



**SEP**  
SECRETARÍA DE  
EDUCACIÓN PÚBLICA



## **Lenguajes y Autómatas I**

### **Unidad 4. Análisis Léxico**

**Dueñas Tellechea Glenn**

**Mtra. Millan Castro Ana Luisa**

**11 de junio de 2021**

## Índice

Características principales del lenguaje de su caso de estudio .....	3
Tabla de símbolos .....	7
Tabla de errores léxicos .....	7
Autómata finito .....	8
Matriz de transición .....	9
Corrida del analizador léxico .....	10

## Características principales del lenguaje de su caso de estudio

Las características principales que maneja este lenguaje son principalmente que maneja diferentes tipos de datos como lo son enteros, cadenas, operadores lógicos e identificadores. Destaca que se aceptan la mayoría de los símbolos utilizados en otros lenguajes de programación para en un futuro darles funciones como lo son los paréntesis y las palabras reservadas, las cuales sirven para definir variables y estructuras lógicas que se implementaran en un futuro.

Una parte interesante del lenguaje es que no se utilizan los símbolos de pesos, porcentajes o el et, los cuales en otros lenguajes se utilizan para definir variables o funciones, o incluso como relacional, como lo sería el caso de los dobles et en Java.

En este caso todas las palabras reservadas se escriben en letras minúsculas, ya que es similar al escribir de manera normal y puede ser mas amigable para el usuario.

Las características principales del analizador léxico es la función de RandomAccessFile la cual permite importar y leer archivos de texto para poder realizar el análisis a estos. También los chequeos isLetter, e isCharacter propios de Java son útiles para agilizar el proceso de chequeo. Otra característica principal que se podría considerar el corazón de este programa es la matriz de transición la cual contiene toda la información acerca de como debe de reaccionar el programa a los caracteres.

Otro componente vital son los nodos que se generan ya que estos nos permiten ir guardando los resultados de nuestro análisis léxico para poder desplegarlos posteriormente al usuario.

## BNF

### 1. Algortimo

<algoritmo> ::=

Algoritmo <nombre-algoritmo>() es <sección-declaraciones>

    inicio

        <acción>

    fin

<nombre-algoritmo> ::= <letra> { <caracter-nombre> }

<sección-declaraciones> ::= { <declaración-objeto> }

#### 1.1 Declaración de objetos

<declaración-objeto> ::= <decl-obj-simple>

## 1.2 Definiciones básicas

<letra> ::= A..Z | a..z

<dígito> ::= 0..9

<símbolo> ::= \_ | % | & | : | , | ; | = | ( | etc...

<caracter-nombre> ::= <letra> | <dígito> | \_

<caracter> ::= <letra> | <dígito> | <símbolo>

## 1.3 Declaración de objetos simples

<decl-obj-simple> ::= <decl-variable>

<decl-variable> ::= <nombre-variable> : <nombre-tipo-simple>

<nombre-tipo-simple> ::= Entero | Decimal | Caracter | Cadena | Lógico

## 1.4 Nombres y uso de objetos

<nombre-objeto> ::= <nombre-variable>

<nombre-variable> ::= <letra> { <caracter-nombre> }

## 1.5 Valores y literales

<valor> ::= <valor-numérico> | <valor-caracter> | <valor-cadena> | <valor-lógico>

<valor-numérico> ::= [+ | -] <dígito> {<dígito>} [ . {<dígito>} ]

<valor-caracter> ::= 0 <caracter> 0

<valor-cadena> ::= 0 {<caracter>}

0

<valor-lógico> ::= True | False

## 1.6 Expresiones

$\langle \text{expresion-logica} \rangle ::= \langle \text{expresion-relacional} \rangle \langle \text{expresion-logica1} \rangle$

$\quad | \langle \text{operador-logico1} \rangle \langle \text{expresion-logica} \rangle \langle \text{expresion-logica1} \rangle$

$\quad | (\langle \text{expresion-logica} \rangle) \langle \text{expresion-logica1} \rangle$

