**PRÁCTICA Procesadores Del Lenguaje**

**2022-2023**

**GRUPO 13**

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



* Sentencias: Sentencia de Selección Múltiple (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\* + ASCII 0 fin de cadena “ 🡪 c (cualquier carácter excepto “) 🡪 máx. 64 caracteres

* + Lógicas → pal\_reservada(true, false)
* Operadores →
  + Relación → ==
  + Aritméticos → \*
  + Lógico → &&
  + Asignación → =
  + Asignación con multiplicación → \*=
* Identificadores → (l + \_) + (d + \_ + l) \* 🡪 l (a-z, A-Z) 🡪 d(0-9)
* Declaraciones → pal\_reservada(let)
* Datos→
  + Entero → pal\_reservada (int)
  + Lógico→ pal\_reservada(boolean)
  + Cadena → pal\_reservada (string)
* Entrada/Salida → pal\_reservada (print, input).
* Sentencias → pal\_reservada(return, function, if, switch, case, default, break)
* Otros → ( ) { } ; : ,

# ANALIZADOR LÉXICO

**TOKENS**

*Generales*

<abrirParentesis, - >

<cerrarParentesis, - >

<abrirCorchete, - >

<cerrarCorchete, - >

<coma,->

<ptoComa, - >

<dosPuntos,->

<eof, - >

*Operadores*

<opAritmetico,1 >

<opAritmetico, 2 >

<opRelacional,1 >

<opLogico,1 >

<asigMultiplicacion, - >

<asignacion, - >

*Valores de Datos*

<cte\_entera, valor>

<cadena, lexema>

Declaraciones

<Identificador, posTs>

<let, - >

Tipos de Datos *(Palabras reservadas)*

<int, - >

<boolean, - >

<string, - >

*Funciones (Palabras reservadas)*

<function, - >

<return, - >

*Sentencias (Palabras reservadas)*

<if, - >

<switch, - >

<case, - >

<default, - >

<break, - >

*E/S (Palabras reservadas)*

<input, - >

<print, - >

Op\_aritmético 1: \*

2: +

Op\_relacional 1: ==

Op\_lógico 1: &&

# GRAMÁTICA

| :

LEYENDA:

: {a-z,A\_Z}

: {0 - 9}

: cualquier carácter no contemplado en el nodo

: todos los caracteres – {\*}

: todos los caracteres – {\*,/}

: todos los caracteres – {“}

: tab, espacio, eol

# AUTÓMATA

# 

# ACCIONES SEMÁNTICAS

# 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 ) then error;

# else generarToken(cte\_entera, 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.length > XXXXX ) then error;

# else generarToken(cadena, lexema);

# S→H: leer();

# H→I: generarToken(opRelacional, 1);

# H→J: generarToken(asignación, -);

# S→M: leer();

# M→N: generarToken(asigMultiplicacion, - );

# M→O: generarToken(opAritmetico, 1 );

# S→P: leer();

# P→Q: generarToken(opLogico, 1 );

# S→R: generarToken(abrirParentesis, - );

# S→T: generarToken(cerrarParentesis, - );

# S→U: generarToken(abrirCorchete, - );

# S→V: generarToken(cerrarCorchete, - );

# S→W: generarToken(ptoComa, - );

# S→X: generarToken(coma, - );

# S→Y: generarToken(dosPuntos, - );

# S→Z: generarToken(opAritmetico, 2 );

# [MATRIZ AFD](MATRIZ%20AFD.docx)