET_O operation ACROCHET_F

| ARUBAS ID ACROCHET_O operation ACROCHET_F

affectation → ID EGALE operation | ARUBAS ID EGALE concatination

concatination → tableau CONCAT concatination | tableau

| PARENTHES_O concatination PARENTHES_F

tableau → TABLEAU | ACOLAD_O expression ACOLAD_F | ARUBAS ID

| ACOLAD_O ACOLAD_F

expression → expression VIRGULE operation | operation

Description Fonctionnelle

Fonctionnement du programme

Le programme permet de :

- Effectuer des opérations d'addition et de multiplication sur des entiers positifs en suivant l'ordre de priorité standard, avec une préférence pour les multiplications avant les additions. De plus, prendre en compte les opérations entre parenthèses pour définir clairement l'ordre d'évaluation des expressions.

Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
5+6*2+1
- : entier 18
14+6*2
- : entier 26
(5+4)*6
- : entier 54
(5+6)*2+1
- : entier 23
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Réaliser des concaténations impliquant au moins deux tableaux.

  Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
{1,2,3} ^ {4,5,6,7}
- : tableau [1, 2, 3, 4, 5, 6, 7]
{1,2} ^ {3,8,9} ^ {11,18,22,29}
- : tableau [1, 2, 3, 8, 9, 11, 18, 22, 29]
{}
- : tableau[]
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Réaliser l'accès indicé sur un tableau.

  Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
{5,8,9,16}[3]
- : entier 16
{1,8,6,2,7,6}[4]
- : entier 7
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Effectuer des affectations de valeurs à des variables ou des tableaux tout en respectant les types associés.

- Utiliser la commande PRINT pour afficher toutes les variables présentes dans l'environnement d'exécution.
- Accepter les lignes vides.

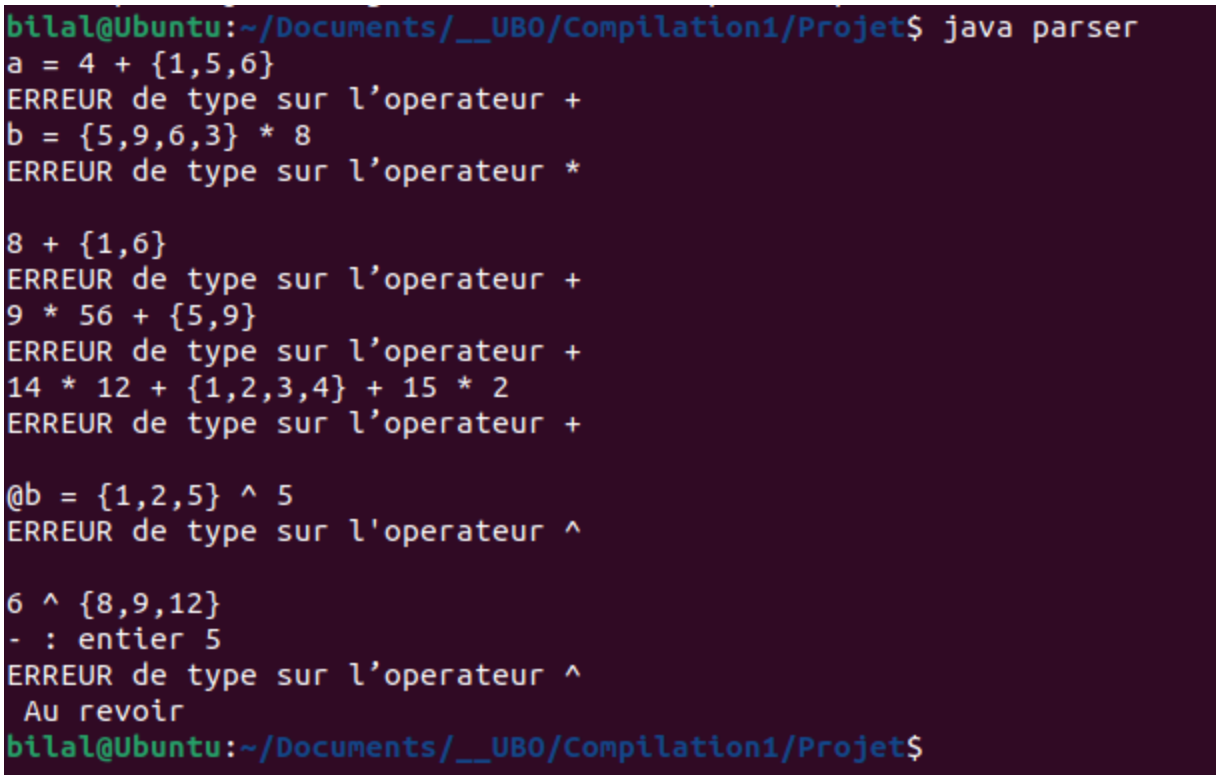
Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
a = 5
a = entier 5
b = 2 + a * 3
b = entier 17
c = b + a * 3
c = entier 32

@t = {1,2,3} ^ {4,5,6,7}
@t = tableau [1, 2, 3, 4, 5, 6, 7]

@g = @t ^ {9,10}
@g = tableau [1, 2, 3, 4, 5, 6, 7, 9, 10]

@f = @t ^ @g
@f = tableau [1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 9, 10]

@e = @t ^ {}
@e = tableau [1, 2, 3, 4, 5, 6, 7]

