Manual Técnico:

Analizador Sintáctico LL

Ambiente de desarrollo:

Entorno de desarrollo: Visual Studio Code 1.55.2
Sistema operativo; Windows 10 x64
Versión del SO: 20H2
Memoria Ram: 12GB
CPU: intel core !7-7700HQ
Versión de Node.js: 14.16.0
Version de Angular cli: 11.2.4

Gramática

Gramática para el analizador Lexico:

Saltos de espacios de cualquier tipo:
\s+

Comentario de linea:
"#"[^\n]*

Comentario de bloque:
/**"([^*]+|*+[^/*])*"*"+"/"

Expresión regular que es igual a : a-zA-Z, para cadenas de letras
"aA-zZ"

Expresión regular que es igual a: 0-9, para números enteros
"0-9"

Nombre de los Terminales
$_"("_"+|[a-zA-Z]+|[0-9]+)

Nombre de los No Terminales
%_"("_"+|[a-zA-Z]+|[0-9]+)

Números enteros
[0-9]+

Cadenas de texto con solo las letras del abecedario
[a-zA-Z]+

Cualquier tipo de comillas
("""|"`"|"‘"|"’")

Palabras Reservadas:

Terminal
Wison
Lex
Syntax
No_Terminal
Initial_Sim

Símbolos:
*
=
-
<
{
}

+
%
(
)
[
]
;
?
¿
:
|

Gramática para el analizador Sintáctico:

Simbolos no terminales:
Inicio, inicio_sig, contenido, lex. cont_lex, only_terminal,expresion_lex,cont_expresion_lex,symb_especiales, cont_expr_regulares, cont_expr_regulares_combinado, expr_reg_comb, cont_expr_reg_comb, param_cont_expr_regulares, sig_cont_expr_regulares,clausula_expr,syn,cont_syn,no_terminales,no_terminal,initial_prod,producciones,produccion, Derivaciones,derivacion

Símbolos terminales:
TERMINAL,WISON,LEX,SYNTAX,NO_TERMINAL,INITIAL_SYM,EXP_REG_ABC,EXP_REG_NUM, NOMBRE_TERMINAL, NOMBRE_PRODUCCION, ENTERO, STRING,*,=,-,<,{,},+,%,(,),[,], ; , ? , ¿ : ,COMILLA, | , EOF , INVALID

Gramática:

```
inicio
        :    WISON inicio_sig EOF
        ;

inicio_sig
        : '¿'    contenido '?' WISON
        |error
        ;

contenido
        : lex syn
        ;

lex
        : LEX '{' ':' cont_lex ':' '}'
        |error
        ;

cont_lex
        : cont_lex only_terminal
        |only_terminal
        ;
```

```
only_terminal
        : TERMINAL NOMBRE_TERMINAL '<' '-' expresion_lex ';'
        |error
        ;

expresion_lex
        : COMILLA cont_expresion_lex COMILLA
        | cont_expr_regulares
        |error
        ;

cont_expresion_lex
        : STRING
        | INVALID
        | symb_especiales
        ;

symb_especiales
        : '*'
        | '='
        | '-'
        | '<'
        | '{'
        | '}'
```

```
        | '+'
        | '%'
        | '('
        | ')'
        | '['
        | ']'
        | ';'
        | '?'
        | '¿'
        | ':'
        | COMILLA
        | '|'
        ;

cont_expr_regulares
        : '[' param_cont_expr_regulares sig_cont_expr_regulares
        | cont_expr_regulares_combinado
        ;

cont_expr_regulares_combinado
        : cont_expr_regulares_combinado expr_reg_comb
        | expr_reg_comb
        ;
```

```
expr_reg_comb
        : '(' cont_expr_reg_comb ')'
        ;

cont_expr_reg_comb
        : cont_expr_regulares
        | NOMBRE_TERMINAL
        |error
        ;

param_cont_expr_regulares
        : EXP_REG_ABC
        | EXP_REG_NUM
        |error
        ;

sig_cont_expr_regulares
        : ']'
        | ']' clausula_expr
        |error
        ;

clausula_expr
        : '*'
        | '+'
```

```
        | '?'
        ;
syn
        : SYNTAX '{' '{' ':' cont_syn ':' '}' '}'
        |error
        ;

cont_syn
        :    no_terminales initial_prod producciones
        ;

no_terminales
        : no_terminales no_terminal
        | no_terminal
        |error
        ;

no_terminal
        : NO_TERMINAL NOMBRE_PRODUCCION ';'
        ;

initial_prod
        : INITIAL_SYM NOMBRE_PRODUCCION ';'
        |error
        ;
```

```
producciones
	: producciones produccion
	| produccion
	;
produccion
	: NOMBRE_PRODUCCION '<' '=' derivaciones ';'
	|error
	;

derivaciones
	: derivaciones derivacion
	|derivacion
	|error
	;

derivacion
		: NOMBRE_PRODUCCION
		| NOMBRE_TERMINAL
		| '|'
		;
```