Projet compilation

Libert Robin
Zielinski Pierre
BA3 Info

14 mai 2018

Introduction

Description des lexèmes

INITIAL STATE

BLOC_BEGIN ← {{

TXT ← [\w|;|&|<|>|"|_|-|.|\|/|\n|\p|:|,| \t]+

IN_STRING_STATE

APO ← '

STRING ← [\w|;|&|<|>|"|_|-|.|\|/|\n|\p|:|,|]+

IN_CODE_STATE

APO ← '

BLOC_END ← }}

FOR ← for

IN ← in

DO ← do

ENDFOR ← endfor

IF ← if

ELSE ← else

ENDIF ← endif

PRINT ← print

INTEGER ← \d+

ADD_OP ← +|-

MUL_OP ← *|/

PAR_FERM ←)

PAR_OUVR ← (

DOT_COMMA ← ;

DOT ← .

ASSIGNEMENT ← :=

BOOLEAN ← true|false

VAR ← [\w]+

OPERATOR ← <|>|=|!=

BINOPERATOR ← or|and

Description de la grammaire

programme ← TXT | TXT programme

programme ← dumboBloc | dumboBloc programme

dumboBloc ← BLOC_BEGIN expressionList BLOC_END

expressionList ← expression DOT_COMMA | expression DOT_COMMA expressionList

expression ← variableN ASSIGNEMENT stringExpression | variableN ASSIGNEMENT list

expression ← PRINT stringExpression

expression ← FOR variableN IN list DO expressionList ENDFOR | FOR variableN IN variable DO expressionList ENDFOR

expression ← IF boolean DO expressionList ENDIF | IF boolean DO expressionList ELSE expressionList ENDIF

stringExpression ← boolean | string | stringExpression DOT stringExpression

list ← PAR_OUVR stringListInterior PAR_FERM | PAR_OUVR integerListInterior PAR_FERM

stringListInterior ← string | string COMMA stringListInterior

integerListInterior ← integer | integer COMMA integerListInterior

variable ← VAR

variableN ← VAR

string ← APO STRING APO

integer ← integerVar | variable | integer ADD_OP integer | integer MUL_OP integer

integerVar ← INTEGER

boolean ← booleanVar | booleanOP | boolean BINOPERATOR boolean

booleanOP ← integer OPERATOR integer | integer

booleanVar ← BOOLEAN

Gestion du if et du for

Pour chaque expression définie dans notre programme lors de l'analyse syntaxique, nous créons une classe correspondant à cette expression. Chaque classe aura une fonction d'évaluation que l'on appellera durant la phase d'analyse syntaxique.

Une boucle for est une expression : expression ← FOR variableN IN list DO expressionList ENDFOR | FOR variableN IN variable DO expressionList ENDFOR
Nous faisons la distinctions entre 2 types de variables. La première, variableN est un nom de variable dans lequel nous allons attribuer une valeur. La seconde, variable, est une variable déjà existante dans laquelle nous allons récupérer une valeur. Une boucle for est gérée par la classe ForExpression qui prend variableN en premier paramètre, list ou variable en second paramètre et une expressionList en dernier paramètre. A chaque itération, nous changeons la valeur à l'emplacement variableN dans un dictionnaire et l'envoyons à expressionList pour être évalué.

Notre IF est une expression : expression ← IF boolean DO expressionList ENDIF | IF boolean DO expressionList ELSE expressionList ENDIF
Un IF est géré par la classe IfExpression qui prend en premier paramètre un boolean en second paramètre une expressionList et en dernier paramètre, soit None soit une autre expressionList.

Difficultés rencontrées