PRINT
Les valeurs des variables sont :
a = entier 5
b = entier 17
c = entier 32
@t = tableau [1, 2, 3, 4, 5, 6, 7]
@e = tableau [1, 2, 3, 4, 5, 6, 7]
@f = tableau [1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 9, 10]
@g = tableau [1, 2, 3, 4, 5, 6, 7, 9, 10]

 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

Traitement d'erreurs

- Le programme est capable de détecter les caractères inconnus et affiche un message d'erreur indiquant le caractère problématique ainsi que sa position.
- Le programme est doté d'une gestion des erreurs de type, que ce soit lors d'une tentative d'addition ou de multiplication entre un entier et un tableau, ou lors d'une concaténation entre un tableau et un entier. En cas de détection de telles erreurs, le programme affiche un message d'erreur approprié et continue le traitement des expressions à partir de la ligne suivante.

  Testes



- Le programme détecte les identifiants inconnus. Si un identifiant utilisé dans une expression n'a pas été préalablement déclaré ou affecté, le programme signale cette erreur, indiquant l'identifiant inconnu.

Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
a + 5
ERREUR semantique -> identificateur inconnu : a
5 + 8 * d
ERREUR semantique -> identificateur inconnu : d
@a ^ @b
ERREUR semantique -> identificateur inconnu : @a
ERREUR semantique -> identificateur inconnu : @b
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Le programme signale l'accès à un tableau avec des indices invalides. Si une tentative est faite pour accéder à un élément d'un tableau avec un indice en dehors de la plage valide.

  Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
@a = {1,2,3,4,5}
@a = tableau [1, 2, 3, 4, 5]
@a[6]
ERREUR Indice 6 INVALID dans le tableau : @a
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Le programme détecte les opérations d'addition ou de multiplication entre deux tableaux, ainsi que la concaténation entre deux entiers. En cas de tentative d'appliquer ces opérations entre des types incompatibles, le programme signale une erreur de type, indiquant clairement la nature de l'opération incorrecte.

Testes

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
5 ^ 4
ERREUR de type sur l'operateur ^

{1,2,3} + {2, 6}
- : entier 4
ERREUR de type sur l'operateur +
{1,2,3} * {3,5}
ERREUR de type sur l'operateur *
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- En cas de rencontre avec une erreur non spécifiée précédemment, le programme affiche un message d'erreur (ERREUR INCONNU) et continue le traitement des expressions suivantes.

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
2, 8
ERREUR INCONNU  ligne 1 colone 2
- : entier 8
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

Limitations éventuelles

- Le programme pourrait ne pas être optimisé pour les grands ensembles de données, ce qui pourrait entraîner des ralentissements ou des problèmes de gestion de la mémoire avec des entrées volumineuses.

Les erreurs de programme

- Le programme actuel ne prend pas en charge l'accès indicé sur le résultat d'une concaténation. Lorsqu'une tentative est faite, telle que (@b ^ {1,2,3}) [2], le programme génère une erreur et bloque le processus de parsing. Cette fonctionnalité devrait être prise en compte pour permettre l'accès indicé sur le résultat des opérations de concaténation.

Exemple

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
@b = {1,2,3}
@b = tableau [1, 2, 3]

(@b ^ {4,5,6})[3]
ERREUR INCONNU  ligne 3 colone 15

Couldn't repair and continue parse
Exception in thread "main" java.lang.Exception: Can't recover from previous error(s)
        at java_cup.runtime.lr_parser.report_fatal_error(lr_parser.java:392)
        at java_cup.runtime.lr_parser.unrecovered_syntax_error(lr_parser.java:539)
        at java_cup.runtime.lr_parser.parse(lr_parser.java:731)
        at parser.main(parser.java:241)
```

- Le programme réussit à identifier les caractères qui ne correspondent pas à la grammaire. Cependant, lorsqu'il détecte une erreur, par exemple, dans la ligne a = 5 / 2 le programme doit améliorer son comportement pour ignorer correctement la partie de la ligne avant et après l'erreur, soit " / ". Actuellement, il ne parvient pas à ignorer cette portion de la ligne après avoir détecté une erreur.

  Exemple

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
a = 5 / 2
ERREUR lexicale : caractere illegal : </> ligne 1 colonne 7
a = entier 5
- : entier 2
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

- Le programme parvient à détecter les erreurs de type lors de la concaténation de deux entiers. Toutefois, il ne parvient pas à ignorer le reste de la ligne après avoir identifié l'erreur, continuant à lire le reste de la ligne jusqu'à la fin.

Exemple

```
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$ java parser
5 ^ 4
ERREUR de type sur l'operateur ^
- : entier 4
 Au revoir
bilal@Ubuntu:~/Documents/__UBO/Compilation1/Projet$
```

Les améliorations possibles

- Intégrer la possibilité d'accéder à un indice spécifique dans le résultat d'une concaténation. par exemple (@a ^ @b) [3]
- Suite à la détection d'une erreur de concaténation entre deux entiers, le programme doit ignorer le reste de la ligne.
- Ignorer entièrement une ligne dès qu'un caractère inconnu est détecté.
- Optimiser le programme pour prendre en charge des opérations sur des entiers de grande taille.
- Permettez à l'utilisateur de spécifier les variables à afficher avec la commande PRINT. Par exemple, PRINT a, b affichera uniquement les variables 'a' et 'b'.