$\quad | \langle \text{id} \rangle \langle \text{expresion-logica1} \rangle$

$\quad | \langle \text{valor-logico} \rangle \langle \text{expresion-logica1} \rangle$

$\langle \text{expresion-logica1} \rangle ::= \langle \text{operador-logico2} \rangle \langle \text{expresion-logica} \rangle \langle \text{expresion-logica} \rangle \mid \epsilon$

$\langle \text{expresion-relacional} \rangle ::= \langle \text{expresion-numerica} \rangle \langle \text{operador-relacional} \rangle \langle \text{expresion-numerica} \rangle$

$\langle \text{expresion-numerica} \rangle ::= (\langle \text{expresion-numerica} \rangle) \langle \text{expresion-numerica1} \rangle$

$\quad | - \langle \text{expresion-numerica} \rangle \langle \text{expresion-numerica1} \rangle$

$\quad | \langle \text{id} \rangle \langle \text{expresion-numerica1} \rangle$

$\quad | \langle \text{valor-numerico} \rangle \langle \text{expresion-numerica1} \rangle$

$\quad | \langle \text{valor-logico} \rangle \langle \text{expresion-numerica1} \rangle$

$\quad | \langle \text{valor-cadena} \rangle \langle \text{expresion-numerica} \rangle$

$\langle \text{expresion-numerica1} \rangle ::= \langle \text{operador-numerico} \rangle \langle \text{expresion-numerica} \rangle \langle \text{expresion-numerica1} \rangle \mid \epsilon$

$\langle \text{operador-logico1} \rangle ::= \text{not}$

$\langle \text{operador-logico2} \rangle ::= \text{and} \mid \text{or}$

$\langle \text{operador-numerico} \rangle ::= + \mid - \mid * \mid /$

$\langle \text{operador-relacional} \rangle ::= < \mid > \mid = \mid <= \mid >= \mid = \mid <>$

$\langle \text{valor-logico} \rangle ::= \text{true} \mid \text{false}$

$\langle \text{valor-numerico} \rangle ::= D^+ (.D^+)?$

$\langle \text{valor-cadena} \rangle ::= \text{" ASCII* "}$

## 1.7 Acciones

$\langle \text{acción} \rangle ::= \langle \text{acción-elemental} \rangle \mid \langle \text{composición-secuencial-acciones} \rangle$

$\quad | \langle \text{esquema-condicional} \rangle \mid \langle \text{esquema-repetitivo} \rangle$

$\langle \text{acción-elemental} \rangle ::= \langle \text{asignación-interna} \rangle \mid \langle \text{asignación-externa} \rangle$   
 $\mid \langle \text{escritura} \rangle$   
 $\langle \text{asignación-interna} \rangle ::= \langle \text{nombre-variable} \rangle := \langle \text{valor} \rangle$   
 $\langle \text{asignación-externa} \rangle ::= \text{Leer } \langle \text{nombre-objeto} \rangle \{ , \langle \text{nombre-objeto} \rangle$   
 $\}$   
 $\langle \text{escritura} \rangle ::= \text{Escribir } \langle \text{nombre-objeto} \rangle \{ , \langle \text{nombre-objeto} \rangle \}$

## 1.8 Esquema composición secuencial

$\langle \text{composición-secuencial-acciones} \rangle ::= \langle \text{acción} \rangle \{ ; \langle \text{acción} \rangle \} [ ; ]$   
 $\langle \text{esquema-condicional} \rangle ::= \langle \text{esquema-cond-2} \rangle$   
 $\langle \text{esquema-cond-2} \rangle ::=$   
SI  $\langle \text{exp-lógica} \rangle$  ENTONCES  
 $\langle \text{acción} \rangle$   
[SINO  
 $\langle \text{acción} \rangle$ ]  
FIN-SI

## 1.9 Esquema repetitivo

$\langle \text{esquema-repetitivo} \rangle ::= \langle \text{esquema-mientras} \rangle$   
 $\langle \text{esquema-mientras} \rangle ::=$   
MIENTRAS  $\langle \text{exp-lógica} \rangle$  HACER  
 $\langle \text{acción} \rangle$   
FIN-MIENTRAS

