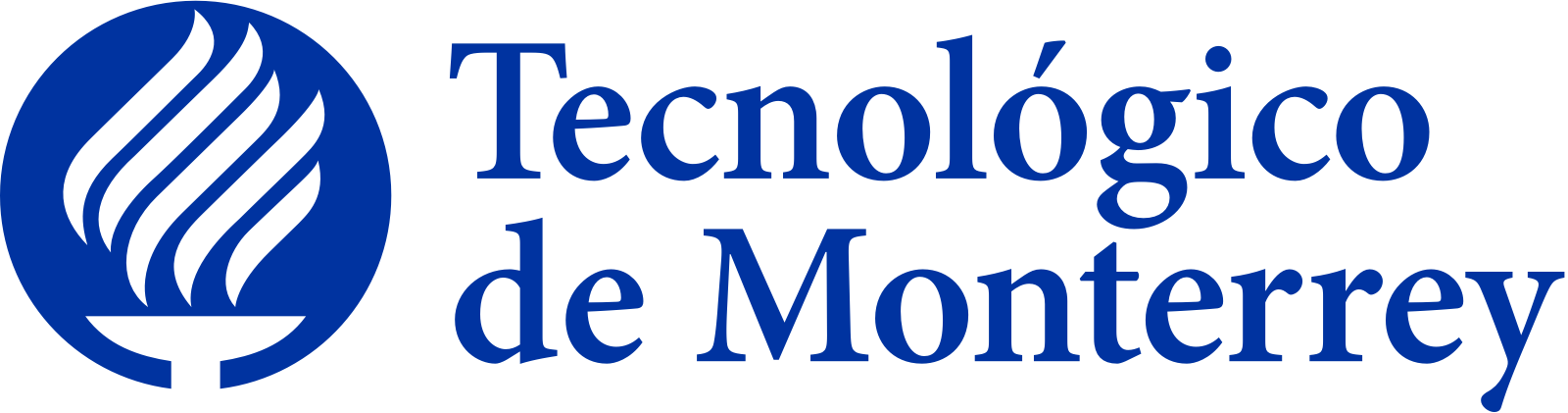
****

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

Campus Monterrey

**Entrega 1.- Análisis de Léxico y Sintaxis (Scanner y Parser)**

***“QUETZAL*”**

**Diseño de compiladores (Ago 19 Gpo 2)**

Ing. Héctor Gibrán Ceballo Cancino

Ing. Elda Guadalupe Quiroga González



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lizzie Marielle Guajardo Mozo  A00818258 | |  | Alejandro González Valles  A00818616 | |

Monterrey, N.L. México 7 de octubre del 2019

En esta entrega daremos la propuesta de nuestro compilador Quetzal,la cual contiene los diagramas de sintaxis y léxico. Adicionalmente, se adjunta el código que engloba el lexer y parser diseñados en Antlr. Se incluye una carpet que contiene un archivo “runlang.py” el cual si se ejecuta analizará el archivo de texto especificado en el código.

También se incluyen tres ejemplos en archivos de txt (EjemploC<1,2,3>)

, todos son válidos y cubren todas las posibilidades gramaticales que se diseñaron.

La entrega está completa y es funcional pero está sujeta a cambios según avancemos en el proyecto.

El lexer y parser no arrojará ningún mensaje si el código dado está correcto, pero sí indicará de forma detallada qué error encuentra (ejemplo: indica si falta un carácter y dice cúal es, indica si esperaba algún valor e indica todas las posibilidades que puede recibir).

