

Nombre de la práctica	ANALIZADOR SINTACTICO (UNIDAD 5)			No.	5
Asignatura:	LENGUAJES Y AUTÓMATAS I	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES-3501	Duración de la práctica (Hrs)	10 horas

NOMBRE DEL ALUMNO: Jesus Navarrete Martínez

GRUPO: 3501

## I. Competencia(s) específica(s):

Construye un analizador sintáctico a partir de un lenguaje de programación.

Encuadre con CACEI: Registra el (los) atributo(s) de egreso y los criterios de desempeño que se evaluarán en la materia.

No. atributo	Atributos de egreso del PE que impactan en la asignatura	No. Criterio	Criterios de desempeño	No. Indicador	Indicadores
2	El estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de Ingeniería en Sistemas Computacionales	CD1	Identifica metodologías y procesos empleados en la resolución de problemas	I1	Identificación y reconocimiento de distintas metodologías para la resolución de problemas
		CD2	Diseña soluciones a problemas, empleando metodologías apropiadas al área	I1	Uso de metodologías para el modelado de la solución de sistemas y aplicaciones
				I2	Diseño algorítmico (Representación de diagramas de transiciones)
3	El estudiante plantea soluciones basadas en tecnologías empleando su juicio ingenieril para valorar necesidades, recursos y resultados esperados.	CD1	Emplea los conocimientos adquiridos para el desarrollar soluciones	I1	Elección de metodologías, técnicas y/o herramientas para el desarrollo de soluciones
				I2	Uso de metodologías adecuadas para el desarrollo de proyectos
				I3	Generación de productos y/o proyectos
		CD2	Analiza y comprueba resultados	I1	Realizar pruebas a los productos obtenidos
				I2	Documentar información de las pruebas realizadas y los resultados

## II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

Laboratorio de cómputo y equipo de cómputo personal.

## III. Material empleado:

- Equipo de cómputo
- Software para desarrollo NetBeans

## IV. Desarrollo de la práctica:

## ANALIZADOR LEXICO

### DESCRIPCIÓN DEL PROBLEMA:

Construcción de un Analizador Léxico

Un analizador léxico, también conocido como lexer o escáner, es una de las primeras y más fundamentales etapas en el proceso de compilación de un lenguaje de programación. Este componente tiene como principal objetivo transformar el código fuente, que generalmente consiste en una secuencia de caracteres, en una secuencia estructurada de tokens. Los tokens representan las unidades básicas y esenciales del lenguaje, tales como palabras reservadas, identificadores, operadores matemáticos y lógicos, delimitadores como paréntesis o llaves, y literales como números o cadenas de texto.

El proceso de análisis léxico no solo consiste en identificar y categorizar estas unidades, sino también en ignorar elementos que no contribuyen directamente a la semántica del lenguaje, como espacios en blanco y comentarios. Esto permite que las etapas posteriores del compilador trabajen con una representación más abstracta y manejable del código fuente.

La construcción de un analizador léxico enfrenta varios desafíos clave. Uno de ellos es la definición clara y precisa de las reglas léxicas que describen los tokens válidos del lenguaje. Estas reglas suelen expresarse mediante expresiones regulares, una herramienta poderosa para capturar patrones en cadenas de texto. Además, es crucial implementar un mecanismo eficiente para procesar el código fuente, ya que un lexer mal diseñado puede convertirse en un cuello de botella en el proceso de compilación.

Para simplificar y automatizar la implementación de analizadores léxicos, se utilizan herramientas como JFlex. JFlex es una librería que permite a los desarrolladores especificar las reglas léxicas de su lenguaje de programación utilizando una sintaxis basada en expresiones regulares. A partir de estas reglas, JFlex genera código Java que implementa el analizador léxico. Esto no solo reduce el tiempo y el esfuerzo de desarrollo, sino que también garantiza que el lexer sea eficiente y robusto.

En el contexto de este proyecto, se busca diseñar un analizador léxico para un lenguaje de programación definido específicamente, con una sintaxis y semántica particulares. Este analizador tendrá la responsabilidad de leer el código fuente proporcionado por el usuario y producir una secuencia de tokens que puedan ser utilizados por las etapas posteriores del compilador, como el analizador sintáctico y el analizador semántico. Estas etapas adicionales utilizarán los tokens para verificar la estructura y el significado del programa, asegurando que sea válido según las reglas del lenguaje y que cumpla con las expectativas del desarrollador. La implementación de un analizador léxico es un paso crucial en el diseño de un compilador. Su correcta ejecución no solo facilita el análisis posterior del código, sino que también permite detectar errores tempranos en el proceso de compilación, mejorando la calidad del software final y la experiencia del usuario.

## EXPLICACIÓN DEL CONTENIDO DE LA TABLA DE TOKENS

### ¿Qué es?

La tabla de tokens es una estructura fundamental en el desarrollo de un analizador léxico. Consiste en un listado de elementos léxicos (tokens), que representan los componentes básicos de un lenguaje de programación, junto con sus respectivos lexemas, que son las representaciones textuales de estos elementos en el código fuente.

### ¿Para qué sirve?

La tabla de tokens tiene múltiples propósitos dentro del proyecto de un compilador o intérprete, entre ellos:

- **Definir los componentes léxicos del lenguaje:** Agrupa operadores, palabras reservadas, símbolos, tipos de datos, y otros elementos que el analizador léxico debe reconocer.
- **Facilitar el reconocimiento del código fuente:** Permite transformar las cadenas de caracteres del programa en una secuencia de tokens, simplificando su procesamiento.
- **Estandarizar el análisis:** Cada token está asociado a una categoría semántica, ayudando a las etapas posteriores del compilador (sintáctica y semántica) a trabajar con datos consistentes y uniformes.

### Importancia dentro del proyecto:

En el contexto del desarrollo de un analizador léxico, la tabla de tokens es esencial por las siguientes razones:

1. **Estructuración del lenguaje:** Define de manera precisa los elementos permitidos en el lenguaje, estableciendo sus reglas y delimitaciones.
2. **Optimización del análisis:** Al categorizar cada elemento del código fuente, se reduce la complejidad del análisis sintáctico y semántico, permitiendo que el compilador trabaje de manera más eficiente.
3. **Prevención de errores:** Facilita la detección de errores léxicos, como el uso de elementos no válidos, y contribuye a generar mensajes de error más claros para el usuario.



# MANUAL DE PRÁCTICAS

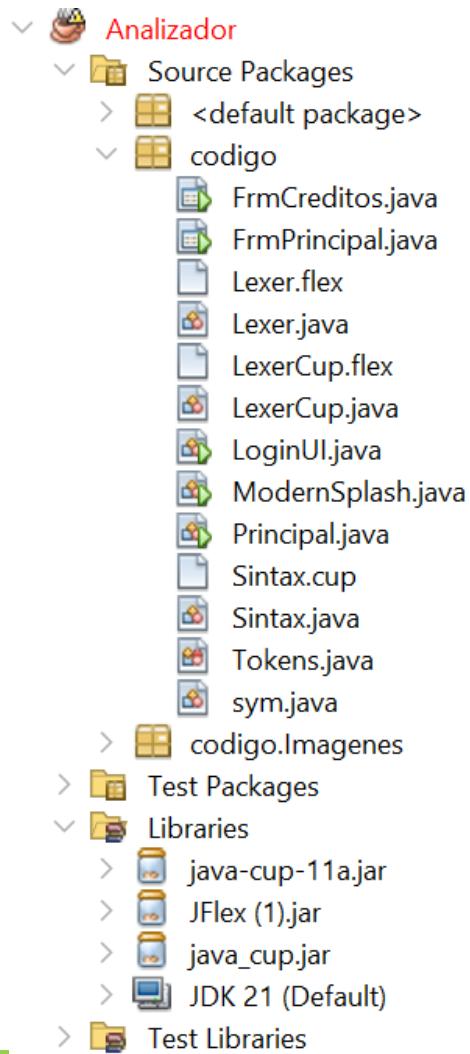


TABLA DE TOKENS			
	# TOKEN	TOKEN	LEXEMA
Operadores Binarios	1	operadorSuma	'+'
	2	operadorResta	'-'
	3	operadorMultiplicacion	'*'
	4	operadorDivision	'/'
	5	operadorModulo	'%'
	6	operadorPotencia	'^'
	7	operadorRaiz	'Rcuad'
	8	asignacion	'=>'
Operador Unario	9	incremento	'++'
	10	decremento	'--'
	11	operadorPositivo	'Pos'
	12	operadorNegativo	'Neg'
Operadores Comparacion	13	comparacionIgualdad	'=='
	14	mayorQue	
	15	menorQue	'<'
	16	mayorIgual	'>='
	17	menorIgual	'<='
Operador Logico	18	operadorY	'&'
	19	operadorO	' '
	20	operadorNo	'!'
Alfabeto	21	letrasMin	[a-z]'
	22	letrasMay	[A-Z]'
	23	numero	[0-9]'
Comentarios	24	inicioComentarioMult	'(:'
	25	finalComentarioMult	')'
	26	comentarioLinea	')=)'
Tipos de Datos	27	valorEntero	'Ent'
	28	valorFlotante	'Flot'
	29	valorBooleano	'Bool'
	30	cadena	'Cad'
	31	caracter	'Cars'
Simbolos Especiales	32	finLinea	'-;'
	33	saltoLinea	'\sl'
	34	inicioTexto	'<<'
	35	finalTexto	'>>'
	36	parentIzq	'('
	37	parentDer	')'
	38	llaveIzq	{'}
	39	llaveDer	'}'
	40	corchetIzq	'['
	41	corchetDer	']'
	42	concatenar	'-:'
	43	arreglo	'[]-[]'
	44	puntoAcceso	'. '
	45	puntoDecimial	'.'
	46	dosPuntos	::'
	47	espacio	' '
	48	tabulacion	'\t'
	49	comillaDoble	""
	50	comillaSimple	'"
	51	coma	','
	52	guionBajo	'_'
	53	guionMedio	'-'
	54	diagonalInvertido	'\'
	55	diagonal	'/'
	56	signoAdmirationAbre	'@'
	57	signoAdmirationCierra	'@'
	58	gato	'#'
	59	pesos	'\$'



Palabras reservadas	60	condicionalIf	'si'
	61	condicionalElse	'siNo'
	62	cicloFor	'ciclo'
	63	cicloWhile	'whilo'
	64	cicloDo	'hacer'
	65	funcionSwitch	'segun'
	66	funcionCase	'caso'
	67	funcionSalirCase	'salir'
	68	predeterminado	'predeterminado'
	69	detener	'Detener'
	70	constante	'Constante'
	71	variable	'Variable'
	72	funcion	'funcion'
	73	clase	'Clase'
	74	imprimir	'Imprimir'
	75	importar	'Importar'
	76	funcionLeer	'Leer'
	77	funcionIntroducirDatos	'IntroducirD'
Constantes	78	pi	'PI'
	79	euler	'E'

## Estructura Del Proyecto



## DESCRIPCIÓN DE CADA UNO DE LOS ARCHIVOS GENERADOS EN EL PROYECTO

### Archivo Lexer.flex

El archivo Lexer.flex es un componente clave en el desarrollo de un compilador o intérprete. Este archivo define el analizador léxico, una herramienta que descompone el código fuente en unidades fundamentales llamadas *tokens*. Un *token* es una representación simbólica de elementos como operadores, palabras reservadas, identificadores, números, o símbolos especiales.

El propósito principal de este archivo es procesar y clasificar el código fuente de manera sistemática, eliminando elementos irrelevantes como espacios y comentarios, para facilitar el análisis sintáctico en las etapas posteriores del procesamiento del lenguaje. Además, al manejar errores mediante el token ERROR, el analizador contribuye a detectar entradas no válidas en el código, mejorando la eficiencia del sistema.

Este archivo actúa como el primer filtro del procesamiento del lenguaje, estableciendo una base sólida para que el resto del sistema interprete o compile el código fuente de manera eficiente y precisa. Es esencial para automatizar el reconocimiento de los elementos del lenguaje y garantizar que el código fuente sea correctamente comprendido, minimizando errores y facilitando el desarrollo general del proyecto.

### Código Realizado:

```
1 package codigo;
2 import static codigo.Tokens.*;
3 
4 %%
5 %class Lexer
6 %type Tokens
7 L=[a-zA-Z_]+
8 D=[0-9]+
9 espacio=[ ,\t,\r]+
10 {
11     public String lexeme;
12 }
13 %%
14 /* Espacios en blanco */
15 {espacio} /*Ignore*/
16 
17 /* Comentarios */
18 ///*.* /*Ignore*/
19 
20 /* Salto de linea */
21 "\n" {return Linea;}
22 
23 /* Comillas */
24 """ {lexeme=yytext(); return comillaDoble;}
25 ''' {lexeme=yytext(); return comillaSimple;}
26 
27 /* Tipos de datos */
28 "Ent" {lexeme=yytext(); return valorEntero;}
29 "Flot" {lexeme=yytext(); return valorFlotante;}
30 "Boo" {lexeme=yytext(); return valorBooleano;}
31 "Cad" {lexeme=yytext(); return cadena;}
32 "Cars" {lexeme=yytext(); return carácter;}
```



```
34 /* Palabras reservadas */
35 "si" {lexeme=yytext(); return condicionalIf;}
36 "siNo" {lexeme=yytext(); return condicionalElse;}
37 "ciclo" {lexeme=yytext(); return cicloFor;}
38 "mientras" {lexeme=yytext(); return cicloWhile;}
39 "hacer" {lexeme=yytext(); return cicloDo;}
40 "segun" {lexeme=yytext(); return funcionSwitch;}
41 "caso" {lexeme=yytext(); return funcionCase;}
42 "salir" {lexeme=yytext(); return funcionSalirCase;}
43 "predeterminado" {lexeme=yytext(); return predeterminado;}
44 "Detener" {lexeme=yytext(); return detener;}
45 "Constante" {lexeme=yytext(); return constante;}
46 "Variable" {lexeme=yytext(); return variable;}
47 "funcion" {lexeme=yytext(); return funcion;}
48 "Clase" {lexeme=yytext(); return clase;}
49 "Imprimir" {lexeme=yytext(); return imprimir;}
50 "Leer" {lexeme=yytext(); return funcionLeer;}
51 "IntroducirD" {lexeme=yytext(); return funcionIntroducirDatos;}
52
53 /* Operadores matemáticos */
54 "+" {lexeme=yytext(); return operadorSuma;}
55 "-" {lexeme=yytext(); return operadorResta;}
56 "*" {lexeme=yytext(); return operadorMultiplicacion;}
57 "/" {lexeme=yytext(); return operadorDivision;}
58 "%" {lexeme=yytext(); return operadorModulo;}
59 "^" {lexeme=yytext(); return operadorPotencia;}
60 "Rcuad" {lexeme=yytext(); return operadorRaiz;}
61
62 /* Operadores de asignación y comparación */
63 ">=" {lexeme=yytext(); return asignacion;}
64 "==" {lexeme=yytext(); return comparacionIgualdad;}
65 ">" {lexeme=yytext(); return mayorQue;}
66 "<" {lexeme=yytext(); return menorQue;}
67
67 "<" {lexeme=yytext(); return menorQue;}
68 ">=<" {lexeme=yytext(); return mayorIgual;}
69 "<=>" {lexeme=yytext(); return menorIgual;}
69
70 /* Operadores lógicos */
71 "s" {lexeme=yytext(); return operadorY;}
72 "o" {lexeme=yytext(); return operadorO;}
73 "no" {lexeme=yytext(); return operadorNo;}
74
75 /* Operadores incrementales */
76 "++" {lexeme=yytext(); return incremento;}
77 "-+" {lexeme=yytext(); return decremento;}
78 /* Caracteres especiales */
79 "(" {lexeme=yytext(); return parentIzq;}
80 ")" {lexeme=yytext(); return parentDer;}
81 "{" {lexeme=yytext(); return llaveIzq;}
82 "}" {lexeme=yytext(); return llaveDer;}
83 "[" {lexeme=yytext(); return corchetIzq;}
84 "]" {lexeme=yytext(); return corchetDer;}
85 ";" {lexeme=yytext(); return finLinea;}
86 "<<" {lexeme=yytext(); return inicioTexto;}
87 ">>" {lexeme=yytext(); return finalTexto;}
88 "." {lexeme=yytext(); return puntoDecimal;}
89 ":" {lexeme=yytext(); return dosPuntos;}
90 " " {lexeme=yytext(); return espacio;}
91 /* Identificadores y números */
92 ({L}|{D})* {lexeme=yytext(); return Identificador;}
93 ("-"{D}+"){D}+ {lexeme=yytext(); return Número;}
94
95 /* Errores */
96 . {return ERROR;}
```

## Archivo Lexer.java (Generado automáticamente)

El archivo Lexer.java es una clase generada automáticamente por la herramienta JFlex, diseñada para construir un analizador léxico o escáner. Este archivo se encarga de identificar los distintos componentes léxicos o tokens de un lenguaje definido en una especificación, en este caso el archivo Lexer.flex. Los tokens pueden representar palabras clave, identificadores, operadores, literales, y otros elementos sintácticos básicos del lenguaje.

