**PRÁCTICA Procesadores Del Lenguaje**

**2022-2023**

**GRUPO 13**

Integrantes: Mario López Estaire, Andrés Bravo Francos, Grettell Umpierrez Sardiñas



* Sentencias: Sentencia repetitiva (switch-case)
* Técnicas de Análisis Sintáctico: Descendente con tablas
* Operadores especiales: Asignación con multiplicación(\*=)
* Comentarios: Comentario de bloque (/\*\*/)
* Cadenas: Con comillas dobles (“ ”)
* Comentarios → /\*...\*/
* Constantes →
  + Enteras → d⁺ → 16b(1 palabra) → máx (32767)

- Cadena …. → “c+, / (/n) … Ascii(0)”

* + Lógicas → pal\_reservada(true, false)
* Operadores →
  + Relación → ==
  + Aritméticos → \*
  + Lógico → &&
  + Asignación → =
  + Asignación con multiplicación → \*=
* Identificadores → l⁺ ⊕ d/\_
* Declaraciones → pal\_reservada(let)
* Datos→
  + Entero → pal\_reservada (int)
  + Lógico→ “ (boolean)
  + Cadena → (String)
* Entrada/Salida → pal\_reservada (print,input).
* Sentencias → =, pal\_reservada(return, function, if, for)
* Otros → (,),{,},;

# ANALIZADOR LÉXICO

**TOKENS**

<abrirParentesis,->

<cerrarParentesis,->

<abrirCorchete,->

<cerrarCorchete,->

<coma,->

<ptoComa,->

<opAritmetico,1>

<opAritmetico,2>

<opRelacional,1>

<opLogico,1>

<asigMultiplicacion,->

<asignación,->

<switch,->

<case,->

<default,->

<break,->

<cte\_entera,valor>

<cadena,lexema>

<identificador, posTs>

<let,->

<int,->

<boolean,->

<string,->

<if,->

<else,->

<function,->

<input,->

<if,->

<return, - >

<eof,->

Op\_aritmetico 1: \*

2. +

Op\_relacional 1: ==

Op\_logico 1: &&

# GRAMÁTICA

S → delS | lA | dB | =C | “D | \*E | &F | /G | ( | ) | { | } | ; | , +

A → lA | dA | \_A | o.c

B → dB | o.c

C → = | o.c

D → c3D | o.c

E → = | o.c

F → &

H → c1H | \*I

I → c2H | \*I | /S

LEYENDA:

L : {a-z,A\_Z}

d : {0\_9}

c1 : todos los caracteres – {\*}

c2 : todos los caracteres – {\*,/}

c3 : todos los caracteres – {“}

del: tab, espacio, eol , …

# AUTÓMATA

# Imagen que contiene Forma Descripción generada automáticamente

# ACCIONES SEMANTICAS

# S→S: leer();

# S→A: lexema = c; leer();

# A→A: lexema = lexema + c; leer();

# A→B: if(lexema == palabraReservada) then generarToken(lexema, - );

# else{

# Pos=BuscarLugarTS(lexema);

# if(pos ¡= null) then generarToken(lexema, pos);

# else{

# pos=insertarIdTS(lexema);

# generarToken(lexema,pos);

# }

# S→C: valor = char\_int(d); leer();

# C→C: valor = valor \* 10 + char\_int(d); leer();

# C→D: if (valor > XXXXXX ) tren error;

# else generar\_token(cteEntera, valor);

# S→E: leer();

# E→F: leer();

# F→F: leer();

# F→G: leer();

# G→F: leer();

# G→G: leer();

# G→S: leer();

# S→K: lexema = “ ”; leer();

# 

# K→K: lexema = lexema + c; leer();

# K→L: if(lexema.lenght > XXXXX ) then error;

# else generar\_Token(cadena, lexema);

# 

# S→H: leer();

# H→I: generar\_Token(asignación, -);

# H→J: generar\_Token(opRelacional, 1);

# S→M: leer();

# M→N: generar\_Token(asigMultiplicacion, - );

# M→O: generar\_Token(opAritmetico, 1 );

# S→P: leer();

# P→Q: generar\_Token(opLogico, 1 );

# S→R: generar\_Token(abrirParentesis, - );

# S→T: generar\_Token(cerrarParentesis, - );

# S→U: generar\_Token(abrirCorchete, - );

# S→V: generar\_Token(cerrarCorchete, - );

# S→W: generar\_Token(ptoComa, - );

# S→X: generar\_Token(coma, - );

# S→Y: generar\_Token(opAritmetico, 2 );

# MATRIZ AFD

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Estado/carácter | l | d | \_ | C1 | C2 | C3 | ¨ | / | ; | , | \* | + | & | ( | ) | { | } | o.c. | = | del |
| S | A-2 | C-5 | C-6 |  |  |  | K-9 | E-8 | W-24 | X-25 | M-15 | Y-26 | P-18 | R-20 | T-21 | U-22 | V-23 |  | H-12 | S-21 |
| A | A-3 | A-3 | A-3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | B-4 |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | D-7 |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  | F-8 |  |  |  |  |  |  |  |  |  |
| F |  |  |  | F-8 |  |  |  |  |  |  | G-8 |  |  |  |  |  |  |  |  |  |
| G |  |  |  |  | F-8 |  |  | S-8 |  |  | G-8 |  |  |  |  |  |  |  |  |  |
| H |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | J-14 | I-13 |  |
| I |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| J |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| K |  |  |  |  |  | K-10 | L-11 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| L |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | O-17 | N-16 |  |
| N |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| O |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P |  |  |  |  |  |  |  |  |  |  |  |  | Q-19 |  |  |  |  |  |  |  |
| Q |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| T |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| U |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| X |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Y |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# \*Los estados encerrados EN ROJO representan estados finales (y por tanto no reciben nada).

# \*Las casillas no rellenas representan ERRORES.