## 2.0 Documentación

$\langle \text{comentario} \rangle ::= \text{'\{ } \langle \text{caracter} \rangle \text{'}}$

## Tabla de símbolos

### Tabla de símbolos

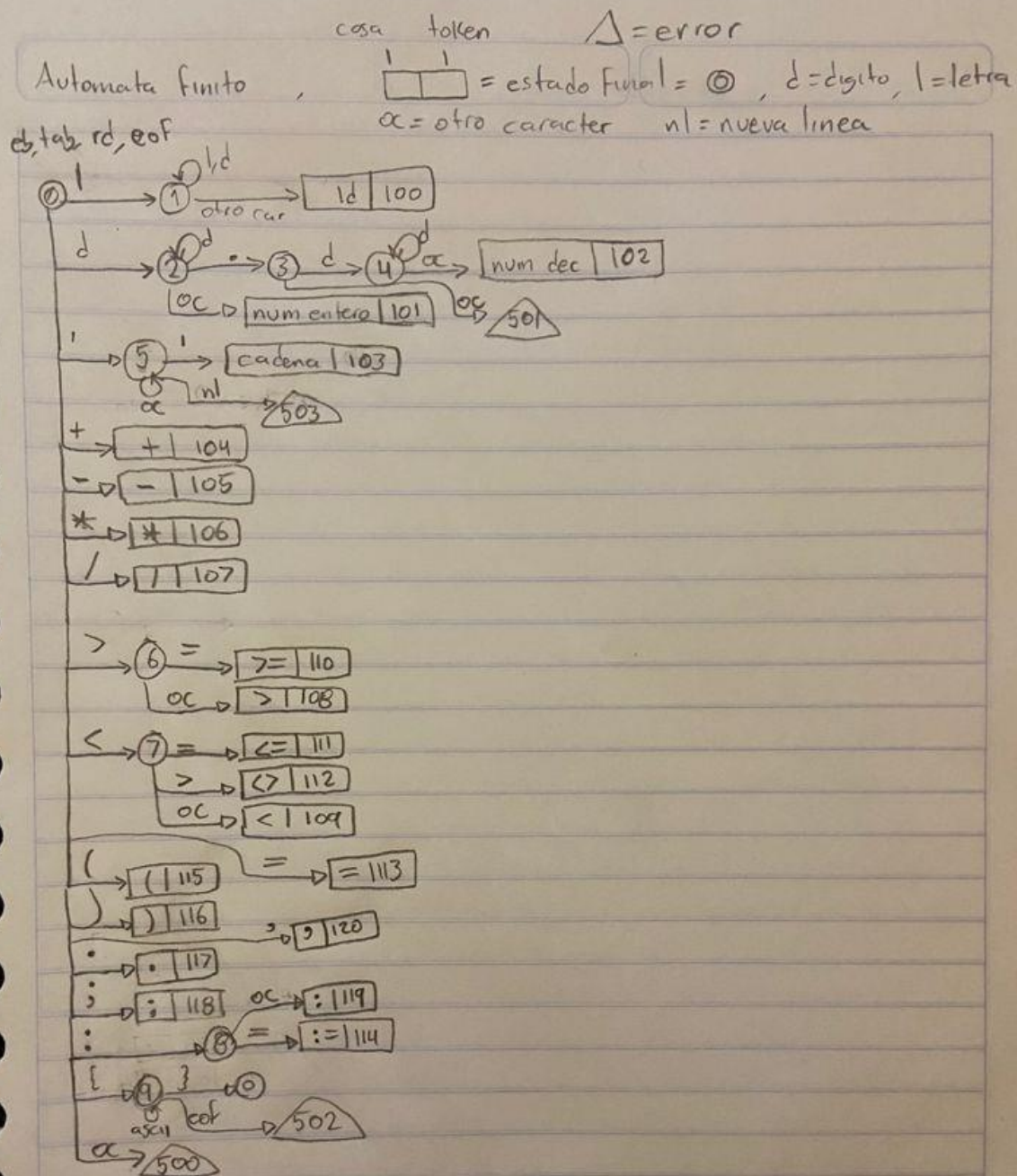
- identificadores 100 ::= | (l|d)\*
- numero (enteros 101 y decimales 102) ::= D<sup>+</sup> (.d<sup>+</sup>)
- cadenas 103 ::= "Caracteres\_ascii"
- caracteres ::= |...|...| etc. son 255 caracteres no generan token
- operadores aritméticos ::= |+ 104 | - 105 | \*106 | / 107
- operadores relacionales ::= |>108|<109|>=110|<=111|<> 112 | = 113
- operador de asignación ::= | := 114 |
- operadores lógicos ::= |and|or|not
- símbolos de agrupación ::= | (115 | )116
- símbolos de puntuación ::= | . 117 | ; 118 | : 119 | , 120
- palabras reservadas ::= | and 200|or 201|not 202 | verdadero 203 | falso 204 | leer 205| escribir 206 | si 207| entonces 208 | fin\_si 209 | sino 210|mientras 211|fin\_mientras 212| hacer 213 | algoritmo 214| es 215| inicio 216 | fin 217 | entero 218 | decimal 219 | cadena 220 | lógico 221
- comentarios ::= | {carácter\_ascii} no generan token
- delimitadores ::= | espacio en blanco | tab | fin de línea | no generan no generan token