Gramática del Lenguaje en ANTLR

grammar Quetzal;

/\* Parser Rules \*/

program: variables? function? main;

main: TK\_FUNC TK\_MAIN SYM\_PAREN\_OPEN SYM\_PAREN\_CLOSE SYM\_CURLY\_BRACK\_OPEN variables? statute\* SYM\_CURLY\_BRACK\_CLOSE;

variables: (TK\_DEFINE types TYPE\_ID (SYM\_ASSIGN expression)? (SYM\_COMMA TYPE\_ID (SYM\_ASSIGN expression)?)\* SYM\_SEMI\_COL)+;

function: TK\_FUNC (types | TK\_VOID) TYPE\_ID SYM\_PAREN\_OPEN (types TYPE\_ID (SYM\_COMMA types TYPE\_ID)\*)? SYM\_PAREN\_CLOSE SYM\_CURLY\_BRACK\_OPEN variables? statute\* SYM\_CURLY\_BRACK\_CLOSE;

block: SYM\_CURLY\_BRACK\_OPEN statute SYM\_CURLY\_BRACK\_CLOSE;

types: (TK\_INT | TK\_FLOAT | TK\_COLOR | TK\_BOOL) (SYM\_SQUARE\_BRACK\_OPEN expression SYM\_SQUARE\_BRACK\_CLOSE)\*;

constants:(TYPE\_INT | TYPE\_FLOAT | CTE\_TAG | TYPE\_BOOL | TYPE\_COLOR);

statute: returning|condition|prints|read|callfunc|specfunct|loop|assignation;

assignation: TYPE\_ID (SYM\_SQUARE\_BRACK\_OPEN expression SYM\_SQUARE\_BRACK\_CLOSE)\* SYM\_ASSIGN (specfunct|(expression SYM\_SEMI\_COL));

condition: TK\_IF SYM\_PAREN\_OPEN expression SYM\_PAREN\_CLOSE block (TK\_ELSE block)?;

prints: TK\_PRINT SYM\_PAREN\_OPEN expression SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

var\_cte: constants

| TYPE\_ID (SYM\_SQUARE\_BRACK\_OPEN expression SYM\_SQUARE\_BRACK\_CLOSE)\*

| TYPE\_ID SYM\_PAREN\_OPEN expression (SYM\_COMMA expression)\* SYM\_PAREN\_CLOSE;

expression: exp (logic\_op exp)? ((SYM\_OR | SYM\_AND) exp (logic\_op exp)?)\*;

exp: term ((SYM\_PLUS | SYM\_MINUS) term)\*;

term: factor (( SYM\_MULT | SYM\_DIV ) factor)\*;

factor: (SYM\_PAREN\_OPEN expression SYM\_PAREN\_CLOSE ) | ( (SYM\_PLUS|SYM\_MINUS)? var\_cte );

logic\_op: SYM\_EQUAL | SYM\_GRE\_THAN | SYM\_LOW\_THAN | SYM\_NOT\_EQUAL | SYM\_GRE\_EQ | SYM\_LOW\_EQ;

read: TK\_READ SYM\_PAREN\_OPEN TYPE\_ID (SYM\_SQUARE\_BRACK\_OPEN expression SYM\_SQUARE\_BRACK\_CLOSE(SYM\_SQUARE\_BRACK\_OPEN expression SYM\_SQUARE\_BRACK\_CLOSE)?)? SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

returning: TK\_RETURN expression? SYM\_SEMI\_COL;

callfunc: TYPE\_ID SYM\_PAREN\_OPEN (expression (SYM\_COMMA expression)\*)? SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

loop: TK\_WHILE SYM\_PAREN\_OPEN expression SYM\_PAREN\_CLOSE block;

//SPECIAL FUNCTIONS

openimg: TK\_OPENIMG SYM\_PAREN\_OPEN CTE\_TAG SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

saveimg: TK\_SAVEIMG SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA CTE\_TAG SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

color\_replace: TK\_COLOR\_REPLACE SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA (TYPE\_COLOR | TYPE\_ID) SYM\_COMMA (TYPE\_COLOR | TYPE\_ID) SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

grayscale: TK\_GRAYSCALE SYM\_PAREN\_OPEN TYPE\_ID SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

color\_filter: TK\_COLOR\_REPLACE SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA (TYPE\_COLOR | TYPE\_ID) SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

