# Memoria de la práctica

Creación de un lexer Grupo 9

Autoría: Carlos Santiago Miguel Gutiérrez Henny Sánchez

Asignatura: Procesadores de Lenguajes

Fecha: October 12, 2022

# **Contents**

1	Especialidad del grupo	1
2	Tokens	1
3	Gramática y Autómata 3.1 Gramática	
4	Pasos semánticos	3
5	Diseño de la tabla de símbolos	
6	Anexo	4

# 1 Especialidad del grupo

Nuestro equipo contiene las siguientes especialidades:

- Sentencias de repeticion for
- Asignación con suma +=
- Comentarios en línea //
- Cadenas de comillas simples ' '

## 2 Tokens

Los tokens seleccionados son los siguientes:

Table 1

Token	Genera
$\overline{\langle ParI, - \rangle}$	paréntesis (
$\langle ParF, - \rangle$	paréntesis )
$\langle CorI, - \rangle$	llave {
$\langle CorF, - \rangle$	llave }
$\langle PyC, - \rangle$	punto y coma;
$\langle Com, - \rangle$	coma ,
$\langle Dif, - \rangle$	not lógico!
$\langle Eq, - \rangle$	asignación =
$\langle Eqq, - \rangle$	equals lógico =
$\langle Mas, - \rangle$	suma aritmética +
$\langle MasI, - \rangle$	asignación com suma +=
$\langle Id, posTS \rangle$	identificador
$\langle for, - \rangle$	palabra reservada for
$\langle let, - \rangle$	palabra reservada let
$\langle func, - \rangle$	palabra reservada function
$\langle int, valor \rangle$	número entero
$\langle cad, lexema \rangle$	cadena de caractéres
$\langle bool, valor \rangle$	true o false
$\langle print, - \rangle$	palabra reservada print
$\langle ret, - \rangle$	palabra reservada return
$\langle assign, - \rangle$	
$\langle if, - \rangle$	palabra reservada if
$\langle else, - \rangle$	palabra reservada else
$\langle while, - \rangle$	palabra reservada while
$\langle do, - \rangle$	palabra reservada do

# 3 Gramática y Autómata

## 3.1 Gramática

La gramática generada es:

 $S := delS | 'A | dB | lD | _D | , | ; | ( | ) | { | } | +C | =C | /E$ 

A := o.c1A

 $B := dB \mid lambda$ 

C := | lambda

D := |ID| LD | dD | lambda

E := /F

 $F := o.c2 \mid CR$ 

d: dígitos (0-9)l: letras (a-z)del: delimitadores

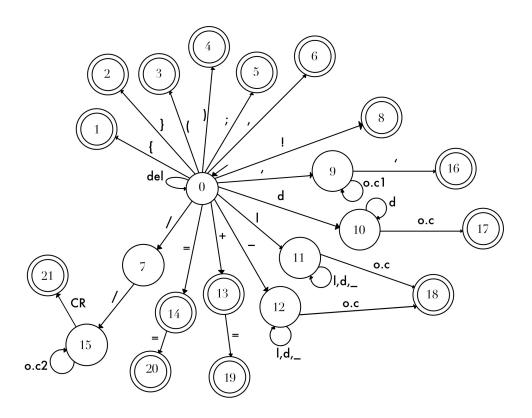
o.c: cualquier otro caracter

o.c1: cualquier otro caracter menos '

o.c2: cualquier otro caracter menos salto de línea

#### 3.2 Autómata

No es el definitivo.



## 4 Pasos semánticos

```
A: Lee;
B: Lee; GenToken(ParI, -)
C: Lee; GenToken(ParF, -)
D: Lee; GenToken(CorI, -)
E: Lee; GenToken(CorF, -)
F: Lee; GenToken(PyC, -)
G: Lee; GenToken(Com, -)
H: Lee; GenToken(Dif, -)
I: Lee; GenToken(Eq, -)
J: Lee; GenToken (Eqq, -)
K: Lee; GenToken(Mas, -)
L: Lee; GenToken(MasI, -)
M: Reescribir Lee; GenToken(id, posTS)
N: Lee; valor:= valorHexadecimal (h)
O: Lee; valor:= valor * 16 + valorHexadecimal (h)
P: Lee; lexema:= car
Q: Lee; lexema:= lexema + car
```

```
R: Lee; if valor< 2^31 then GenToken(int, valor); else error()
S: para GenToken(cadena,lexema)</pre>
```

#### 5 Diseño de la tabla de símbolos

### 6 Anexo

Aquí hay que poner el resultado de nuestra implementación, tanto buena como mala (6 en total, 3 buenas y 3 con errores)

# References