## Tabla de errores léxicos

### Errores

- 500 ::= carácter no valido
- 501 ::= número no valido o mal formado
- 502 ::= comentario no cerrado
- 503 ::= cadena no cerrada

## Autómata finito



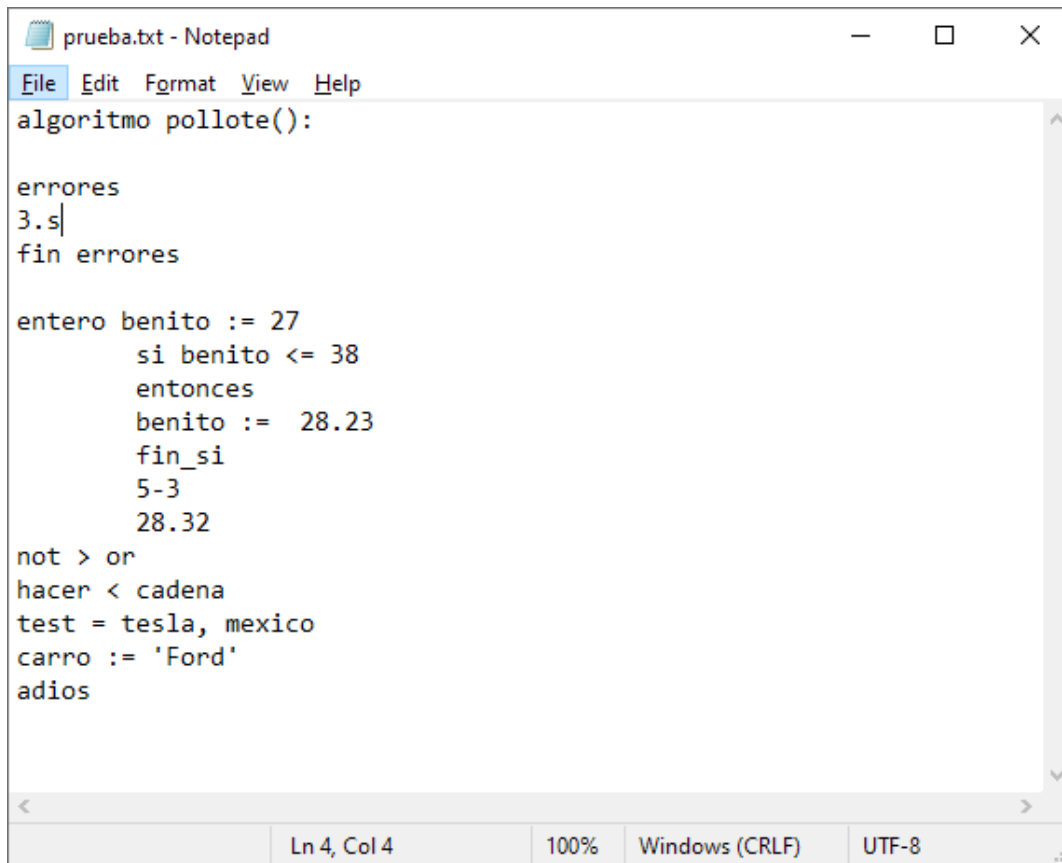


## Matriz de transición

	L	D	_	.	'	+	-	*	/	>	<	=	(	)	,	;	:	{	}	E b	T a b	N l	E o f	O c
0	1	2	5 0 0	1 1 7	5	1 0 4	1 0 5	1 0 6	1 0 7	6	7	1 1 3	1 1 5	1 1 6	1 2 0	1 1 8	8	9	5 0 0	0	0	0	0	5 0 0
1	1	1	1	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0	1	1	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0	1	1	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0	1
2	1 0 2	2	1 0 2	3	1 0 2	1 0 2	1 0 2	1 0 2	1 0 2	1	1	1 0 2	1 0 2	1 0 2	1 0 2	1 0 2	1	1	1 0 2	1 0 2	1 0 2	1 0 2	1 0 2	1
3	5 0 1	4	5 0 1	5 0 1	5 0 1	5 0 1	5 0 1	5 0 1	5 0 1	5	5	5 0 1	5 0 1	5 0 1	5 0 1	5 0 1	5	5	5 0 1	5 0 1	5 0 1	5 0 1	5 0 1	5
4	1 0 3	4	1 0 3	1 0 3	1 0 3	1 0 3	1 0 3	1 0 3	1 0 3	1	1	1 0 3	1 0 3	1 0 3	1 0 3	1 0 3	1	1	1 0 3	1 0 3	1 0 3	1 0 3	1 0 3	1
5	5	5	5	5	1 0 4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5 0 3	5 0 3	5
6	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1	1	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1	1	1 0 8	1 0 8	1 0 8	1 0 8	1 0 8	1
7	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1	1	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1	1	1 0 9	1 0 9	1 0 9	1 0 9	1 0 9	1
8	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1	1	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1	1	1 1 9	1 1 9	1 1 9	1 1 9	1 1 9	1
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	0	9	9	9	5 0 2	9