El analizador léxico es una parte esencial de un compilador o intérprete, ya que actúa como el primer paso en el análisis del código fuente. Este archivo tiene la responsabilidad de leer el flujo de entrada de caracteres y convertirlo en una secuencia organizada de tokens que el analizador sintáctico puede entender. Su importancia radica en garantizar que el código fuente se procese de manera eficiente y que cualquier error léxico se detecte desde etapas tempranas del análisis. Este proceso es fundamental para garantizar la correcta interpretación y ejecución del código fuente.

### Código Generado:

```
1  /* The following code was generated by JFlex 1.4.3 on 01/01/25, 12:03 */
2
3  package codigo;
4  import static codigo.Tokens.*;
5
6  /**
7   * This class is a scanner generated by
8   * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9   * on 01/01/25, 12:03 from the specification file
10  * <tt>C:/Users/jesus/Downloads/AnalizadorLexico/src/codigo/Lexer.flex</tt>
11  */
12  class Lexer {
13
14  /** This character denotes the end of file */
15  public static final int YYEOF = -1;
16
17  /** initial size of the lookahead buffer */
18  private static final int ZZ_BUFSIZE = 16384;
19
20  /** lexical states */
21  public static final int YYINITIAL = 0;
22
23  /**
24   * ZZ_LEXSTATE[l] is the state in the DFA for the lexical state l
25   * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
26   * at the beginning of a line
27   * l is of the form l = 2*k, k a non negative integer
28   */
29  private static final int ZZ_LEXSTATE[] = {
30      0, 0
31  };
32
33  /**
```



# **MANUAL DE PRACTICAS**



```
34     * Translates characters to character classes
35     */
36     private static final String ZZ_CMAP_PACKED =
37         "\u11\u01\u31\u16\u20\u11\u32\u01\u47\u14\u21\u65\u171+" +
38         "\u11\u60\u11\u22\u14\u40\u11\u66\u14\u45\u11\u50\u11\u21\u11\u17\u13\u120+" +
39         "\u11\u64\u11\u15\u12\u11\u46\u11\u57\u11\u37\u11\u31\u132\u20\u11\u44+" +
40         "\u11\u54\u11\u55\u11\u74\u11\u52\u11\u53\u24\u44\u11\u77\u21\u44\u11\u100\u11\u44+" +
41         "\u11\u35\u11\u44\u11\u33\u11\u44\u11\u24\u3\u44\u11\u75\u4\u44\u11\u62\u11\u67+" +
42         "\u11\u63\u11\u23\u11\u11\u01\u27\u11\u76\u11\u25\u11\u30\u11\u10\u11\u7+" +
43         "\u11\u36\u11\u14\u11\u2\u4\u2\u43\u11\u11\u17\u2\u11\u5\u11\u34\u11\u73\u11\u43+" +
44         "\u11\u56\u11\u12\u11\u6\u11\u26\u11\u43\u11\u13\u3\u43\u11\u61\u11\u41\u11\u51+" +
45         "\u11\u43\u01\u11\u70\uuff5e\u10";
46
47     /**
48     * Translates characters to character classes
49     */
50     private static final char [] ZZ_CMAP = zzUnpackCMap(ZZ_CMAP_PACKED);
51
52     /**
53     * Translates DFA states to action switch labels.
54     */
55     private static final int [] ZZ_ACTION = zzUnpackAction();
56
57     private static final String ZZ_ACTION_PACKED_0 =
58         "\u10\u11\u12\u13\u14\u15\u16\u17+" +
59         "\u11\u10\u11\u12\u13\u14\u15\u11\u15+" +
60         "\u12\u14\u11\u16\u17\u12\u20\u11\u21\u11\u14\u11\u22\u11\u23+" +
61         "\u11\u24\u11\u25\u11\u14\u11\u11\u26\u11\u27\u11\u30\u11\u31+" +
62         "\u11\u32\u11\u33\u11\u34\u11\u35\u11\u36\u11\u37\u2\u54\u11\u14+" +
63         "\u11\u40\u11\u41\u11\u40\u11\u42\u2\u40\u11\u43\u4\u40\u11\u4+" +
64         "\u11\u44\u11\u45\u11\u40\u11\u46\u11\u47\u11\u01\u11\u50\u11\u40+" +
65         "\u11\u51\u11\u40\u11\u01\u11\u52\u11\u01\u11\u53\u11\u06\u40+" +
66         "\u11\u01\u54\u11\u01\u21\u40\u11\u55\u11\u56\u11\u57\u11\u60+" +
67
68         "\u11\u01\u61\u11\u62\u11\u63\u11\u64\u11\u40\u11\u65\u11\u40+" +
69         "\u11\u66\u2\u40\u11\u67\u11\u01\u10\u40\u11\u70\u5\u40\u11\u71+" +
70         "\u11\u41\u11\u72\u11\u40\u11\u73\u11\u40\u11\u01\u7\u40\u11\u74+" +
71         "\u11\u40\u11\u75\u11\u76\u11\u77\u11\u100\u11\u101\u11\u102\u11\u40+" +
72         "\u11\u103\u20\u40\u11\u104\u3\u11\u40\u11\u105\u5\u40\u11\u106\u11\u40+" +
73         "\u11\u107\u11\u40\u11\u110\u11\u111\u11\u112\u5\u40\u11\u113\u2\u40+" +
74         "\u11\u114";
75
76     private static int [] zzUnpackAction() {
77         int [] result = new int[208];
78         int offset = 0;
79         offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);
80         return result;
81     }
82
83     private static int zzUnpackAction(String packed, int offset, int [] result) {
84         int i = 0; /* index in packed string */
85         int j = offset; /* index in unpacked array */
86         int l = packed.length();
87         while (i < l) {
88             int count = packed.charAt(i++);
89             int value = packed.charAt(i++);
90             do result[j++] = value; while (--count > 0);
91         }
92         return j;
93     }
94
95     /**
96     * Translates a state to a row index in the transition table
97     */
98     private static final int [] ZZ_ROWMAP = zzUnpackRowMap();
```



```
100 private static final String ZZ_ROWMAP_PACKED_0 =
101     "\u00a0\u00a0\u101\u00a0\u202\u00a0\u303\u00a0\u00a0\u0104\u00a0\u00a0\u0145\u00a0\u202\u00a0\u00a0\u0186"+
102     "\u00a0\u01c7\u00a0\u00a0\u208\u00a0\u00a0\u0249\u00a0\u00a0\u028a\u00a0\u00a0\u02cb\u00a0\u00a0\u030c\u00a0\u00a0\u034d\u00a0\u101"+
103     "\u00a0\u101\u00a0\u101\u00a0\u00a0\u038e\u00a0\u00a0\u03cf\u00a0\u00a0\u0410\u00a0\u00a0\u0451\u00a0\u00a0\u0492\u00a0\u00a0\u04d3"+
104     "\u00a0\u0514\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u202\u00a0\u00a0\u0555\u00a0\u00a0\u0596\u00a0\u101"+
105     "\u00a0\u05d7\u00a0\u00a0\u0618\u00a0\u00a0\u0659\u00a0\u00a0\u069a\u00a0\u00a0\u06db\u00a0\u00a0\u071c\u00a0\u00a0\u075d\u00a0\u101"+
106     "\u00a0\u079e\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101"+
107     "\u00a0\u07df\u00a0\u00a0\u0820\u00a0\u00a0\u0861\u00a0\u00a0\u0882\u00a0\u00a0\u08e3\u00a0\u00a0\u0924\u00a0\u202\u00a0\u303"+
108     "\u00a0\u0965\u00a0\u202\u00a0\u00a0\u09a6\u00a0\u00a0\u09e7\u00a0\u00a0\u0a28\u00a0\u00a0\u0aa\u00a0\u00a0\u0aeb"+
109     "\u00a0\u0b2c\u00a0\u00a0\u0b6d\u00a0\u101\u00a0\u101\u00a0\u00a0\u0bae\u00a0\u00a0\u0bef\u00a0\u00a0\u0c30\u00a0\u101"+
110     "\u00a0\u101\u00a0\u0c71\u00a0\u101\u00a0\u00a0\u0cb2\u00a0\u202\u00a0\u00a0\u0cf3\u00a0\u00a0\u0d34\u00a0\u101"+
111     "\u00a0\u0d75\u00a0\u00a0\u0db6\u00a0\u00a0\u0df7\u00a0\u101\u00a0\u00a0\u0e38\u00a0\u00a0\u0eba\u00a0\u00a0\u0efb"+
112     "\u00a0\u0f3c\u00a0\u00a0\u0fd7\u00a0\u00a0\u0fbef\u00a0\u00a0\u101\u00a0\u00a0\u1040\u00a0\u00a0\u1081\u00a0\u10c2"+
113     "\u00a0\u1103\u00a0\u1144\u00a0\u1185\u00a0\u00a0\u11c6\u00a0\u00a0\u1207\u00a0\u00a0\u1248\u00a0\u00a0\u1289\u00a0\u12ca"+
114     "\u00a0\u130b\u00a0\u134c\u00a0\u138d\u00a0\u00a0\u13ce\u00a0\u00a0\u140f\u00a0\u00a0\u1450\u00a0\u00a0\u1491\u00a0\u101"+
115     "\u00a0\u202\u00a0\u202\u00a0\u101\u00a0\u00a0\u14d2\u00a0\u101\u00a0\u00a0\u101\u00a0\u00a0\u202"+
116     "\u00a0\u1513\u00a0\u202\u00a0\u00a0\u1554\u00a0\u202\u00a0\u00a0\u1599\u00a0\u00a0\u15d6\u00a0\u101\u00a0\u00a0\u1617"+
117     "\u00a0\u1658\u00a0\u00a0\u1699\u00a0\u00a0\u16da\u00a0\u00a0\u171b\u00a0\u00a0\u175c\u00a0\u00a0\u179d\u00a0\u00a0\u17de\u00a0\u00a0\u181f"+
118     "\u00a0\u202\u00a0\u1860\u00a0\u00a0\u18a1\u00a0\u00a0\u18e2\u00a0\u00a0\u1923\u00a0\u00a0\u1964\u00a0\u202\u00a0\u101"+
119     "\u00a0\u202\u00a0\u19a5\u00a0\u202\u00a0\u00a0\u199e\u00a0\u00a0\u1a27\u00a0\u00a0\u1a68\u00a0\u00a0\u1aa9\u00a0\u00a0\u1aea"+
120     "\u00a0\u1b2b\u00a0\u00a0\u1b6c\u00a0\u00a0\u1bad\u00a0\u00a0\u1bee\u00a0\u202\u00a0\u00a0\u1c2f\u00a0\u00a0\u202\u00a0\u202"+
121     "\u00a0\u202\u00a0\u202\u00a0\u00a0\u1c70\u00a0\u00a0\u101\u00a0\u00a0\u1cb1\u00a0\u00a0\u1cf2"+
122     "\u00a0\u01d3\u00a0\u00a0\u1d74\u00a0\u00a0\u1fdb5\u00a0\u00a0\u1df6\u00a0\u00a0\u1e37\u00a0\u00a0\u1e78\u00a0\u00a0\u1e99\u00a0\u00a0\u1efa"+
123     "\u00a0\u1f3b\u00a0\u00a0\u1f7c\u00a0\u00a0\u1fbf\u00a0\u00a0\u1ffe\u00a0\u00a0\u203f\u00a0\u00a0\u2080\u00a0\u202\u00a0\u20c1"+
124     "\u00a0\u2102\u00a0\u2143\u00a0\u202\u00a0\u2184\u00a0\u00a0\u21c5\u00a0\u00a0\u2206\u00a0\u00a0\u2247\u00a0\u00a0\u2288"+
125     "\u00a0\u202\u00a0\u22c9\u00a0\u202\u00a0\u00a0\u230a\u00a0\u00a0\u202\u00a0\u00a0\u202\u00a0\u00a0\u234b"+
126     "\u00a0\u238c\u00a0\u00a0\u23cd\u00a0\u00a0\u240e\u00a0\u00a0\u244f\u00a0\u00a0\u202\u00a0\u00a0\u2490\u00a0\u00a0\u24d1\u00a0\u00a0\u202";
127
128     private static int [] zzUnpackRowMap() {
129         int [] result = new int[208];
130         int offset = 0;
131         offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);
132         return result;
133     }
134
135     private static int zzUnpackRowMap(String packed, int offset, int [] result) {
136         int i = 0; /* index in packed string */
137         int j = offset; /* index in unpacked array */
138         int l = packed.length();
139         while (i < l) {
140             int high = packed.charAt(i++) << 16;
141             result[j++] = high | packed.charAt(i++);
142         }
143         return j;
144     }
145
146     /**
147      * The transition table of the DFA
148     */
149     private static final int [] ZZ_TRANS = zzUnpackTrans();
150
151     private static final String ZZ_TRANS_PACKED_0 =
152     "\u1\u2\u1\u3\u1\u4\u1\u5\u1\u6\u2\u7\u1\u10\u1\u11"+
153     "\u1\u7\u1\u12\u1\u13\u1\u14\u1\u15\u1\u16\u1\u17"+
154     "\u1\u20\u1\u21\u1\u22\u1\u23\u1\u24\u1\u25\u1\u26"+
155     "\u1\u27\u1\u1\u30\u1\u1\u31\u1\u32\u1\u33\u1\u34"+
156     "\u1\u1\u35\u1\u36\u1\u37\u1\u5\u1\u40\u1\u41\u1\u42"+
157     "\u1\u43\u1\u44\u1\u45\u1\u46\u1\u47\u1\u50\u1\u51"+
158     "\u1\u52\u1\u53\u1\u54\u1\u55\u1\u56\u1\u57\u1\u60\u1\u61"+
159     "\u1\u62\u1\u63\u1\u64\u1\u65\u1\u66\u1\u67\u1\u68"+
160     "\u1\u69\u1\u1\u70\u1\u71\u1\u72\u1\u73\u1\u74\u1\u75"+
161     "\u5\u0\u5\u67\u1\u3\u0\u7\u1\u67\u2\u0\u1\u70\u1\u0\u1\u5"+
162     "\u1\u2\u0\u1\u3\u1\u30\u0\u1\u5\u32\u0\u2\u67\u1\u0\u1\u67"+
163     "\u1\u71\u1\u67\u1\u72\u1\u67\u1\u67\u1\u67\u2\u0\u4\u67"+
164     "\u4\u0\u2\u67\u1\u5\u0\u5\u67\u1\u3\u0\u1\u76\u1\u0\u2\u67"+
165     "\u1\u0\u1\u1\u67\u1\u70\u2\u67\u1\u73\u2\u67\u2\u0\u4\u67";
```



