Universidad
Carlos III de Madrid

Práctica Final

Analizador Lenguaje BSL (Parte 1 - Análisis Léxico + Gramática)

Procesadores del Lenguaje

Carlos Dumont Cabrilla – 100074907

Universidad Carlos III de Madrid. 2020

Table of Contents

1 Introducción

Se En la siguiente memoria se indica la gramática diseñada para el Analizador de Lenguaje BSL. Los tokens y símbolos asociados se encuentran definidos en el archivo lexer.jflex

2 Gramática

La gramática planteada sigue la especificación del lenguaje BSL.

Símbolos Terminales

- SEMI, PLUS, MINUS, TIMES, DIVIDEDBY, LPAREN, RPAREN
- AND, OR, NOT, EQUALTO, LEQUAL, GEQUAL
- INTTYPE, REALTYPE, BOOLEANTYPE, STRUCTTYPE, CHARTYPE
- TRUEVALUE, FALSEVALUE
- LBRACE, RBRACE, DOT, COMMA, SIMPLECOMMA, ASSIGNSYMBOL
- SI, ENTONCES, SINO, FINSI, MIENTRAS, FINMIENTRAS
- FUNCION, RETURN
- NUMBER, DOUBLENUMBER, HEXNUMBER, EXPONENTIAL, LOGARITHM
- COMMENT
- ID, CAPSID, ATTRIBUTEID

Símbolos No Terminales

- statement_list
- statement
- expr
- term
- factor
- assign
- declare
- simple_declare
- declare_list

- struct_define
- struct_declare
- logic_expr
- block
- if_block
- while_block
- struct_attribute
- attribute_chain
- function_definition
- function_call

Producciones

statement_list ::= statement_list statement
| statement

statement ::= expr SEMI
| COMMENT
| assign SEMI
| declare SEMI
| struct_define SEMI
| if_block
| while_block
| function_definition
| function_call SEMI

factor ::= NUMBER
| DOUBLENUMBER
| HEXNUMBER
| LPAREN expr RPAREN
| EXPONENTIAL expr RPAREN
| LOGARITHM expr RPAREN
| MINUS factor
| PLUS factor
| ID
| function_call

term ::= term TIMES factor
| term DIVIDEDBY factor
| factor

expr ::= expr PLUS term
| expr MINUS term
| term

simple_declare ::= INTTYPE ID
| REALTYPE ID
| BOOLEANTYPE ID
| CHARTYPE ID

struct_define ::= STRUCTTYPE CAPSID LBRACE declare_list RBRACE

struct_declare ::= CAPSID ID

declare_list ::= declare_list COMMA declare
| declare

declare ::= simple_declare
| struct_declare

assign ::= ID ASSIGNSYMBOL expr
| ID ASSIGNSYMBOL ID
| simple_declare ASSIGNSYMBOL expr
| ID ASSIGNSYMBOL logic_expr
| simple_declare ASSIGNSYMBOL logic_expr
| struct_attribute ASSIGNSYMBOL logic_expr

struct_attribute ::= ID attribute_chain

attribute_chain ::= ATTRIBUTEID attribute_chain
| ATTRIBUTEID

logic_expr ::= logic_expr AND logic_expr
| logic_expr OR logic_expr
| NOT logic_expr
| expr EQUALTO expr
| expr LEQUAL expr
| expr GEQUAL expr
| LPAREN logic_expr RPAREN

block ::= LBRACE statement_list RBRACE

if_block ::= SI logic_expr ENTONCES block FINSI
| SI logic_expr ENTONCES block SINO block FINSI

while_block ::= MIENTRAS logic_expr block FINMIENTRAS

function_definition ::= FUNCION ID LPAREN simple_declare RPAREN RETURN INTTYPE LBRACE block RBRACE

| FUNCION ID LPAREN simple_declare RPAREN RETURN REALTYPE LBRACE block RBRACE

| FUNCION ID LPAREN simple_declare RPAREN RETURN BOOLEANTYPE LBRACE block RBRACE

| FUNCION ID LPAREN simple_declare RPAREN RETURN CHARTYPE LBRACE block RBRACE

function_call ::= ID LPAREN expr RPAREN

| ID LPAREN logic_expr RPAREN

3 Conclusión

Esta primera parte de la práctica final ha servido para realizar el analizador léxico de un Lenguaje BSL y afianzar en general conceptos mas avanzados sobre las producciones de una gramática para un analizador sintáctico aplicado a sentencias mas complejas.

En primer lugar, ha servido para ampliar el **analizador léxico** hasta poder aceptar el **Lenguaje BSL**.

En segundo lugar, me ha permitido desarrollar una **gramática** bastante compleja para el **analizador sintáctico** que permita incluso generación y asignación de variables compuestas como los STRUCT, así como definición y llamada de funciones.