## Corrida del analizador léxico

### Error



```
prueba.txt - Notepad
File Edit Format View Help
algoritmo pollote():

errores
3.s
fin errores

entero benito := 27
    si benito <= 38
    entonces
    benito := 28.23
    fin_si
5-3
28.32

not > or
hacer < cadena
test = tesla, mexico
carro := 'Ford'
adios

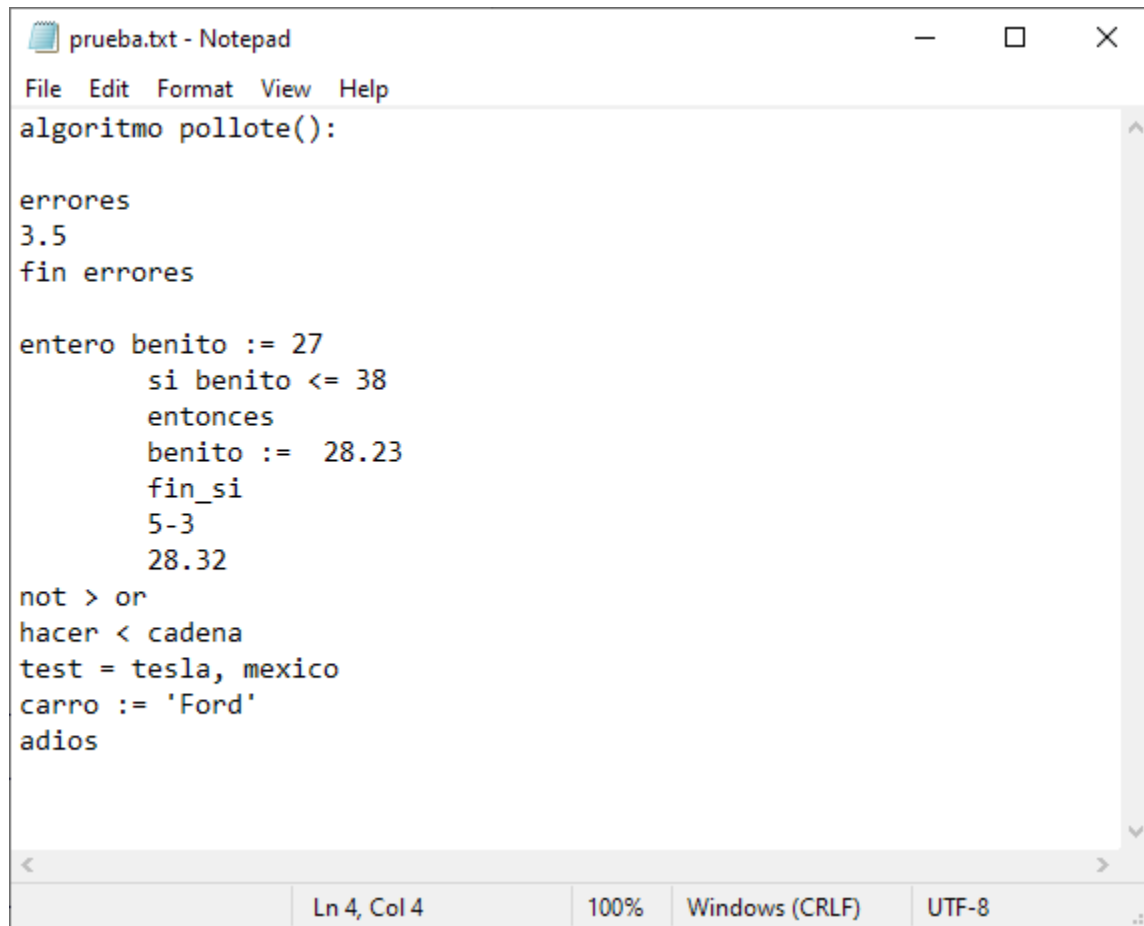
Ln 4, Col 4 100% Windows (CRLF) UTF-8
```

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Compilador ---
El error encontrado fue número no valido o mal formado error 501 caracter 115 en el renglon 4
algoritmo 214 1
pollote 100 1
( 115 1
) 116 1
: 119 1
errores 100 3

-----
BUILD SUCCESS
-----

Total time: 1.313 s
Finished at: 2021-06-10T18:24:34-07:00
Final Memory: 8M/37M
-----
```

## Corrección error



```
prueba.txt - Notepad
File Edit Format View Help
algoritmo pollote():

errores
3.5
fin errores

entero benito := 27
    si benito <= 38
    entonces
    benito := 28.23
    fin_si
    5-3
    28.32
not > or
hacer < cadena
test = tesla, mexico
carro := 'Ford'
adios

Ln 4, Col 4 100% Windows (CRLF) UTF-8
```

```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Compilador ---
```

```
algoritmo 214 1
pollote 100 1
( 115 1
) 116 1
: 119 1
errores 100 3
3.5 103 4
fin 217 5
errores 100 5
entero 218 7
benito 100 7
:= 114 7
27 102 7
si 207 8
benito 100 8
<= 111 8
38 102 8
entonces 208 9
benito 100 10
:= 114 10
28.23 103 10
fin_si 209 11
5 102 12
- 105 12
3 102 12
28.32 103 13
not 202 14
> 108 14
or 201 14
hacer 213 15
< 109 15
cadena 220 15
test 100 16
= 113 16
tesla 100 16
, 120 16
mexico 100 16
carro 100 17
:= 114 17
'Ford' 104 17
adios 100 18
```