# MANUAL DE PRÁCTICAS



166 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\0\2\67"+  
167 "1\1\0\5\67\1\74\3\67\7\0\5\67\2\0\4\67"+  
168 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\0\2\67"+  
169 "1\1\0\1\1\75\3\67\1\1\76\4\67\7\1\0\3\67\1\1\77"+  
170 "1\1\67\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\0"+  
171 "1\7\67\1\1\0\2\67\1\0\1\0\67\1\1\100\7\0\5\67"+  
172 "1\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\7\67"+  
173 "1\1\0\2\67\1\1\0\1\1\67\7\0\3\67\1\1\101\1\67"+  
174 "1\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\7\67"+  
175 "1\1\5\0\1\1\102\1\0\1\1\103\1\0\1\1\104\61\0\2\67"+  
176 "1\1\0\1\1\1\67\1\1\0\1\1\67\1\1\105\3\67\2\0\1\4\67"+  
177 "1\4\0\2\67\5\0\5\67\1\3\0\1\67\1\1\0\2\67"+  
178 "1\1\0\1\1\106\1\0\1\67\7\0\3\67\1\1\107\1\67\2\0\0"+  
179 "1\4\0\67\4\0\2\67\5\0\5\67\1\3\0\7\67\3\1\0\0"+  
180 "1\1\110\1\1\111\77\0\1\1\112\1\1\113\4\7\1\0\2\67\1\1\0\0"+  
181 "1\1\1\67\7\1\0\5\67\1\2\0\1\67\1\1\114\2\1\67\4\1\0\0"+  
182 "1\2\1\67\5\1\0\5\67\1\3\0\5\67\1\1\115\1\1\67\1\1\0\0"+  
183 "1\2\1\67\1\1\0\4\67\1\1\116\4\67\7\0\5\67\1\2\0\0"+  
184 "1\4\0\67\4\0\2\67\5\0\5\67\1\3\0\7\67\3\1\0\0"+  
185 "1\1\117\5\0\1\1\120\61\0\1\1\121\25\0\1\1\122\52\5\0\0"+  
186 "1\1\123\27\0\1\1\124\61\0\1\1\125\50\0\1\2\67\1\1\0\0"+  
187 "1\1\167\1\1\126\7\1\67\7\0\5\67\2\0\1\4\67\1\4\1\0\0"+  
188 "1\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
189 "1\5\1\67\1\1\127\3\1\67\7\0\5\67\2\0\0\4\67\1\4\1\0\0"+  
190 "1\2\1\67\5\1\0\5\67\1\3\0\7\1\67\1\1\10\2\67\1\1\0\0"+  
191 "1\1\1\67\7\1\0\5\67\1\2\0\1\67\1\1\130\2\1\67\4\1\0\0"+  
192 "1\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
193 "1\5\1\67\1\1\131\3\1\67\7\1\0\3\67\1\1\132\1\1\67\2\0\0"+  
194 "1\1\1\67\1\1\133\2\1\67\4\0\1\2\67\5\0\1\5\67\1\3\0\0"+  
195 "1\7\1\67\1\2\0\0\1\1\34\7\1\0\1\1\135\152\0\1\1\136\16\0\0"+  
196 "1\2\1\67\1\0\1\1\137\10\1\67\7\0\5\67\2\0\1\4\67"+  
197 "1\4\0\2\67\5\0\5\67\1\3\0\1\7\67\1\1\0\2\67"+  
198 "1\1\0\1\1\67\1\7\0\5\67\2\0\1\4\67\1\0\2\67"+

199 "1\5\1\0\4\67\1\1\140\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
200 "1\4\1\67\1\1\141\4\1\67\7\0\5\67\2\0\1\4\67\1\4\1\0\0"+  
201 "1\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
202 "1\1\1\67\7\1\0\3\67\1\1\142\3\1\67\2\0\1\4\67\1\4\1\0\0"+  
203 "1\2\1\67\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
204 "1\1\1\67\1\1\143\1\67\7\1\0\5\67\2\0\1\4\67\1\4\1\0\0"+  
205 "1\2\1\67\5\1\0\5\67\1\3\0\1\144\6\1\67\1\1\10\2\67"+  
206 "1\1\1\0\4\67\1\1\145\4\67\7\1\0\5\67\2\0\1\4\67"+  
207 "1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
208 "1\1\0\2\1\67\1\1\172\6\1\67\1\7\1\0\5\67\2\0\1\4\67"+  
209 "1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
210 "1\1\0\1\1\67\1\1\146\7\1\67\7\1\0\5\67\2\0\1\4\67"+  
211 "1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
212 "1\1\0\1\1\67\1\1\147\1\2\67\1\7\1\0\5\67\2\0\1\4\67"+  
213 "1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
214 "1\1\0\1\1\67\1\7\0\5\67\1\2\1\0\2\67\1\1\150\1\67"+  
215 "1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
216 "1\1\0\1\1\67\1\7\0\5\67\1\2\1\0\3\67\1\1\151\1\4\1\0\0"+  
217 "1\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
218 "1\1\1\67\1\1\152\3\1\67\1\7\1\0\5\67\2\0\1\4\67\1\4\1\0\0"+  
219 "1\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
220 "1\1\1\53\1\0\67\1\3\0\5\67\1\2\0\1\4\67\1\4\1\0\2\67"+  
221 "1\1\0\5\67\1\1\13\0\7\67\1\1\10\2\67\1\1\0\1\1\67"+  
222 "1\1\0\1\1\67\1\1\154\3\1\67\2\0\1\4\67\1\4\1\0\2\67"+  
223 "1\1\0\5\67\1\1\13\0\7\67\1\1\16\102\1\1\0\62\1\0\2\1\1\0\0"+  
224 "1\2\1\67\1\1\0\1\1\67\1\7\1\0\2\67\1\1\155\2\1\67\2\0\0"+  
225 "1\1\1\67\1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
226 "1\2\1\67\1\1\0\1\1\67\1\7\1\0\1\1\67\1\1\156\3\1\67\2\0\0"+  
227 "1\1\1\67\1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
228 "1\2\1\67\1\1\0\1\1\67\1\1\157\2\1\67\1\7\1\0\5\67\2\0\0"+  
229 "1\1\1\67\1\1\0\2\1\67\5\1\0\5\67\1\3\0\7\67\1\1\10\2\67"+  
230 "1\1\1\60\1\42\1\0\2\1\67\1\3\0\1\67\1\1\161\1\2\67\1\7\1\0\0"+  
231 "1\1\1\67\1\2\0\1\4\67\1\4\0\1\2\67\5\1\0\5\67\1\3\0\0"+



# MANUAL DE PRÁCTICAS



232 "x\1\67\1\0\2\67\1\0\1\1\67\1\0\5\67\1\2\0" +  
233 "x\3\67\1\162\4\0\2\67\5\0\5\67\1\3\0\7\67" +  
234 "x\3\2\0\1\163\5\0\1\164\145\0\1\165\7\0\1\166" +  
235 "x\1\0\2\0\1\167\3\1\0\2\67\1\0\1\2\67\1\1\170\6\67" +  
236 "x\7\0\1\5\67\1\2\0\4\67\4\0\2\67\5\0\5\67" +  
237 "x\3\2\0\1\7\67\1\1\171\2\67\4\0\1\2\67\5\0\5\67" +  
238 "x\2\0\1\1\67\1\1\171\2\67\1\1\0\1\1\67\1\0\5\67" +  
239 "x\1\3\0\1\7\67\1\1\0\2\67\1\1\0\1\1\67\1\0\5\67" +  
240 "x\2\0\1\1\67\1\1\172\2\67\1\4\0\2\1\67\5\0\5\67" +  
241 "x\1\3\0\1\7\67\1\1\0\2\67\1\1\0\1\1\67\1\0\3\67" +  
242 "x\1\1\73\1\1\67\1\2\0\4\67\4\0\2\67\5\0\5\67" +  
243 "x\1\3\0\1\7\67\1\1\0\2\67\1\1\0\1\1\67\1\0\4\67" +  
244 "x\1\1\74\1\2\0\4\67\1\1\0\2\67\5\0\4\67\1\1\175" +  
245 "x\1\3\0\1\7\67\1\1\0\2\67\1\1\0\1\1\67\1\1\176\7\67" +  
246 "x\7\0\1\5\67\1\2\0\4\67\4\0\2\67\5\0\5\67" +  
247 "x\1\3\0\1\7\67\5\7\0\1\1\77\4\1\0\1\2\0\0\61\0\2\1\67" +  
248 "x\1\1\0\1\4\67\1\1\2\0\1\4\67\1\0\5\67\1\2\0\0\4\67" +  
249 "x\4\0\1\2\67\5\0\5\67\1\3\1\0\7\67\1\1\0\2\1\67" +  
250 "x\1\1\0\1\4\67\1\1\2\0\2\4\67\1\7\0\1\5\67\1\2\0\4\67" +  
251 "x\4\0\1\2\67\5\0\5\67\1\3\1\0\7\67\1\1\0\2\1\67" +  
252 "x\1\1\0\1\2\67\1\1\2\0\3\67\1\7\0\1\5\67\1\2\0\4\67" +  
253 "x\4\0\1\2\67\5\0\5\67\1\3\1\0\7\67\1\1\0\2\1\67" +  
254 "x\1\1\0\1\1\1\67\1\1\0\5\67\1\2\0\4\67\1\0\2\1\67" +  
255 "x\5\0\1\4\67\1\1\2\0\4\13\0\7\67\1\1\0\2\1\67\1\3\0" +  
256 "x\2\1\67\1\1\2\0\5\67\1\1\0\5\67\1\2\0\4\13\0\7\67" +  
257 "x\2\1\67\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
258 "x\1\1\67\1\7\0\1\5\67\1\2\0\4\167\4\0\1\2\67\5\0\5" +  
259 "x\5\1\67\1\1\3\0\1\67\1\1\2\0\5\67\1\1\0\2\1\67\1\1\0" +  
260 "x\4\1\67\1\1\2\0\7\4\67\1\7\0\1\5\67\1\2\0\4\167\1\1\0" +  
261 "x\2\1\67\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
262 "x\1\1\67\1\7\0\1\1\67\1\1\2\1\0\3\67\1\7\1\2\0\4\167\1\4\0" +  
263 "x\2\1\67\1\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
264 "x\4\1\67\1\1\2\1\4\67\1\7\0\1\5\67\1\2\0\4\167\1\4\0" +

265 "x\2\1\67\1\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
266 "x\1\1\67\1\7\0\1\5\67\1\2\0\1\67\1\1\2\1\1\2\1\67\4\0\0" +  
267 "x\2\1\67\1\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
268 "x\1\1\67\1\7\0\1\67\1\1\2\1\2\1\67\1\2\0\1\4\167\4\0\0" +  
269 "x\2\1\67\1\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0" +  
270 "x\1\1\2\1\3\1\0\67\1\7\0\1\5\67\1\2\0\1\4\167\4\0\1\2\67" +  
271 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0\1\5\67" +  
272 "x\1\1\1\4\7\1\3\1\67\1\7\0\1\5\67\1\2\0\1\4\167\4\1\0\2\67" +  
273 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0\1\4\67" +  
274 "x\1\1\2\1\4\1\67\1\7\0\1\5\67\1\2\0\1\4\167\4\1\0\2\67" +  
275 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0\1\1\1\67" +  
276 "x\7\0\1\3\67\1\1\2\1\5\1\67\1\2\0\1\4\167\4\1\0\2\67" +  
277 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0\1\5\67" +  
278 "x\1\1\2\1\6\1\3\1\67\1\7\0\1\5\67\1\2\0\1\4\167\4\1\0\2\67" +  
279 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\1\0\2\1\67\1\1\0\1\1\1\67" +  
280 "x\7\0\1\5\67\1\2\0\1\1\67\1\1\2\1\2\1\67\1\1\0\2\1\67" +  
281 "x\5\1\0\5\67\1\1\3\0\1\7\67\1\2\0\1\1\164\1\5\1\0\1\2\20" +  
282 "x\3\1\0\1\2\1\67\1\1\0\2\1\67\1\1\2\2\1\6\1\67\1\7\0\1\5\67" +  
283 "x\2\1\0\1\4\67\1\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\1\7\67" +  
284 "x\1\1\0\1\2\67\1\1\0\1\1\67\1\1\2\2\1\2\1\67\1\7\0\1\5\67" +  
285 "x\2\1\0\1\4\67\1\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\1\7\67" +  
286 "x\1\1\0\1\2\67\1\1\0\1\1\67\1\1\2\2\1\2\1\67\1\7\0\1\5\67" +  
287 "x\2\1\0\1\4\67\1\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\1\7\67" +  
288 "x\1\1\0\1\2\67\1\1\0\1\1\67\1\1\2\2\1\2\1\67\1\7\0\1\5\67" +  
289 "x\2\1\0\1\4\67\1\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\1\7\67" +  
290 "x\6\2\0\1\1\2\2\1\7\1\0\1\2\1\67\1\1\0\1\1\67\1\1\2\2\1\7\67" +  
291 "x\7\1\0\5\67\1\2\0\1\4\167\4\0\1\2\1\67\1\5\0\1\5\67" +  
292 "x\1\3\0\1\7\67\1\1\0\1\2\1\67\1\1\0\1\1\67\1\7\0\1\4\67" +  
293 "x\1\1\2\2\1\2\1\0\4\167\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\0" +  
294 "x\7\1\0\1\2\1\67\1\1\0\1\1\67\1\1\2\3\0\1\4\167\1\7\0\1\4\67" +  
295 "x\5\1\67\1\2\1\0\4\167\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\0" +  
296 "x\7\1\1\0\1\2\1\67\1\1\0\1\1\67\1\1\2\3\1\1\0\67\7\1\0\1\5\67" +  
297 "x\2\1\0\1\4\67\1\4\0\1\2\1\67\1\5\0\1\5\67\1\1\3\0\1\7\67"