edgeDetection: TK\_EDGE\_DETECTION SYM\_PAREN\_OPEN TYPE\_ID SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

scaleImg: TK\_SCALE\_IMAGE SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA (SYM\_PLUS|SYM\_MINUS)?(TYPE\_FLOAT | TYPE\_ID) SYM\_COMMA (SYM\_PLUS|SYM\_MINUS)?(TYPE\_FLOAT | TYPE\_ID) SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

getColorPalette: TK\_GET\_COLOR\_PALETTE SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA (TYPE\_FLOAT | TYPE\_ID) SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

colorMatchImage: TK\_COLOR\_MATCH\_IMAGE SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA TYPE\_ID SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

encodeSteganography: TK\_ENCODE\_STEGANOGRAPHY SYM\_PAREN\_OPEN TYPE\_ID SYM\_COMMA CTE\_TAG SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

decodeSteganography: TK\_DECODE\_STEGANOGRAPHY SYM\_PAREN\_OPEN TYPE\_ID SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

getImageHeight: TK\_GET\_IMAGE\_HEIGHT SYM\_PAREN\_OPEN CTE\_TAG SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

getImageWidth: TK\_GET\_IMAGE\_WIDTH SYM\_PAREN\_OPEN CTE\_TAG SYM\_PAREN\_CLOSE SYM\_SEMI\_COL;

specfunct:openimg | saveimg | grayscale | color\_replace | color\_filter | edgeDetection | scaleImg | getColorPalette | colorMatchImage | encodeSteganography | decodeSteganography | getImageHeight | getImageWidth;

/\* Lexer Rules \*/

/\* TOKENS \*/

TK\_PROGRAM: 'program';

TK\_FUNC: 'func';

TK\_DEFINE: 'define';

TK\_RETURN: 'return';

TK\_IF: 'if';

TK\_ELSE: 'else';

TK\_WHILE: 'loop';

TK\_PRINT: 'prints';

TK\_READ: 'read';

TK\_TRUE: 'true';

TK\_FALSE: 'false';

TK\_MAIN: 'main';

TK\_VOID: 'void';

//Special Functions

TK\_OPENIMG: 'openImg';

TK\_SAVEIMG: 'saveImg';

TK\_GRAYSCALE: 'grayscale';

TK\_COLOR\_REPLACE: 'color\_replace';

TK\_COLOR\_FILTER: 'color\_filter';

TK\_EDGE\_DETECTION: 'edgeDetection';

TK\_SCALE\_IMAGE: 'scaleImg';

TK\_GET\_COLOR\_PALETTE: 'getColorPalette';

TK\_COLOR\_MATCH\_IMAGE: 'colorMatchImage';

TK\_ENCODE\_STEGANOGRAPHY: 'encodeSteganography';

TK\_DECODE\_STEGANOGRAPHY: 'decodeSteganography';

TK\_GET\_IMAGE\_HEIGHT: 'getImageHeight';

TK\_GET\_IMAGE\_WIDTH: 'getImageWidth';

//Types Tokens

TK\_BOOL: 'bool';

TK\_INT: 'int';

TK\_FLOAT: 'float';

TK\_COLOR: 'color';

//Reserved Color Names

CTE\_COLOR: TK\_WHITE | TK\_RED | TK\_LIME | TK\_BLUE | TK\_SILVER | TK\_MAROON | TK\_GREEN | TK\_NAVY | TK\_GRAY | TK\_YELLOW | TK\_AQUA | TK\_FUCHSIA | TK\_BLACK | TK\_OLIVE | TK\_TEAL | TK\_PURPLE;