```
Analisis Lexico Terminado
```

```
-----
BUILD SUCCESS
-----
```

```
Total time: 1.187 s
Finished at: 2021-06-10T18:26:46-07:00
Final Memory: 8M/37M
-----
```

```
!
```

## Código

```
1.  /*
2.   * To change this license header, choose License Headers in Project Properties.
3.   * To change this template file, choose Tools | Templates
4.   * and open the template in the editor.
5.   */
6.  package Main;
7.
8.  import java.io.RandomAccessFile;
9.
10. /**
11.  *
12.  * @author glenn
13.  */
14. class lexico {
15.
16.     nodo cabeza = null, p;
17.     int estado = 0, columna, valorMT, numRenglon = 1;
18.     int caracter = 0;
19.     String lexema = "";
20.     boolean errorEncontrado = false;
21.
22.     String archivo = "C:\\Users\\glenn\\Documents\\Escuela\\Semestre 6\\Lenguajes y
    automatas\\Compi\\Compilador\\roberto.txt";
23.
24.     int matriz[][] = { //
ponerle 13 v
25.         //      L      D      _      .      '      +      -      *      /      >      <      =      (      )      ,      ;
:      {      }      Eb      Tab      Nl      Eof      Oc
26.         //      0      1      2      3      4      5      6      7      8      9      10      11      12      13      14      15
16      17      18      19      20      21      22      23
27.         /*0*/ {1, 2, 500, 117, 5, 104, 105, 106, 107, 6, 7, 113, 115, 116, 120,
118, 8, 9, 500, 0, 0, 0, 500},
28.         /*1*/ {1, 1, 1, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100},
29.         /*2*/ {102, 2, 102, 3, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102, 102,
102, 102, 102, 102},
30.         /*3*/ {501, 4, 501, 501, 501, 501, 501, 501, 501, 501, 501, 501, 501, 501,
501, 501, 501, 501},
31.         /*4*/ {103, 4, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103,
103, 103, 103, 103},
32.         /*5*/ {5, 5, 5, 5, 104, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5},
33.         /*6*/ {108, 108, 108, 108, 108, 108, 108, 108, 108, 108, 108, 110, 108, 108, 108,
108, 108, 108, 108},
34.         /*7*/ {109, 109, 109, 109, 109, 109, 109, 109, 109, 109, 112, 109, 111, 109, 109,
109, 109, 109, 109},
35.         /*8*/ {119, 119, 119, 119, 119, 119, 119, 119, 119, 119, 119, 114, 119, 119, 119,
119, 119, 119, 119},
36.         /*9*/ {9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9},
37.     };
38.
39.     String palReservadas[][] = {
40.         {"and", "200"},
41.         {"or", "201"},
42.         {"not", "202"},
43.         {"verdadero", "203"},
44.         {"falso", "204"},
45.         {"leer", "205"},
46.         {"escribir", "206"},
47.         {"si", "207"},

```

```

48.         {"entonces", "208"},
49.         {"fin_si", "209"},
50.         {"sino", "210"},
51.         {"mientras", "211"},
52.         {"fin_mientras", "212"},
53.         {"hacer", "213"},
54.         {"algoritmo", "214"},
55.         {"es", "215"},
56.         {"inicio", "216"},
57.         {"fin", "217"},
58.         {"entero", "218"},
59.         {"decimal", "219"},
60.         {"cadena", "220"},
61.         {"lógico", "221"}
62.     };
63.
64.     String Errores[][] = {
65.         {"carácter no valido", "500"},
66.         {"número no valido o mal formado", "501"},
67.         {"comentario no cerrado", "502"},
68.         {"cadena no cerrada", "503"},};
69.
70.     RandomAccessFile file = null;
71.
72.     public lexico() {
73.         try {
74.             file = new RandomAccessFile(archivo, "r");
75.             while (caracter != -1) {
76.                 caracter = file.read();
77.
78.                 if (Character.isLetter(((char) caracter))) {
79.                     columna = 0;
80.                 } else if (Character.isDigit(((char) caracter))) {
81.                     columna = 1;
82.                 } else {
83.                     switch ((char) caracter) {
84.                         case '_':
85.                             columna = 2;
86.                             break;
87.                         case '.':
88.                             columna = 3;
89.                             break;
90.                         case '3':
91.                             columna = 4;
92.                             break;
93.                         case '+':
94.                             columna = 5;
95.                             break;
96.                         case '-':
97.                             columna = 6;
98.                             break;
99.                         case '*':
100.                            columna = 7;
101.                            break;
102.                         case '/':
103.                            columna = 8;
104.                            break;
105.                         case '>':
106.                            columna = 9;
107.                            break;
108.                         case '<':
109.                            columna = 10;
110.                            break;
111.                         case '=':
112.                            columna = 11;

```