# MANUAL DE PRÁCTICAS



298 "11026710116717056720467"+  
299 "41026715014671232130767110"+  
300 "21671101116717056712011671233"+  
301 "2167141026715014671234130767"+  
302 "11026711011167170567201467"+  
303 "41026715014671235130767110"+  
304 "216711011236106710567201467"+  
305 "4102671501467123107671101267"+  
306 "110167123717671705671201467"+  
307 "4102167150146711307671101267"+  
308 "110111671705672014671401267"+  
309 "51046712401307671101267110"+  
310 "1116717056712014671401267150"+  
311 "41671241130767110126711011167"+  
312 "710467124212014671401267150"+  
313 "51671310767110126711011167170"+  
314 "516712014671124312671401267150"+  
315 "516713107671101267110146711244"+  
316 "41671705672014671401267150"+  
317 "51671310767110126711012671245"+  
318 "61671705672014671401267150"+  
319 "5167131076716310112461610267110"+  
320 "21671247161705671201467110"+  
321 "21671305671310767110267110"+  
322 "416712125014671705671201467110"+  
323 "21671305671310767110267110"+  
324 "11167125171671705671201467140"+  
325 "216715105671310767110267110"+  
326 "11167170367125211671201467140"+  
327 "216715105671310767110267110"+  
328 "111671705671201167112532167140"+  
329 "216715105671310767110267110"+  
330 "1116717056712014671401267150"+  
  
331 "41671125413071671102671301255"+  
332 "11016717056712014671401267150"+  
333 "51671310767110126711011167170"+  
334 "5167121011671125612671401267150"+  
335 "51671310767110126711011167170"+  
336 "316712125716712014671401267150"+  
337 "51671310767110126711011167170"+  
338 "51671210467140126715046711260"+  
339 "113017671201267110267112611667"+  
340 "71056712014671401267150567"+  
341 "1130176712012671101467112621467"+  
342 "7101567120146714012671501567"+  
343 "113017671210267110111671701567"+  
344 "210146714012671505671310467"+  
345 "112632167110267110111671701467"+  
346 "11264210146714012671505671310"+  
347 "71671102671101267112651667170"+  
348 "51671210146714012671505671310"+  
349 "7167110267110111671701567120"+  
350 "416714012671501567131012661667"+  
351 "1101267110111671267171671101567"+  
352 "21011671101267150156713107167"+  
353 "1101267110111671270171671101567"+  
354 "2101467140126715056713107167"+  
355 "1101267110111671710367112711367"+  
356 "2101467140126715056713107167"+  
357 "11012671101467127214671701567"+  
358 "2101467140126715056713107167"+  
359 "11012671101116717015671201467"+  
360 "41012671501467127313071671110"+  
361 "216713105671274136717015671200"+  
362 "416714012671501567131071671110"+  
363 "2167110111671102671127521671200"+



```
364 "14\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
365 "\2\67\1\0\11\67\7\0\3\67\1\276\1\67\2\0"+
366 "\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
367 "\2\67\1\0\1\277\10\67\7\0\5\67\2\0\4\67"+
368 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
369 "\1\0\2\67\1\300\6\67\7\0\5\67\2\0\4\67"+
370 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
371 "\1\0\6\67\1\301\2\67\7\0\5\67\2\0\4\67"+
372 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
373 "\1\0\11\67\7\0\5\67\2\0\4\67\4\0\2\67"+
374 "\5\0\4\67\1\302\13\0\7\67\1\0\2\67\1\0"+
375 "\4\67\1\1\303\4\67\7\0\5\67\2\0\4\67\4\0"+
376 "\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+
377 "\1\67\7\0\1\67\1\304\3\67\2\0\4\67\4\0"+
378 "\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+
379 "\1\1\67\7\0\5\67\12\0\4\67\1\0\2\67\5\0"+
380 "\4\67\1\1\305\13\0\7\67\1\0\2\67\1\0\11\67"+
381 "\7\0\5\67\2\0\4\67\1\0\2\67\1\0\2\67\0\4\67"+
382 "\1\306\13\0\7\67\1\0\2\67\1\0\4\67\1\307"+
383 "\4\67\1\0\5\67\2\0\4\67\4\0\2\67\5\0\0"+
384 "\5\67\13\0\7\67\1\0\2\67\1\0\11\67\7\0"+
385 "\5\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+
386 "\1\310\6\67\1\0\2\67\1\0\1\311\10\67\7\0"+
387 "\5\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+
388 "\7\67\1\0\2\67\1\0\1\312\10\67\7\0\5\67"+
389 "\2\0\4\67\4\0\2\67\5\0\5\67\13\0\7\67"+
390 "\1\0\2\67\1\0\11\67\7\0\5\67\2\0\4\67"+
391 "\4\0\2\67\5\0\4\67\1\313\13\0\7\67\1\0"+
392 "\2\67\1\0\1\67\1\314\7\67\7\0\5\67\2\0"+
393 "\4\67\1\0\2\67\5\0\5\67\13\0\7\67\1\0"+
394 "\2\67\1\0\11\67\7\0\5\67\2\0\4\67\4\0"+
395 "\2\67\5\0\5\67\13\0\2\67\1\315\4\67\1\0"+
396 "\2\67\1\0\11\67\7\0\3\67\1\316\1\67\2\0"+
```

```
397 "\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
398 "\2\67\1\0\11\67\7\0\4\67\1\317\2\0\4\67"+
399 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
400 "\1\0\11\67\7\0\5\67\2\0\4\67\1\320\2\67"+
401 "\4\0\2\67\5\0\5\67\13\0\7\67"+
402 private static int [] zzUnpackTrans() {
403     int [] result = new int[9490];
404     int offset = 0;
405     offset = zzUnpackTrans(SE_TRANS_PACKED_0, offset, result);
406     return result;
407 }
408
409 private static int zzUnpackTrans(String packed, int offset, int [] result) {
410     int i = 0; /* index in packed string */
411     int j = offset; /* index in unpacked array */
412     int l = packed.length();
413     while (i < l) {
414         int count = packed.charAt(i++);
415         int value = packed.charAt(i++);
416         value--;
417         do result[j++] = value; while (--count > 0);
418     }
419     return j;
420 }
421
422
423 /* error codes */
424 private static final int ZZ_UNKNOWN_ERROR = 0;
425 private static final int ZZ_NO_MATCH = 1;
426 private static final int ZZ_PUSHBACK_2BIG = 2;
427
428 /* error messages for the codes above */
```



## **MANUAL DE PRACTICAS**





```
496
497     /** the number of characters up to the start of the matched text */
498     private int yychar;
499
500     /**
501      * the number of characters from the last newline up to the start of the
502      * matched text
503      */
504     private int yycolumn;
505
506     /**
507      * zzAtBOL == true <=> the scanner is currently at the beginning of a line
508      */
509     private boolean zzAtBOL = true;
510
511     /**
512      * zzAtEOF == true <=> the scanner is at the EOF
513      */
514     private boolean zzAtEOF;
515
516     /**
517      * denotes if the user-EOF-code has already been executed
518      */
519     private boolean zEOFDone;
520
521     /**
522      * Creates a new scanner
523      * There is also a java.io.InputStream version of this constructor.
524      *
525      * @param in the java.io.Reader to read input from.
526      */
527     Lexer(java.io.Reader in) {
528         this.zzReader = in;
529     }
530
531     /**
532      * Creates a new scanner.
533      * There is also java.io.Reader version of this constructor.
534      *
535      * @param in the java.io.InputStream to read input from.
536      */
537     Lexer(java.io.InputStream in) {
538         this(new java.io.InputStreamReader(in));
539     }
540
541     /**
542      * Unpacks the compressed character translation table.
543      *
544      * @param packed the packed character translation table
545      * @return the unpacked character translation table
546      */
547     private static char[] zzUnpackCMap(String packed) {
548         char[] map = new char[0x10000];
549         int i = 0; /* index in packed string */
550         int j = 0; /* index in unpacked array */
551         while (i < 166) {
552             int count = packed.charAt(i++);
553             char value = packed.charAt(i++);
554             do map[j++] = value; while (--count > 0);
555         }
556         return map;
557     }
558
559
560     /**
561      * Refills the input buffer.
```



# MANUAL DE PRÁCTICAS



```
562      *
563      * @return      <code>false</code>, iff there was new input.
564      *
565      * @exception    java.io.IOException  if any I/O-Error occurs
566      */
567  private boolean zzRefill() throws java.io.IOException {
568
569      /* first: make room (if you can) */
570      if (zzStartRead > 0) {
571          System.arraycopy(zzBuffer, zzStartRead,
572                          zzBuffer, 0,
573                          zzEndRead-zzStartRead);
574
575          /* translate stored positions */
576          zzEndRead-= zzStartRead;
577          zzCurrentPos-= zzStartRead;
578          zzMarkedPos-= zzStartRead;
579          zzStartRead = 0;
580      }
581
582      /* is the buffer big enough? */
583      if (zzCurrentPos >= zzBuffer.length) {
584          /* if not: blow it up */
585          char newBuffer[] = new char[zzCurrentPos*2];
586          System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
587          zzBuffer = newBuffer;
588      }
589
590      /* finally: fill the buffer with new input */
591      int numRead = zzReader.read(zzBuffer, zzEndRead,
592                                  zzBuffer.length-zzEndRead);
593
594      if (numRead > 0) {
595          zzEndRead+= numRead;
596          return false;
597      }
598      // unlikely but not impossible: read 0 characters, but not at end of stream
599      if (numRead == 0) {
600          int c = zzReader.read();
601          if (c == -1) {
602              return true;
603          } else {
604              zzBuffer[zzEndRead++] = (char) c;
605              return false;
606          }
607      }
608
609      // numRead < 0
610      return true;
611  }
612
613
614  /**
615   * Closes the input stream.
616   */
617  public final void yclose() throws java.io.IOException {
618      zzAtEOF = true;           /* indicate end of file */
619      zzEndRead = zzStartRead; /* invalidate buffer */
620
621      if (zzReader != null)
622          zzReader.close();
623  }
624
625
626  /**
627   * Resets the scanner to read from a new input stream.
```



```
628     * Does not close the old reader.  
629     *  
630     * All internal variables are reset, the old input stream  
631     * <b>cannot</b> be reused (internal buffer is discarded and lost).  
632     * Lexical state is set to <tt>ZZ_INITIAL</tt>.   
633     *  
634     * @param reader the new input stream  
635     */  
636     public final void yyreset(java.io.Reader reader) {  
637         zzReader = reader;  
638         zzAtBOL = true;  
639         zzAtEOF = false;  
640         zzEOFDone = false;  
641         zzEndRead = zzStartRead = 0;  
642         zzCurrentPos = zzMarkedPos = 0;  
643         yyline = yychar = yycolumn = 0;  
644         zzLexicalState = YYINITIAL;  
645     }  
646  
647  
648     /**  
649      * Returns the current lexical state.  
650      */  
651     public final int yystate() {  
652         return zzLexicalState;  
653     }  
654  
655  
656     /**  
657      * Enters a new lexical state  
658      *  
659      * @param newState the new lexical state  
660      */
```

```
661     public final void yybegin(int newState) {  
662         zzLexicalState = newState;  
663     }  
664  
665  
666     /**  
667      * Returns the text matched by the current regular expression.  
668      */  
669     public final String yytext() {  
670         return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );  
671     }  
672  
673  
674     /**  
675      * Returns the character at position <tt>pos</tt> from the  
676      * matched text.  
677      *  
678      * It is equivalent to yytext().charAt(pos), but faster  
679      *  
680      * @param pos the position of the character to fetch.  
681      *          A value from 0 to yylength()-1.  
682      *  
683      * @return the character at position pos  
684      */  
685     public final char yycharat(int pos) {  
686         return zzBuffer[zzStartRead+pos];  
687     }  
688  
689  
690     /**  
691      * Returns the length of the matched text region.  
692      */  
693     public final int yylength() {
```



```
694     return zzMarkedPos-zzStartRead;
695 }
696
697
698 /**
699 * Reports an error that occurred while scanning.
700 *
701 * In a wellformed scanner (no or only correct usage of
702 * yypushback(int) and a match-all fallback rule) this method
703 * will only be called with things that "Can't Possibly Happen".
704 * If this method is called, something is seriously wrong
705 * (e.g. a JFlex bug producing a faulty scanner etc.).
706 *
707 * Usual syntax/scanner level error handling should be done
708 * in error fallback rules.
709 *
710 * @param errorCode the code of the errormessage to display
711 */
712 private void zzscanError(int errorCode) {
713     String message;
714     try {
715         message = ZZ_ERROR_MSG[errorCode];
716     }
717     catch (ArrayIndexOutOfBoundsException e) {
718         message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
719     }
720
721     throw new Error(message);
722 }
723
724
725 /**
726 * Pushes the specified amount of characters back into the input stream.
727
728 * They will be read again by then next call of the scanning method
729 *
730 * @param number the number of characters to be read again.
731 *                This number must not be greater than yylength()!
732 */
733 public void yypushback(int number) {
734     if (number > yylength())
735         zzScanError(ZZ_PUSHBACK_2BIG);
736
737     zzMarkedPos -= number;
738 }
739
740
741 /**
742 * Resumes scanning until the next regular expression is matched,
743 * the end of input is encountered or an I/O-Error occurs.
744 *
745 * @return the next token
746 * @exception java.io.IOException if any I/O-Error occurs
747 */
748 public Tokens yylex() throws java.io.IOException {
749     int zzInput;
750     int zzAction;
751
752     // cached fields:
753     int zzCurrentPosL;
754     int zzMarkedPosL;
755     int zzEndReadL = zzEndRead;
756     char [] zzBufferL = zzBuffer;
757     char [] zzCMapL = ZZ_CMAP;
758
759     int [] zzTransL = ZZ_TRANS;
```



```
760 int [] zzRowMapL = ZZ_ROWMAP;
761 int [] zzAttrL = ZZ_ATTRIBUTE;
762
763 while (true) {
764     zzMarkedPosL = zzMarkedPos;
765
766     zzAction = -1;
767
768     zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
769
770     zzState = ZZ_LEXSTATE[zzLexicalState];
771
772     zzForAction: {
773         while (true) {
774
775             if (zzCurrentPosL < zzEndReadL)
776                 zzInput = zzBufferL[zzCurrentPosL++];
777             else if (zzAtEOF) {
778                 zzInput = YYEOF;
779                 break zzForAction;
780             }
781             else {
782                 // store back cached positions
783                 zzCurrentPos = zzCurrentPosL;
784                 zzMarkedPos = zzMarkedPosL;
785                 boolean eof = zzRefill();
786                 // get translated positions and possibly new buffer
787                 zzCurrentPosL = zzCurrentPos;
788                 zzMarkedPosL = zzMarkedPos;
789                 zzBufferL = zzBuffer;
790                 zzEndReadL = zzEndread;
791                 if (eof) {
792
793                     zzInput = YYEOF;
794                     break zzForAction;
795                 }
796                 else {
797                     zzInput = zzBufferL[zzCurrentPosL++];
798                 }
799             }
800             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
801             if (zzNext == -1) break zzForAction;
802             zzState = zzNext;
803
804             int zzAttributes = zzAttrL[zzState];
805             if ( (zzAttributes & 1) == 1 ) {
806                 zzAction = zzState;
807                 zzMarkedPosL = zzCurrentPosL;
808                 if ( (zzAttributes & 8) == 8 ) break zzForAction;
809             }
810
811         }
812     }
813
814     // store back cached position
815     zzMarkedPos = zzMarkedPosL;
816
817     switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
818         case 20:
819             { return parentDer; }
820         case 77: break;
821         case 21:
822             { return llaveDer; }
823         case 78: break;
824     }
825 }
```