TK\_WHITE: 'White'; //FFFFFF

TK\_RED: 'Red'; //FF0000

TK\_LIME: 'Lime'; //00FF00

TK\_BLUE: 'Blue'; //0000FF

TK\_SILVER: 'Silver'; //C0C0C0

TK\_MAROON: 'Maroon'; //800000

TK\_GREEN: 'Green'; //008000

TK\_NAVY: 'Navy'; //000080

TK\_GRAY: 'Gray'; //808080

TK\_YELLOW: 'Yellow'; //FFFF00

TK\_AQUA: 'Aqua'; //00FFFF

TK\_FUCHSIA: 'Fuchsia'; //FF00FF

TK\_BLACK: 'Black'; //000000

TK\_OLIVE: 'Olive'; //808000

TK\_TEAL: 'Teal'; //008080

TK\_PURPLE: 'Purple'; //800080

/\* SYMBOL \*/

SYM\_SEMI\_COL : ';';

SYM\_DOUB\_COL : ':';

SYM\_COMMA : ',';

SYM\_UNDER\_SCORE: '\_';

SYM\_CURLY\_BRACK\_OPEN: '{';

SYM\_CURLY\_BRACK\_CLOSE: '}';

SYM\_SQUARE\_BRACK\_OPEN: '[';

SYM\_SQUARE\_BRACK\_CLOSE: ']';

SYM\_PAREN\_OPEN: '(';

SYM\_PAREN\_CLOSE: ')';

SYM\_ASSIGN : '=';

SYM\_EQUAL : '==';

SYM\_GRE\_THAN : '>';

SYM\_LOW\_THAN : '<';

SYM\_NOT\_EQUAL : '!=';

SYM\_GRE\_EQ: '>=';

SYM\_LOW\_EQ: '<=';

SYM\_OR: '||';

SYM\_AND: '&&';

SYM\_QUOT : '"';

SYM\_PLUS: '+';

SYM\_MINUS: '-';

SYM\_MULT: '\*';

SYM\_DIV: '/';

/\* TYPES \*/

TYPE\_FLOAT : FRAG\_DIGIT+ ('.' FRAG\_DIGIT+)?;

TYPE\_INT : FRAG\_DIGIT+;

TYPE\_ID : FRAG\_LOWER\_CASE FRAG\_FOLLOW\*;

TYPE\_COLOR : CTE\_COLOR | FRAG\_HEX\_COLOR;

CTE\_TAG: SYM\_QUOT FRAG\_FOLLOW\* SYM\_QUOT;

TYPE\_BOOL: TK\_TRUE | TK\_FALSE;

/\* FRAGMENTS \*/

fragment FRAG\_LETTER: FRAG\_UPPER\_CASE | FRAG\_LOWER\_CASE;

fragment FRAG\_FOLLOW: FRAG\_LETTER|FRAG\_DIGIT|SYM\_UNDER\_SCORE|'/'|':'|'.'|'\\';

fragment FRAG\_DIGIT : [0-9];

fragment FRAG\_UPPER\_CASE: [A-Z];

fragment FRAG\_LOWER\_CASE: [a-z];

fragment FRAG\_HEX\_VAL: ('A'|'B'|'C'|'D'|'E'|'F') | ('a'|'b'|'c'|'d'|'e'|'f') | FRAG\_DIGIT;

fragment FRAG\_HEX\_COLOR: '#' FRAG\_HEX\_VAL FRAG\_HEX\_VAL FRAG\_HEX\_VAL FRAG\_HEX\_VAL FRAG\_HEX\_VAL FRAG\_HEX\_VAL;

/\* NEW LINES AND WHITESPACE

Telling the parser to skip new lines and white spaces (works in windows and linux)\*/

NEWLINE : '\r'? '\n' -> skip;

WS: (' ' | '\t')+ -> skip;

/\* COMMENTS \*/

BLOCK\_COMMENT : SYM\_DIV SYM\_MULT .\*? SYM\_MULT SYM\_DIV -> channel(HIDDEN); /\* ALLOW FOR BLOCK COMMENTS WITH THE FORMAT OF '/\* this is a block comment \*/

LINE\_COMMENT : SYM\_DIV SYM\_DIV ~[\r\n]\* -> channel(HIDDEN); /\* ALLOW FOR LINE COMMENTS WITH THE FORMAT OF //this is a line comment \*/