```

113.         break;
114.     case '(':
115.         columna = 12;
116.         break;
117.     case ')':
118.         columna = 13;
119.         break;
120.     case ',':
121.         columna = 14;
122.         break;
123.     case ';':
124.         columna = 15;
125.         break;
126.     case ':':
127.         columna = 16;
128.         break;
129.     case '{':
130.         columna = 17;
131.         break;
132.     case '}':
133.         columna = 18;
134.         break;
135.     case 32:
136.         columna = 19;
137.         break;
138.     case 9:
139.         columna = 20;
140.         break;
141.     case 10: {
142.         columna = 21;
143.         numRenglon = numRenglon + 1;
144.     }
145.     case 13:
146.         columna = 22;
147.         break;
148.     default:
149.         columna = 23;
150.     }
151. }
152.
153. valorMT = matriz[estado][columna];
154.
155. if (valorMT < 100) {
156.     estado = valorMT;
157.
158.     if (estado == 0) {
159.         lexema = "";
160.     } else {
161.         lexema = lexema + (char) caracter;
162.     }
163.
164. } else if (valorMT >= 100 && valorMT < 500) {
165.     if (valorMT == 100) {
166.         validarPalabraReservada();
167.     }
168.     if (valorMT == 100 || valorMT == 101 || valorMT == 102 || valorMT ==
103 || valorMT == 108 || valorMT == 109 || valorMT == 119 || valorMT >= 200) {
169.         file.seek(file.getFilePointer() - 1);
170.     } else {
171.         lexema = lexema + (char) caracter;
172.     }
173.
174.     insertarNodo();
175.     estado = 0;
176.     lexema = "";

```

```

177.         } else {
178.             imprimirError();
179.             break;
180.         }
181.     }
182.     imprimirNodos();
183. } catch (Exception e) {
184.     System.out.println(e.getMessage());
185. } finally {
186.     try {
187.         if (file != null) {
188.             file.close();
189.         }
190.     } catch (Exception e) {
191.         System.out.println(e.getMessage());
192.     }
193. }
194. }
195.
196. private void imprimirError() {
197.     if (caracter != -1 && valorMT >= 500) {
198.         for (String[] errores : Errores) {
199.             if (valorMT == Integer.valueOf(errores[1])) {
200.                 System.out.println("El error encontrado fue " + errores[0] + " error "
+ valorMT + " caracter " + caracter + " en el renglon " + numRenglon);
201.             }
202.         }
203.         errorEncontrado = true;
204.     }
205. }
206.
207. private void validarPalabraReservada() {
208.     for (String[] palReservada : palReservadas) {
209.         if (lexema.equals(palReservada[0])) {
210.             valorMT = Integer.valueOf(palReservada[1]);
211.         }
212.     }
213. }
214.
215. private void insertarNodo() {
216.     nodo nodo = new nodo(lexema, valorMT, numRenglon);
217.
218.     if (cabeza == null) {
219.         cabeza = nodo;
220.         p = cabeza;
221.     } else {
222.         p.sig = nodo;
223.         p = nodo;
224.     }
225. }
226.
227. private void imprimirNodos() {
228.     p = cabeza;
229.     while (p != null) {
230.         System.out.println(p.lexema + " " + p.token + " " + p.renglon);
231.         p = p.sig;
232.     }
233. }
234. }
235.

```