# MANUAL DE PRÁCTICAS



```
826         case 10:
827             { return operadorModulo;
828         }
829         case 79: break;
830         case 32:
831             { lexeme=yytext(); return Identificador;
832         }
833         case 80: break;
834         case 22:
835             { return pesos;
836         }
837         case 81: break;
838         case 54:
839             { return cadena;
840         }
841         case 82: break;
842         case 69:
843             { return detener;
844         }
845         case 83: break;
846         case 31:
847             { return gato;
848         }
849         case 84: break;
850         case 3:
851             { return numero;
852         }
853         case 85: break;
854         case 2:
855             { return guionBajo;
856         }
857         case 86: break;
858         case 67:
```

```
859             { return arreglo;
860         }
861         case 87: break;
862         case 11:
863             { return operadorPotencia;
864         }
865         case 88: break;
866         case 72:
867             { return importar;
868         }
869         case 89: break;
870         case 34:
871             { lexeme=yytext(); return Reservadas;
872         }
873         case 90: break;
874         case 75:
875             { return funcionIntroducirDatos;
876         }
877         case 91: break;
878         case 36:
879             { return incremento;
880         }
881         case 92: break;
882         case 24:
883             { return corchetIzq;
884         }
885         case 93: break;
886         case 17:
887             { return operadorNo;
888         }
889         case 94: break;
890         case 4:
891             { /*Ignore*/
```



```
892 }  
893 case 95: break;  
894 case 8:  
895 { return operadorResta;  
896 }  
897 case 96: break;  
898 case 60:  
899 { return funcionLeer;  
900 }  
901 case 97: break;  
902 case 46:  
903 { return operadorPositivo;  
904 }  
905 case 98: break;  
906 case 14:  
907 { return menorQue;  
908 }  
909 case 99: break;  
910 case 5:  
911 { return letrasMin;  
912 }  
913 case 100: break;  
914 case 70:  
915 { return cicloWhile; }  
916 }  
917 case 101: break;  
918 case 53:  
919 { return valorBooleano;  
920 }  
921 case 102: break;  
922 case 65:  
923 { return cicloFor;  
924 }
```

```
925 case 103: break;  
926 case 42:  
927 { return inicioTexto;  
928 }  
929 case 104: break;  
930 case 66:  
931 { return clase;  
932 }  
933 case 105: break;  
934 case 1:  
935 { return ERROR;  
936 }  
937 case 106: break;  
938 case 19:  
939 { return dosPuntos;  
940 }  
941 case 107: break;  
942 case 28:  
943 { return comillaSimple;  
944 }  
945 case 108: break;  
946 case 76:  
947 { return predeterminado;  
948 }  
949 case 109: break;  
950 case 50:  
951 { return concatenar;  
952 }  
953 case 110: break;  
954 case 56:  
955 { return condicionalElse;  
956 }  
957 case 111: break;
```



```
958     case 39:  
959     { return asignacion;  
960     }  
961     case 112: break;  
962     case 44:  
963     { return saltoLinea;  
964     }  
965     case 113: break;  
966     case 63:  
967     { return cicloDo;  
968     }  
969     case 114: break;  
970     case 49:  
971     { return inicioComentarioMult;  
972     }  
973     case 115: break;  
974     case 37:  
975     { return decremento;  
976     }  
977     case 116: break;  
978     case 13:  
979     { return mayorQue;  
980     }  
981     case 117: break;  
982     case 64:  
983     { return operadorRaiz;  
984     }  
985     case 118: break;  
986     case 35:  
987     { return condicionalIf;  
988     }  
989     case 119: break;  
990     case 57:
```

```
991     { return funcionCase;  
992     }  
993     case 120: break;  
994     case 16:  
995     { return operadorO;  
996     }  
997     case 121: break;  
998     case 43:  
999     { return finalComentarioMult;  
1000    }  
1001    case 122: break;  
1002    case 74:  
1003    { return constante;  
1004    }  
1005    case 123: break;  
1006    case 25:  
1007    { return corchetDer;  
1008    }  
1009    case 124: break;  
1010    case 58:  
1011    { return valorFlotante;  
1012    }  
1013    case 125: break;  
1014    case 51:  
1015    { return comentarioLinea;  
1016    }  
1017    case 126: break;  
1018    case 47:  
1019    { return operadorNegativo;  
1020    }  
1021    case 127: break;  
1022    case 26:  
1023    { return puntoAcceso;
```



```
1024     }
1025     case 128: break;
1026     case 18:
1027     {
1028         return parentIzq;
1029     }
1030     case 129: break;
1031     case 23:
1032     {
1033         return llaveIzq;
1034     }
1035     case 130: break;
1036     case 40:
1037     {
1038         return finalTexto;
1039     }
1040     case 131: break;
1041     case 33:
1042     {
1043         lexeme=yytext(); return Numero;
1044     }
1045     case 132: break;
1046     case 48:
1047     {
1048         return menorIgual;
1049     }
1050     case 133: break;
1051     case 27:
1052     {
1053         return comillaDoble;
1054     }
1055     case 134: break;
1056     case 12:
1057     {
1058         return operadorSuma;
1059     }
1060     case 135: break;
1061     case 136: break;
1062     case 45:
1063     {
1064         return mayorIgual;
1065     }
1066     case 137: break;
1067     case 62:
1068     {
1069         return funcionSalirCase;
1070     }
1071     case 138: break;
1072     case 68:
1073     {
1074         return funcion;
1075     }
1076     case 139: break;
1077     case 29:
1078     {
1079         return diagonalInvertido;
1080     }
1081     case 140: break;
1082     case 30:
1083     {
1084         return signoAdmiracionAbre;
1085     }
1086     case 141: break;
1087     case 71:
1088     {
1089         return variable;
1090     }
1091     case 142: break;
1092     case 73:
1093     {
1094         return imprimir;
1095     }
1096     case 143: break;
1097     case 15:
1098     {
1099         return operadorY;
1100     }
1101     case 144: break;
```

```
1102     case 145: break;
1103     case 146:
1104     {
1105         return operadorX;
1106     }
1107     case 147: break;
1108     case 148:
1109     {
1110         return operadorMultiplicacion;
1111     }
1112     case 149: break;
1113     case 150:
1114     {
1115         return operadorDivision;
1116     }
1117     case 151: break;
1118     case 152:
1119     {
1120         return operadorModulo;
1121     }
1122     case 153: break;
1123     case 154:
1124     {
1125         return operadorPotencia;
1126     }
1127     case 155: break;
1128     case 156:
1129     {
1130         return operadorComplemento;
1131     }
1132     case 157: break;
1133     case 158:
1134     {
1135         return operadorNegacion;
1136     }
1137     case 159: break;
1138     case 160:
1139     {
1140         return operadorAsignacion;
1141     }
1142     case 161: break;
1143     case 162:
1144     {
1145         return operadorRelacional;
1146     }
1147     case 163: break;
1148     case 164:
1149     {
1150         return operadorLogico;
1151     }
1152     case 165: break;
1153     case 166:
1154     {
1155         return operadorAritmetico;
1156     }
1157     case 167: break;
1158     case 168:
1159     {
1160         return operadorComparacion;
1161     }
1162     case 169: break;
1163     case 170:
1164     {
1165         return operadorRelacional;
1166     }
1167     case 171: break;
1168     case 172:
1169     {
1170         return operadorLogico;
1171     }
1172     case 173: break;
1173     case 174:
1174     {
1175         return operadorAritmetico;
1176     }
1177     case 178: break;
1178     case 179:
1179     {
1180         return operadorComparacion;
1181     }
1182     case 180: break;
1183     case 181:
1184     {
1185         return operadorRelacional;
1186     }
1187     case 188: break;
1188     case 189:
1189     {
1190         return operadorLogico;
1191     }
1192     case 190: break;
1193     case 191:
1194     {
1195         return operadorAritmetico;
1196     }
1197     case 198: break;
1198     case 199:
1199     {
1200         return operadorComparacion;
1201     }
1202     case 200: break;
1203     case 201:
1204     {
1205         return operadorRelacional;
1206     }
1207     case 208: break;
1208     case 209:
1209     {
1210         return operadorLogico;
1211     }
1212     case 209: break;
1213     case 210:
1214     {
1215         return operadorAritmetico;
1216     }
1217     case 218: break;
1218     case 219:
1219     {
1220         return operadorComparacion;
1221     }
1222     case 220: break;
1223     case 221:
1224     {
1225         return operadorRelacional;
1226     }
1227     case 228: break;
1228     case 229:
1229     {
1230         return operadorLogico;
1231     }
1232     case 229: break;
1233     case 230:
1234     {
1235         return operadorAritmetico;
1236     }
1237     case 238: break;
1238     case 239:
1239     {
1240         return operadorComparacion;
1241     }
1242     case 240: break;
1243     case 241:
1244     {
1245         return operadorRelacional;
1246     }
1247     case 248: break;
1248     case 249:
1249     {
1250         return operadorLogico;
1251     }
1252     case 249: break;
1253     case 250:
1254     {
1255         return operadorAritmetico;
1256     }
1257     case 258: break;
1258     case 259:
1259     {
1260         return operadorComparacion;
1261     }
1262     case 260: break;
1263     case 261:
1264     {
1265         return operadorRelacional;
1266     }
1267     case 268: break;
1268     case 269:
1269     {
1270         return operadorLogico;
1271     }
1272     case 269: break;
1273     case 270:
1274     {
1275         return operadorAritmetico;
1276     }
1277     case 278: break;
1278     case 279:
1279     {
1280         return operadorComparacion;
1281     }
1282     case 280: break;
1283     case 281:
1284     {
1285         return operadorRelacional;
1286     }
1287     case 288: break;
1288     case 289:
1289     {
1290         return operadorLogico;
1291     }
1292     case 289: break;
1293     case 290:
1294     {
1295         return operadorAritmetico;
1296     }
1297     case 298: break;
1298     case 299:
1299     {
1300         return operadorComparacion;
1301     }
1302     case 300: break;
1303     case 301:
1304     {
1305         return operadorRelacional;
1306     }
1307     case 308: break;
1308     case 309:
1309     {
1310         return operadorLogico;
1311     }
1312     case 309: break;
1313     case 310:
1314     {
1315         return operadorAritmetico;
1316     }
1317     case 318: break;
1318     case 319:
1319     {
1320         return operadorComparacion;
1321     }
1322     case 320: break;
1323     case 321:
1324     {
1325         return operadorRelacional;
1326     }
1327     case 328: break;
1328     case 329:
1329     {
1330         return operadorLogico;
1331     }
1332     case 329: break;
1333     case 330:
1334     {
1335         return operadorAritmetico;
1336     }
1337     case 338: break;
1338     case 339:
1339     {
1340         return operadorComparacion;
1341     }
1342     case 340: break;
1343     case 341:
1344     {
1345         return operadorRelacional;
1346     }
1347     case 348: break;
1348     case 349:
1349     {
1350         return operadorLogico;
1351     }
1352     case 349: break;
1353     case 350:
1354     {
1355         return operadorAritmetico;
1356     }
1357     case 358: break;
1358     case 359:
1359     {
1360         return operadorComparacion;
1361     }
1362     case 360: break;
1363     case 361:
1364     {
1365         return operadorRelacional;
1366     }
1367     case 368: break;
1368     case 369:
1369     {
1370         return operadorLogico;
1371     }
1372     case 369: break;
1373     case 370:
1374     {
1375         return operadorAritmetico;
1376     }
1377     case 378: break;
1378     case 379:
1379     {
1380         return operadorComparacion;
1381     }
1382     case 380: break;
1383     case 381:
1384     {
1385         return operadorRelacional;
1386     }
1387     case 388: break;
1388     case 389:
1389     {
1390         return operadorLogico;
1391     }
1392     case 389: break;
1393     case 390:
1394     {
1395         return operadorAritmetico;
1396     }
1397     case 398: break;
1398     case 399:
1399     {
1400         return operadorComparacion;
1401     }
1402     case 400: break;
1403     case 401:
1404     {
1405         return operadorRelacional;
1406     }
1407     case 408: break;
1408     case 409:
1409     {
1410         return operadorLogico;
1411     }
1412     case 409: break;
1413     case 410:
1414     {
1415         return operadorAritmetico;
1416     }
1417     case 418: break;
1418     case 419:
1419     {
1420         return operadorComparacion;
1421     }
1422     case 420: break;
1423     case 421:
1424     {
1425         return operadorRelacional;
1426     }
1427     case 428: break;
1428     case 429:
1429     {
1430         return operadorLogico;
1431     }
1432     case 429: break;
1433     case 430:
1434     {
1435         return operadorAritmetico;
1436     }
1437     case 438: break;
1438     case 439:
1439     {
1440         return operadorComparacion;
1441     }
1442     case 440: break;
1443     case 441:
1444     {
1445         return operadorRelacional;
1446     }
1447     case 448: break;
1448     case 449:
1449     {
1450         return operadorLogico;
1451     }
1452     case 449: break;
1453     case 450:
1454     {
1455         return operadorAritmetico;
1456     }
1457     case 458: break;
1458     case 459:
1459     {
1460         return operadorComparacion;
1461     }
1462     case 460: break;
1463     case 464:
1464     {
1465         return operadorRelacional;
1466     }
1467     case 468: break;
1468     case 469:
1469     {
1470         return operadorLogico;
1471     }
1472     case 469: break;
1473     case 470:
1474     {
1475         return operadorAritmetico;
1476     }
1477     case 478: break;
1478     case 479:
1479     {
1480         return operadorComparacion;
1481     }
1482     case 480: break;
1483     case 481:
1484     {
1485         return operadorRelacional;
1486     }
1487     case 488: break;
1488     case 489:
1489     {
1490         return operadorLogico;
1491     }
1492     case 489: break;
1493     case 490:
1494     {
1495         return operadorAritmetico;
1496     }
1497     case 498: break;
1498     case 499:
1499     {
1500         return operadorComparacion;
1501     }
1502     case 500: break;
1503     case 501:
1504     {
1505         return operadorRelacional;
1506     }
1507     case 508: break;
1508     case 509:
1509     {
1510         return operadorLogico;
1511     }
1512     case 509: break;
1513     case 510:
1514     {
1515         return operadorAritmetico;
1516     }
1517     case 518: break;
1518     case 519:
1519     {
1520         return operadorComparacion;
1521     }
1522     case 520: break;
1523     case 521:
1524     {
1525         return operadorRelacional;
1526     }
1527     case 528: break;
1528     case 529:
1529     {
1530         return operadorLogico;
1531     }
1532     case 529: break;
1533     case 530:
1534     {
1535         return operadorAritmetico;
1536     }
1537     case 538: break;
1538     case 539:
1539     {
1540         return operadorComparacion;
1541     }
1542     case 540: break;
1543     case 541:
1544     {
1545         return operadorRelacional;
1546     }
1547     case 548: break;
1548     case 549:
1549     {
1550         return operadorLogico;
1551     }
1552     case 549: break;
1553     case 550:
1554     {
1555         return operadorAritmetico;
1556     }
1557     case 558: break;
1558     case 559:
1559     {
1560         return operadorComparacion;
1561     }
1562     case 560: break;
1563     case 564:
1564     {
1565         return operadorRelacional;
1566     }
1567     case 568: break;
1568     case 569:
1569     {
1570         return operadorLogico;
1571     }
1572     case 569: break;
1573     case 570:
1574     {
1575         return operadorAritmetico;
1576     }
1577     case 578: break;
1578     case 579:
1579     {
1580         return operadorComparacion;
1581     }
1582     case 580: break;
1583     case 581:
1584     {
1585         return operadorRelacional;
1586     }
1587     case 588: break;
1588     case 589:
1589     {
1590         return operadorLogico;
1591     }
1592     case 589: break;
1593     case 590:
1594     {
1595         return operadorAritmetico;
1596     }
1597     case 598: break;
1598     case 599:
1599     {
1600         return operadorComparacion;
1601     }
1602     case 600: break;
1603     case 601:
1604     {
1605         return operadorRelacional;
1606     }
1607     case 608: break;
1608     case 609:
1609     {
1610         return operadorLogico;
1611     }
1612     case 609: break;
1613     case 610:
1614     {
1615         return operadorAritmetico;
1616     }
1617     case 618: break;
1618     case 619:
1619     {
1620         return operadorComparacion;
1621     }
1622     case 620: break;
1623     case 621:
1624     {
1625         return operadorRelacional;
1626     }
1627     case 628: break;
1628     case 629:
1629     {
1630         return operadorLogico;
1631     }
1632     case 629: break;
1633     case 630:
1634     {
1635         return operadorAritmetico;
1636     }
1637     case 638: break;
1638     case 639:
1639     {
1640         return operadorComparacion;
1641     }
1642     case 640: break;
1643     case 641:
1644     {
1645         return operadorRelacional;
1646     }
1647     case 648: break;
1648     case 649:
1649     {
1650         return operadorLogico;
1651     }
1652     case 649: break;
1653     case 650:
1654     {
1655         return operadorAritmetico;
1656     }
1657     case 658: break;
1658     case 659:
1659     {
1660         return operadorComparacion;
1661     }
1662     case 660: break;
1663     case 664:
1664     {
1665         return operadorRelacional;
1666     }
1667     case 668: break;
1668     case 669:
1669     {
1670         return operadorLogico;
1671     }
1672     case 669: break;
1673     case 670:
1674     {
1675         return operadorAritmetico;
1676     }
1677     case 678: break;
1678     case 679:
1679     {
1680         return operadorComparacion;
1681     }
1682     case 680: break;
1683     case 681:
1684     {
1685         return operadorRelacional;
1686     }
1687     case 688: break;
1688     case 689:
1689     {
1690         return operadorLogico;
1691     }
1692     case 689: break;
1693     case 690:
1694     {
1695         return operadorAritmetico;
1696     }
1697     case 698: break;
1698     case 699:
1699     {
1700         return operadorComparacion;
1701     }
1702     case 700: break;
1703     case 701:
1704     {
1705         return operadorRelacional;
1706     }
1707     case 708: break;
1708     case 709:
1709     {
1710         return operadorLogico;
1711     }
1712     case 709: break;
1713     case 710:
1714     {
1715         return operadorAritmetico;
1716     }
1717     case 718: break;
1718     case 719:
1719     {
1720         return operadorComparacion;
1721     }
1722     case 720: break;
1723     case 721:
1724     {
1725         return operadorRelacional;
1726     }
1727     case 728: break;
1728     case 729:
1729     {
1730         return operadorLogico;
1731     }
1732     case 729: break;
1733     case 730:
1734     {
1735         return operadorAritmetico;
1736     }
1737     case 738: break;
1738     case 739:
1739     {
1740         return operadorComparacion;
1741     }
1742     case 740: break;
1743     case 741:
1744     {
1745         return operadorRelacional;
1746     }
1747     case 748: break;
1748     case 749:
1749     {
1750         return operadorLogico;
1751     }
1752     case 749: break;
1753     case 750:
1754     {
1755         return operadorAritmetico;
1756     }
1757     case 758: break;
1758     case 759:
1759     {
1760         return operadorComparacion;
1761     }
1762     case 760: break;
1763     case 764:
1764     {
1765         return operadorRelacional;
1766     }
1767     case 768: break;
1768     case 769:
1769     {
1770         return operadorLogico;
1771     }
1772     case 769: break;
1773     case 770:
1774     {
1775         return operadorAritmetico;
1776     }
1777     case 778: break;
1778     case 779:
1779     {
1780         return operadorComparacion;
1781     }
1782     case 780: break;
1783     case 781:
1784     {
1785         return operadorRelacional;
1786     }
1787     case 788: break;
1788     case 789:
1789     {
1790         return operadorLogico;
1791     }
1792     case 789: break;
1793     case 790:
1794     {
1795         return operadorAritmetico;
1796     }
1797     case 798: break;
1798     case 799:
1799     {
1800         return operadorComparacion;
1801     }
1802     case 800: break;
1803     case 801:
1804     {
1805         return operadorRelacional;
1806     }
1807     case 808: break;
1808     case 809:
1809     {
1810         return operadorLogico;
1811     }
1812     case 809: break;
1813     case 810:
1814     {
1815         return operadorAritmetico;
1816     }
1817     case 818: break;
1818     case 819:
1819     {
1820         return operadorComparacion;
1821     }
1822     case 820: break;
1823     case 821:
1824     {
1825         return operadorRelacional;
1826     }
1827     case 828: break;
1828     case 829:
1829     {
1830         return operadorLogico;
1831     }
1832     case 829: break;
1833     case 830:
1834     {
1835         return operadorAritmetico;
1836     }
1837     case 838: break;
1838     case 839:
1839     {
1840         return operadorComparacion;
1841     }
1842     case 840: break;
1843     case 841:
1844     {
1845         return operadorRelacional;
1846     }
1847     case 848: break;
1848     case 849:
1849     {
1850         return operadorLogico;
1851     }
1852     case 849: break;
1853     case 850:
1854     {
1855         return operadorAritmetico;
1856     }
1857     case 858: break;
1858     case 859:
1859     {
1860         return operadorComparacion;
1861     }
1862     case 860: break;
1863     case 864:
1864     {
1865         return operadorRelacional;
1866     }
1867     case 868: break;
1868     case 869:
1869     {
1870         return operadorLogico;
1871     }
1872     case 869: break;
1873     case 870:
1874     {
1875         return operadorAritmetico;
1876     }
1877     case 878: break;
1878     case 879:
1879     {
1880         return operadorComparacion;
1881     }
1882     case 880: break;
1883     case 884:
1884     {
1885         return operadorRelacional;
1886     }
1887     case 888: break;
1888     case 889:
1889     {
1890         return operadorLogico;
1891     }
1892     case 889: break;
1893     case 890:
1894     {
1895         return operadorAritmetico;
1896     }
1897     case 898: break;
1898     case 899:
1899     {
1900         return operadorComparacion;
1901     }
1902     case 1900: break;
1903     case 1901:
1904     {
1905         return operadorRelacional;
1906     }
1907     case 1908: break;
1908     case 1909:
1909     {
1910         return operadorLogico;
1911     }
1912     case 1911: break;
1913     case 1914:
1914     {
1915         return operadorAritmetico;
1916     }
1917     case 1918: break;
1918     case 1919:
1919     {
1920         return operadorComparacion;
1921     }
1922     case 1923: break;
1923     case 1924:
1924     {
1925         return operadorRelacional;
1926     }
1927     case 1928: break;
1928     case 1929:
1929     {
1930         return operadorLogico;
1931     }
1932     case 1931: break;
1933     case 1934:
1934     {
1935         return operadorAritmetico;
1936     }
1937     case 1938: break;
1938     case 1939:
1939     {
1940         return operadorComparacion;
1941     }
1942     case 1943: break;
1943     case 1944:
1944     {
1945         return operadorRelacional;
1946     }
1947     case 1948: break;
1948     case 1949:
1949     {
1950         return operadorLogico;
1951     }
1952     case 1951: break;
1953     case 1954:
1954     {
1955         return operadorAritmetico;
1956     }
1957     case 1958: break;
1958     case 1959:
1959     {
1960         return operadorComparacion;
1961     }
1962     case 1963: break;
1963     case 1964:
1964     {
1965         return operadorRelacional;
1966     }
1967     case 1968: break;
1968     case 1969:
1969     {
1970         return operadorLogico;
1971     }
1972     case 1971: break;
1973     case 1974:
1974     {
1975         return operadorAritmetico;
1976     }
1977     case 1978: break;
1978     case 1979:
1979     {
1980         return operadorComparacion;
1981     }
1982     case 1983: break;
1983     case 1984:
1984     {
1985         return operadorRelacional;
1986     }
1987     case 1988: break;
1988     case 1989:
1989     {
1990         return operadorLogico;
1991     }
1992     case 1991: break;
1993     case 1994:
1994     {
1995         return operadorAritmetico;
1996     }
1997     case 1998: break;
1998     case 1999:
1999     {
2000         return operadorComparacion;
2001     }
2002     case 2003: break;
2003     case 2004:
2004     {
2005         return operadorRelacional;
2006     }
2007     case 2008: break;
2008     case 2009:
2009     {
2010         return operadorLogico;
2011     }
2012     case 2011: break;
2013     case 2014:
2014     {
2015         return operadorAritmetico;
2016     }
2017     case 2018: break;
2018     case 2019:
2019     {
2020         return operadorComparacion;
2021     }
2022     case 2023: break;
2023     case 2024:
2024     {
2025         return operadorRelacional;
2026     }
2027     case 2028: break;
2028     case 2029:
2029     {
2030         return operadorLogico;
2031     }
2032     case 2031: break;
2033     case 2034:
2034     {
2035         return operadorAritmetico;
2036     }
2037     case 2038: break;
2038     case 2039:
2039     {
2040         return operadorComparacion;
2041     }
2042     case 2043: break;
2043     case 2044:
2044     {
2045         return operadorRelacional;
2046     }
2047     case 2048: break;
2048     case 2049:
2049     {
2050         return operadorLogico;
2051     }
2052     case 2051: break;
2053     case 2054:
2054     {
2055         return operadorAritmetico;
2056     }
2057     case 2058: break;
2058     case 2059:
2059     {
2060         return operadorComparacion;
2061     }
2062     case 2063: break;
2063     case 2064:
2064     {
2065         return operadorRelacional;
2066     }
2067     case 2068: break;
2068     case 2069:
2069     {
2070         return operadorLogico;
2071     }
2072     case 2071: break;
2073     case 2074:
2074     {
2075         return operadorAritmetico;
2076     }
2077     case 2078: break;
2078     case 2079:
2079     {
2080         return operadorComparacion;
2081     }
2082     case 2083: break;
2083     case 2084:
2084     {
2085         return operadorRelacional;
2086     }
2087     case 2088: break;
2088     case 2089:
2089     {
2090         return operadorLogico;
2091     }
2092     case 2091: break;
2093     case 2094:
2094     {
2095         return operadorAritmetico;
2096     }
2097     case 2098: break;
2098     case 2099:
2099     {
2100         return operadorComparacion;
2101     }
2102     case 2103: break;
2103     case 2104:
2104     {
2105         return operadorRelacional;
2106     }
2107     case 2108: break;
2108     case 2109:
2109     {
2110         return operadorLogico;
2111     }
2112     case 2111: break;
2113     case 2114:
2114     {
2115         return operadorAritmetico;
2116     }
2117     case 2118: break;
2118     case 2119:
2119     {
2120         return operadorComparacion;
2121     }
2122     case 2123: break;
2123     case 2124:
2124     {
2125         return operadorRelacional;
2126     }
2127     case 2128: break;
2128     case 2129:
2129     {
2130         return operadorLogico;
2131     }
2132     case 2131: break;
2133     case 2134:
2134     {
2135         return operadorAritmetico;
2136     }
2137     case 2138: break;
2138     case 2139:
2139     {
2140         return operadorComparacion;
2141     }
2142     case 2143: break;
2143     case 2144:
2144     {
2145         return operadorRelacional;
2146     }
2147     case 2148: break;
2148     case 2149:
2149     {
2150         return operadorLogico;
2151     }
2152     case 2151: break;
2153     case 2154:
2154     {
2155         return operadorAritmetico;
2156     }
2157     case 2158: break;
2158     case 2159:
2159     {
2160         return operadorComparacion;
2161     }
2162     case 2163: break;
2163     case 2164:
2164     {
2165         return operadorRelacional;
2166     }
2167     case 2168: break;
2168     case 2169:
2169     {
2170         return operadorLogico;
2171     }
2172     case 2171: break;
2173     case 2174:
2174     {
2175         return operadorAritmetico;
2176     }
2177     case 2178: break;
2178     case 2179:
2179     {
2180         return operadorComparacion;
2181     }
2182     case 2183: break;
2183     case 2184:
2184     {
2185         return operadorRelacional;
2186     }
2187     case 2188: break;
2188     case 2189:
2189     {
2190         return
```



```
1090      case 59:
1091      { return carácter;
1092      }
1093      case 145: break;
1094      case 38:
1095      { return comparacionIgualdad;
1096      }
1097      case 146: break;
1098      case 41:
1099      { return pi;
1100      }
1101      case 147: break;
1102      case 6:
1103      { return operadorDivision;
1104      }
1105      case 148: break;
1106      case 9:
1107      { return operadorMultiplicacion;
1108      }
1109      case 149: break;
1110      case 61:
1111      { return funcionSwitch;
1112      }
1113      case 150: break;
1114      case 52:
1115      { return valorEntero;
1116      }
1117      case 151: break;
1118      case 55:
1119      { return finLinea;
1120      }
1121      case 152: break;
1122      default:
```

```
1123      if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
1124          zzAtEOF = true;
1125          return null;
1126      }
1127      else {
1128          zzScanError(ZZ_NO_MATCH);
1129      }
1130  }
1131 }
1132 }
1133
1134
1135 }
1136 }
```

## Archivo Tokens.java

El archivo Tokens.java es una pieza fundamental en la implementación de este analizador léxico, ya que define una enumeración que clasifica los diferentes tipos de tokens que el analizador puede identificar en el código fuente. Los tokens representan las unidades léxicas básicas del lenguaje, como palabras reservadas, operadores, identificadores, números y símbolos especiales, así como elementos estructurales como llaves, paréntesis y comentarios. Además, incluye un token genérico para errores, lo que permite gestionar y reportar entradas no reconocidas.

Este archivo actúa como una referencia central para el análisis y clasificación de los elementos léxicos, estableciendo una relación clara entre el código fuente y los componentes del lenguaje que se está interpretando o compilando. Esta enumeración facilita la implementación de reglas de análisis, mejora la legibilidad del código y contribuye a mantener una estructura ordenada y extensible para futuras actualizaciones o ampliaciones del lenguaje soportado por el analizador.



## Código Realizado:

```
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 codigo;  
7  
8  
9 public enum Tokens {  
10    Linea,  
11    Reservadas,  
12    operadorSuma,  
13    operadorResta,  
14    operadorMultiplicacion,  
15    operadorDivision,  
16    operadorModulo,  
17    operadorPotencia,  
18    operadorRaiz,  
19    asignacion,  
20    incremento,  
21    decremento,  
22    operadorPositivo,  
23    operadorNegativo,  
24    comparacionIgualdad,  
25    mayorQue,  
26    menorQue,  
27    mayorIgual,  
28    menorIgual,  
29    operadorY,  
30    operadorO,  
31    operadorNo,  
32    letrasMin,  
33    letrasMay,  
34    numero,  
35    inicioComentarioMult,  
36    finalComentarioMult,  
37    comentarioLinea,  
38    valorEntero,  
39    valorFlotante,  
40    valorBooleano,  
41    cadena,  
42    caracter,  
43    finLinea,  
44  
45    inicioTexto,  
46    finalTexto,  
47    parentIzq,  
48    parentDer,  
49    llaveIzq,  
50    llaveDer,  
51    corcheteIzq,  
52    corcheteDer,  
53    concatenar,  
54    arreglo,  
55  
56    puntoDecimal,  
57    dosPuntos,  
58    espacio,  
59    tabulacion,  
60    comillaDoble,  
61    comillaSimple,  
62    coma,  
63    guionBajo,  
64    guionMedio,  
65    diagonalInvertido,  
66    diagonal ,
```



```
67 signoAdmirationAbre,
68 signoAdmirationCierra,
69 gato,
70 pesos,
71 condicionalIf,
72 condicionalElse,
73 cicloFor,
74 cicloWhile,
75 cicloDo,
76 funcionesSwitch,
77 funcionCase,
78 funcionesSalirCase,
79 predeterminado,
80 detener,
81 constante,
82 variable,
83 funcion,
84 clase,
85 imprimir,
86 importar,
87 funcionLeer,
88 funcionIntroducirDatos,
89 pi,
90 euler,
91 Numero,
92 Identificador,
93     ERROR
94 }
```

## Archivo Principal.java

El archivo Principal.java actúa como el punto de entrada para la ejecución del analizador léxico en Java. Este archivo combina la inicialización de la interfaz gráfica con la ejecución del proceso que genera el analizador léxico a partir de una especificación escrita en JFlex. La clase implementa un splash screen mediante un hilo separado, que sirve como una introducción visual antes de cargar la interfaz principal del sistema, representada por el componente LoginUI.

Este archivo es un coordinador del flujo de ejecución del programa. Pues gestiona la experiencia del usuario al proporcionar un inicio visualmente atractivo y funcional. Además también automatiza la generación del archivo del analizador léxico mediante la herramienta JFlex, asegurando que el componente encargado del análisis de tokens esté disponible y actualizado. Esto lo convierte en una pieza esencial para la interacción fluida entre el usuario y el sistema, además de garantizar la eficiencia técnica del proceso de análisis léxico.

## Código Realizado:

```
1  package codigo;
2
3
4  import java.io.File;
5  import java.util.logging.Level;
6  import java.util.logging.Logger;
7
8  public class Principal {
9
10    public static void main(String[] args) {
11
12        // Implementación del Runnable para el Splash y la pantalla principal
13        Runnable mRun = () -> {
14            // JFrame que funge como splash
15            ModernSplash mSplash = new ModernSplash();
16            mSplash.setVisible(true);
17
18            try {
19                Thread.sleep(10000); // 5000 milisegundos equivale a 5 segundos.
20            } catch (InterruptedException ex) {
21                Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
22            }
23
24            mSplash.dispose();
25
26            // JFrame que funge como pantalla principal
27            LoginUI mFrmPrincipal = new LoginUI(); // Renombrado de FrmBienvenida
28            mFrmPrincipal.setVisible(true);
29        };
29
30
31        // Creación e inicio del hilo para el Splash
32        Thread mHiloSplash = new Thread(mRun);
33        mHiloSplash.start();
33
34
35        // Ejecución del método para generar el Lexer
36        String ruta = "C:/Users/jesus/Downloads/AnalizadorLexico/src/codigo/Lexer.flex";
37        generarLexer(ruta);
38    }
39
40    public static void generarLexer(String ruta) {
41        File archivo = new File(ruta);
42        JFlex.Main.generate(archivo);
43    }
44}
```

### Archivo ModernSplash.java (Splash del proyecto)

El splash de este proyecto, llamado ModernSplash, es una pantalla inicial diseñada para ofrecer una experiencia visual atractiva mientras se cargan los recursos necesarios del sistema prácticamente es una ventana inicial moderna e interactiva que combina varias animaciones y efectos visuales para ofrecer una experiencia de bienvenida atractiva al usuario.. Este componente destaca por su diseño moderno, que incluye un fondo degradado en tonos azules, un título animado que realiza un efecto de caída y rebote simulado con física, y un ícono que flota suavemente gracias a una animación sinusoidal. Además, incorpora una barra de progreso estilizada con colores dinámicos y mensajes de estado que informan al usuario sobre el avance de la carga, como "Preparando recursos..." y "Cargando módulos...".

La finalidad principal de este splash es mejorar la interacción inicial con el usuario, proporcionando un entorno visual que no solo entretiene, sino que también comunica claramente el progreso mientras se inicializan los módulos del proyecto. Este tipo de elemento no solo ayuda a gestionar la percepción del tiempo de carga, sino que también refuerza la identidad visual del proyecto, haciendo que la primera impresión sea profesional y agradable.



## Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class ModernSplash extends JFrame {
8     private JProgressBar progressBar;
9     private JLabel titleLabel, loadingLabel, iconLabel;
10    private Timer progressTimer, fadeTimer, bounceTimer, colorTimer, floatTimer;
11    private float opacity = 0.0f;
12    private Color[] gradientColors = {new Color(76, 175, 80)};
13    private int gradientIndex = 0;
14
15    // Variables para animación del título
16    private int titleYPosition = 50; // Posición inicial
17    private int targetY = 170; // Posición final
18    private double velocity = 0; // Velocidad inicial
19    private double gravity = 1.2; // Simula la gravedad
20    private int bouncesRemaining = 3; // Número de rebotes
21    private double dampingFactor = 0.6; // Factor de amortiguación
22
23    // Variables para animación flotante del ícono
24    private int iconBaseY = 30; // Posición base del ícono
25    private int floatAmplitude = 10; // Amplitud de la flotación
26    private double floatAngle = 0; // Ángulo para la animación sinusoidal
27
28    public ModernSplash() {
29        initComponents();
30    }
31
32    private void initComponents() {
33        // Configuración principal del JFrame
34        setUndecorated(true);
35        setSize(400, 300);
36        setOpacity(0.0f);
37        setLocationRelativeTo(null);
38        setLayout(null);
39
40        // Fondo personalizado
41        JPanel backgroundPanel = new JPanel() {
42            @Override
43            protected void paintComponent(Graphics g) {
44                super.paintComponent(g);
45                Graphics2D g2d = (Graphics2D) g;
46                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
47
48                // Degrado del fondo
49                GradientPaint gradient = new GradientPaint(0, 0, new Color(30, 136, 229), getWidth(), getHeight(), new Color(63, 81, 181));
50                g2d.setPaint(gradient);
51                g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
52            }
53        };
54        backgroundPanel.setBounds(0, 0, 400, 300);
55        backgroundPanel.setLayout(null);
56        add(backgroundPanel);
57
58        // Ícono
59        iconLabel = new JLabel(new ImageIcon("C:\\Users\\jesus\\Downloads\\AnalizadorLexico\\src\\codigo\\icono.png"));
60        iconLabel.setBounds(125, iconBaseY, 150, 150);
61        iconLabel.setHorizontalTextPosition(SwingConstants.CENTER);
62        backgroundPanel.add(iconLabel);
63    }
}
```



```
64     // Título con animación
65     titleLabel = new JLabel("Token-Visor");
66     titleLabel.setBounds(50, titleYPosition, 300, 40);
67     titleLabel.setFont(new Font("Roboto", Font.BOLD, 24));
68     titleLabel.setForeground(Color.WHITE);
69     titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
70     backgroundPanel.add(titleLabel);
71
72     // Barra de progreso
73     progressBar = new JProgressBar() {
74         @Override
75         protected void paintComponent(Graphics g) {
76             super.paintComponent(g);
77             Graphics2D g2d = (Graphics2D) g;
78             g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
79
80             int progressWidth = (int) (getWidth() * (getValue() / 100.0));
81             g2d.setColor(gradientColors[gradientIndex]); // Color dinámico
82             g2d.fillRoundRect(0, 0, progressWidth, getHeight(), 10, 10);
83         }
84     };
85     progressBar.setBounds(50, 230, 300, 10);
86     progressBar.setValue(0);
87     backgroundPanel.add(progressBar);
88
89     // Texto de carga
90     loadingLabel = new JLabel("Iniciando...");
91     loadingLabel.setBounds(50, 250, 300, 20);
92     loadingLabel.setFont(new Font("Roboto", Font.PLAIN, 14));
93     loadingLabel.setForeground(Color.WHITE);
94     loadingLabel.setHorizontalAlignment(SwingConstants.CENTER);
95     backgroundPanel.add(loadingLabel);
96
97     // Iniciar animaciones
98     startAnimations();
99 }
100
101 private void startAnimations() {
102     // Fade-in de la ventana
103     fadeTimer = new Timer(30, (ActionEvent e) -> {
104         opacity += 0.02f;
105         if (opacity >= 1.0f) {
106             setOpacity(1.0f);
107             fadeTimer.stop();
108             startFloatAnimation(); // Inicia la animación flotante del ícono
109             startBounceAnimation(); // Inicia la animación del título
110         } else {
111             setOpacity(opacity);
112         }
113     });
114     fadeTimer.start();
115
116     // Cambio de color dinámico para la barra de progreso
117     colorTimer = new Timer(2000, (ActionEvent e) -> {
118         gradientIndex = (gradientIndex + 1) % gradientColors.length;
119         progressBar.repaint();
120     });
121     colorTimer.start();
122 }
123
124 private void startFloatAnimation() {
125     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez
126         // Animación sinusoidal para "flotar"
127         floatAngle += 0.05f; // Incremento más pequeño para ralentizar el movimiento
128         int offsety = (int) (Math.sin(floatAngle) * floatAmplitude);
129         iconLabel.setBounds(125, iconBaseY + offsety, 150, 150);
130     });
131 }
```



```
130     });
131     floatTimer.start();
132 }
133
134
135     private void startBounceAnimation() {
136         bounceTimer = new Timer(20, (ActionEvent e) -> {
137             velocity += gravity; // Incrementa velocidad por efecto de "gravedad"
138             titleYPosition += velocity; // Actualiza posición del título
139
140             // Detecta colisión con el suelo virtual
141             if (titleYPosition >= targetY) {
142                 titleYPosition = targetY; // Ajusta la posición al suelo
143                 velocity = -velocity * dampingFactor; // Rebote con amortiguación
144                 bouncesRemaining--;
145
146                 // Si no quedan más rebotes, detiene la animación
147                 if (bouncesRemaining == 0) {
148                     bounceTimer.stop();
149                     titleYPosition = targetY; // Asegura posición final
150                     startProgressBar(); // Inicia la barra de progreso
151                 }
152             }
153
154             // Actualiza la posición del título en el componente
155             titleLabel.setBounds(50, titleYPosition, 300, 40);
156         });
157
158         bounceTimer.start();
159     }
160
161     private void startProgressBar() {
162         progressTimer = new Timer(50, (ActionEvent e) -> {
163             int value = progressBar.getValue();
164             if (value < 100) {
165                 progressBar.setValue(value + 1);
166
167                 // Cambiar texto dinámicamente
168                 if (value < 30) {
169                     loadingLabel.setText("Preparando recursos...");
170                 } else if (value < 60) {
171                     loadingLabel.setText("Cargando módulos...");
172                 } else if (value < 90) {
173                     loadingLabel.setText("Finalizando configuración...");
174                 } else {
175                     loadingLabel.setText("¡Listo para comenzar!");
176                 }
177             } else {
178                 progressTimer.stop();
179                 startFadeout();
180             }
181         });
182         progressTimer.start();
183     }
184
185     private void startFadeout() {
186         fadeTimer = new Timer(30, (ActionEvent e) -> {
187             opacity -= 0.02f;
188             if (opacity <= 0.0f) {
189                 setOpacity(0.0f);
190                 fadeTimer.stop();
191                 dispose(); // Cierra la ventana
192             } else {
193                 setOpacity(opacity);
194             }
195         });
196         fadeTimer.start();
197     }
198
199     public static void main(String[] args) {
200         EventQueue.invokeLater(() -> {
201             ModernSplash splash = new ModernSplash();
202             splash.setVisible(true);
203         });
204     }
205 }
```



## Archivo LoginUI.java

El archivo LoginUI.java es una implementación de una interfaz gráfica de usuario (GUI) en Java diseñada para ofrecer una experiencia interactiva de inicio de sesión dentro del analizador léxico. Utilizando la biblioteca Swing, esta clase presenta una ventana dividida en dos paneles: un lado izquierdo visualmente atractivo con un ícono flotante animado y un mensaje de bienvenida, y un lado derecho funcional que contiene el formulario de inicio de sesión con campos para el usuario y la contraseña.

La importancia de este archivo radica en su combinación de funcionalidad y estética. Además de validar credenciales de acceso con una lógica sencilla, proporciona un diseño profesional que mejora la interacción del usuario mediante detalles como animaciones suaves y botones personalizados. Su estructura modular y enfoque en la usabilidad lo convierten en un componente clave del sistema, creando una primera impresión positiva y facilitando la transición hacia las funcionalidades principales del programa.

### Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class LoginUI extends JFrame {
8
9     private JTextField registerNameField;
10    private JPasswordField registerPasswordField;
11    private JPanel leftPanel;
12    private JLabel iconLabel;
13    private Timer floatTimer;
14    private int iconRageY = 100; // Posición base del ícono en el eje Y
15    private int floatAmplitude = 10; // Amplitud de la flotación
16    private double floatAngle = 0; // Ángulo para la animación sinusoidal
17
18    public LoginUI() {
19        initComponents();
20        startFloatAnimation(); // Inicia la animación flotante del ícono
21    }
22
23    private void initComponents() {
24        setTitle("Login UI");
25        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26        setSize(800, 500);
27        setLocationRelativeTo(null);
28       .setLayout(new GridLayout(1, 2));
29
30        // Panel izquierdo (Bienvenida)
31        leftPanel = new JPanel();
32        leftPanel.setBackground(new Color(0, 38, 95));
33        leftPanel.setLayout(null); // Posición absoluta para incluir el ícono
```



# MANUAL DE PRÁCTICAS



```
35     JLabel welcomeLabel = new JLabel("<html><center>Nos Alegra Verte De Nuevo!<br>Inicia sesión para continuar</center></html>");
36     welcomeLabel.setForeground(Color.WHITE);
37     welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));
38     welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
39     welcomeLabel.setBounds(50, 300, 300, 100);
40     leftPanel.add(welcomeLabel);
41
42     // ícono flotante
43     iconLabel = new JLabel(new ImageIcon("C:\\\\Users\\\\jesus\\\\Downloads\\\\AnalizadorLexico\\\\src\\\\codigo\\\\icono.png"));
44     iconLabel.setBounds(0, iconBaseY, 150, 150); // La posición X se ajustará dinámicamente
45     leftPanel.add(iconLabel);
46
47     add(leftPanel);
48
49     // Panel derecho (Formulario de registro)
50     JPanel rightPanel = new JPanel();
51     rightPanel.setBackground(Color.WHITE);
52     rightPanel.setLayout(new GridBagLayout());
53
54     GridBagConstraints gbc = new GridBagConstraints();
55     gbc.insets = new Insets(10, 10, 10, 10);
56     gbc.gridx = 0;
57     gbc.anchor = GridBagConstraints.CENTER;
58
59     JLabel createAccountLabel = new JLabel("Inicio De Sesión");
60     createAccountLabel.setFont(new Font("Arial", Font.BOLD, 24));
61     createAccountLabel.setForeground(Color.BLACK);
62     gbc.gridy = 0;
63     rightPanel.add(createAccountLabel, gbc);
64
65     registerNameField = new JTextField(20);
66     registerNameField.setFont(new Font("Arial", Font.PLAIN, 14));
67     registerNameField.setBorder(BorderFactory.createTitledBorder("Usuario"));
68
68     gbc.gridy = 1;
69     rightPanel.add(registerNameField, gbc);
70
71     registerPasswordField = new JPasswordField(20);
72     registerPasswordField.setFont(new Font("Arial", Font.PLAIN, 14));
73     registerPasswordField.setBorder(BorderFactory.createTitledBorder("Contraseña"));
74     gbc.gridy = 2;
75     rightPanel.add(registerPasswordField, gbc);
76
77     CustomButton signUpButton = new CustomButton("Ingresar");
78     signUpButton.setPreferredSize(new Dimension(200, 50));
79     signUpButton.addActionListener(e -> {
80         String name = registerNameField.getText();
81         String password = new String(registerPasswordField.getPassword());
82
83         // Validación simple
84         if (name.equals("jesus") && password.equals("jesus123")) {
85             JOptionPane.showMessageDialog(this, "Inicio de sesión exitoso");
86             this.dispose(); // Cierra la ventana actual
87             FrmPrincipal mainFrame = new FrmPrincipal();
88             mainFrame.setVisible(true); // Abre la ventana principal
89         } else {
90             JOptionPane.showMessageDialog(this, "Credenciales inválidas. Intenta de nuevo.", "Error", JOptionPane.ERROR_MESSAGE);
91         }
92     });
93     gbc.gridy = 3;
94     rightPanel.add(signUpButton, gbc);
95
96     add(rightPanel);
97 }
98
99 private void startFloatAnimation() {
100     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez
```



```
101     // Animación sinusoidal para "flotar"
102     floatAngle += 0.05; // Incremento más pequeño para ralentizar el movimiento
103     int offsetY = (int) (Math.sin(floatAngle) * floatAmplitude);
104
105     // Recalcula la posición horizontal centrada
106     int iconX = (leftPanel.getWidth() - iconLabel.getWidth()) / 2;
107
108     // Aplica la posición centrada y el desplazamiento vertical
109     iconLabel.setBounds(iconX, iconBaseY + offsetY, 150, 150);
110   });
111   floatTimer.start();
112 }
113
114 public static void main(String[] args) {
115   SwingUtilities.invokeLater(() -> {
116     new LoginUI().setVisible(true);
117   });
118 }
119
120
121 // Clase personalizada para el botón
122 class CustomButton extends JButton {
123   private boolean isHovered = false;
124
125   public CustomButton(String text) {
126     super(text);
127     setContentAreaFilled(false);
128     setFocusPainted(false);
129     setBorderPainted(false);
130    setFont(new Font("Arial", Font.BOLD, 16));
131     setForeground(Color.WHITE);
132     setCursor(new Cursor(Cursor.HAND_CURSOR));
133
134   addMouseListener(new java.awt.event.MouseAdapter() {
135     @Override
136     public void mouseEntered(java.awt.event.MouseEvent e) {
137       isHovered = true;
138       repaint();
139     }
140
141     @Override
142     public void mouseExited(java.awt.event.MouseEvent e) {
143       isHovered = false;
144       repaint();
145     }
146   });
147
148   @Override
149   protected void paintComponent(Graphics g) {
150     Graphics2D g2d = (Graphics2D) g.create();
151     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
152
153     Color topColor = isHovered ? new Color(0, 120, 215) : new Color(33, 97, 140);
154     Color bottomColor = isHovered ? new Color(0, 85, 170) : new Color(0, 51, 102);
155
156     GradientPaint gradient = new GradientPaint(0, 0, topColor, 0, getHeight(), bottomColor);
157     g2d.setPaint(gradient);
158     g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
159
160     g2d.setColor(new Color(0, 0, 50));
161     g2d.fillRoundRect(2, 2, getWidth() - 4, getHeight() - 4, 30, 30);
162
163     super.paintComponent(g2d);
164     g2d.dispose();
165   }
166 }
167
168   @Override
169   protected void paintBorder(Graphics g) {
170 }
171 }
```

## Archivo FrmPrincipal.java

El archivo FrmPrincipal.java representa la interfaz principal del analizador léxico, diseñada para simular un entorno de desarrollo integrado (IDE). Utilizando la biblioteca Swing, esta clase combina características interactivas y estéticas para ofrecer un espacio de trabajo funcional. Entre sus componentes destacan un área de entrada para escribir código, un panel de resultados donde se muestra el análisis realizado, una barra lateral que simula un explorador de archivos y una barra de estado que comunica el estado actual del programa.

### Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.io.*;
6 import java.util.logging.*;
7 import javax.swing.*;
8
9 public class FrmPrincipal extends javax.swing.JFrame {
10
11     public FrmPrincipal() {
12         initComponents();
13         this.setLocationRelativeTo(null);
14         this.setTitle("Analizador Léxico - IDE Simulado");
15     }
16
17     private void initComponents() {
18         // Panel principal
19         mainPanel = new JPanel(new BorderLayout());
20         mainPanel.setBackground(new Color(30, 30, 30)); // Fondo oscuro principal
21
22         // Barra lateral simulada (como árbol de archivos)
23         JPanel sidebar = new JPanel();
24         sidebar.setPreferredSize(new Dimension(150, 0));
25         sidebar.setBackground(new Color(40, 40, 40));
26
27         JLabel lblExplorer = new JLabel("Explorador");
28         lblExplorer.setForeground(new Color(200, 200, 200));
29         lblExplorer.setFont(new Font("Consolas", Font.BOLD, 14));
30         lblExplorer.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
31
32         JList<String> fileTree = new JList<>(new String[]{"archivo.txt", "output.log", "README.md"});
33         fileTree.setBackground(new Color(50, 50, 50));
```



```
34     fileTree.setForeground(new Color(220, 220, 220));
35     fileTree.setFont(new Font("Consolas", Font.PLAIN, 14));
36     fileTree.setSelectionBackground(new Color(70, 130, 180));
37     fileTree.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
38
39     sidebar.setLayout(new BorderLayout());
40     sidebar.add(lblExplorer, BorderLayout.NORTH);
41     sidebar.add(fileTree, BorderLayout.CENTER);
42
43     // Barra de pestañas simulada
44     tabBar = new JPanel(new FlowLayout(FlowLayout.LEFT, 5, 5));
45     tabBar.setBackground(new Color(45, 45, 48));
46     tabBar.setPreferredSize(new Dimension(0, 35));
47
48     tabl = new JLabel("archivo.txt");
49     tabl.setForeground(new Color(220, 220, 220));
50     tabl.setFont(new Font("Consolas", Font.BOLD, 14));
51     tabl.setBorder(BorderFactory.createCompoundBorder(
52         BorderFactory.createMatteBorder(0, 0, 2, 0, new Color(97, 218, 251)),
53         BorderFactory.createEmptyBorder(0, 10, 0, 10)));
54
55     tabBar.add(tabl);
56
57     // Panel para entrada y números de linea
58     JPanel editorPanel = new JPanel(new BorderLayout());
59     editorPanel.setBackground(new Color(28, 28, 28));
60
61     // Números de linea
62     lineNumberArea = new JTextArea("1\n2\n3\n4\n5\n");
63     lineNumberArea.setFont(new Font("Consolas", Font.PLAIN, 16));
64     lineNumberArea.setBackground(new Color(35, 35, 35));
65     lineNumberArea.setForeground(new Color(150, 150, 150));
66     lineNumberArea.setEditable(false);
67
67     lineNumberArea.setMargin(new Insets(5, 5, 5, 5));
68
69     // Área de entrada de código
70     txtEntrada = new JTextArea();
71     txtEntrada.setFont(new Font("Consolas", Font.PLAIN, 16));
72     txtEntrada.setBackground(new Color(40, 44, 52));
73     txtEntrada.setForeground(new Color(200, 200, 200));
74     txtEntrada.setCaretColor(new Color(97, 218, 251));
75     txtEntrada.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
76     txtEntrada.setText("// Escribe tu código aquí...");
77     txtEntrada.setLineWrap(true);
78     txtEntrada.setWrapStyleWord(true);
79
80     JScrollPane scrollEntrada = new JScrollPane(txtEntrada);
81     scrollEntrada.setRowHeaderView(lineNumberArea);
82     scrollEntrada.setBorder(BorderFactory.createLineBorder(new Color(60, 63, 65)));
83
84     editorPanel.add(scrollEntrada, BorderLayout.CENTER);
85
86     // Área de resultados
87     txtResultado = new JTextArea();
88     txtResultado.setFont(new Font("Consolas", Font.PLAIN, 16));
89     txtResultado.setBackground(new Color(35, 35, 35));
90     txtResultado.setForeground(new Color(200, 200, 200));
91     txtResultado.setCaretColor(new Color(97, 218, 251));
92     txtResultado.setEditable(false);
93     txtResultado.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
94     txtResultado.setLineWrap(true);
95     txtResultado.setWrapStyleWord(true);
96     txtResultado.setText("// Los resultados del análisis aparecerán aquí...");
97
98     JScrollPane scrollResultado = new JScrollPane(txtResultado);
99     scrollResultado.setBorder(BorderFactory.createLineBorder(new Color(60, 63, 65)));
```



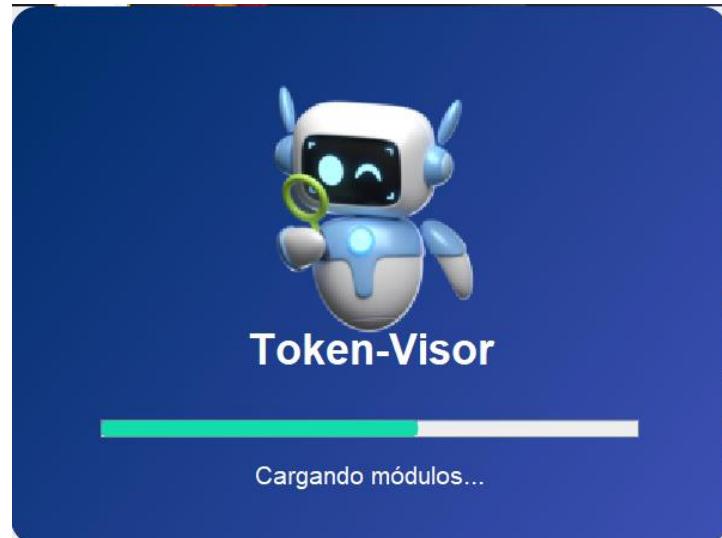
```
100
101     // Botón analizar
102     btnAnalizar = new JButton("Analizar Código");
103     btnAnalizar.setFont(new Font("Consolas", Font.BOLD, 16));
104     btnAnalizar.setBackground(new Color(97, 218, 251));
105     btnAnalizar.setForeground(new Color(28, 28, 28));
106     btnAnalizar.setFocusPainted(false);
107     btnAnalizar.setBorder(BorderFactory.createCompoundBorder(
108         BorderFactory.createLineBorder(new Color(60, 63, 65)),
109         BorderFactory.createEmptyBorder(10, 20, 10, 20)));
110
111     btnAnalizar.addActionListener(evt -> {
112         btnAnalizar.setText("Cargando...");
113         btnAnalizar.setEnabled(false);
114         btnAnalizar.setBackground(new Color(70, 130, 180));
115
116         Timer timer = new Timer(2000, e -> {
117             btnAnalizarActionPerformed(evt);
118             btnAnalizar.setText("Analizar Código");
119             btnAnalizar.setEnabled(true);
120             btnAnalizar.setBackground(new Color(97, 218, 251));
121         });
122         timer.setRepeats(false);
123         timer.start();
124     });
125
126     btnAnalizar.addMouseListener(new java.awt.event.MouseAdapter() {
127         @Override
128         public void mouseEntered(java.awt.event.MouseEvent evt) {
129             btnAnalizar.setBackground(new Color(135, 206, 250));
130         }
131
132         @Override
133         public void mouseExited(java.awt.event.MouseEvent evt) {
134             btnAnalizar.setBackground(new Color(97, 218, 251));
135         }
136     });
137
138     // Barra de estado inferior
139     statusBar = new JLabel("Modo: Edición");
140     statusBar.setFont(new Font("Consolas", Font.PLAIN, 14));
141     statusBar.setForeground(new Color(220, 220, 220));
142     statusBar.setBackground(new Color(35, 35, 35));
143     statusBar.setOpaque(true);
144     statusBar.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));
145
146     // Panel inferior
147     JPanel bottomPanel = new JPanel(new BorderLayout());
148     bottomPanel.setBackground(new Color(28, 28, 28));
149     bottomPanel.add(btnAnalizar, BorderLayout.EAST);
150     bottomPanel.add(statusBar, BorderLayout.CENTER);
151
152     // Divisor ajustable
153     JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, editorPanel, scrollResultado);
154     splitPane.setDividerLocation(300);
155     splitPane.setResizeWeight(0.7);
156     splitPane.setContinuousLayout(true);
157     splitPane.setBorder(null);
158
159     mainPanel.add(sidebar, BorderLayout.WEST);
160     mainPanel.add(tabBar, BorderLayout.NORTH);
161     mainPanel.add(splitPane, BorderLayout.CENTER);
162     mainPanel.add(bottomPanel, BorderLayout.SOUTH);
163
164     getContentPane().add(mainPanel);
165     pack();
```



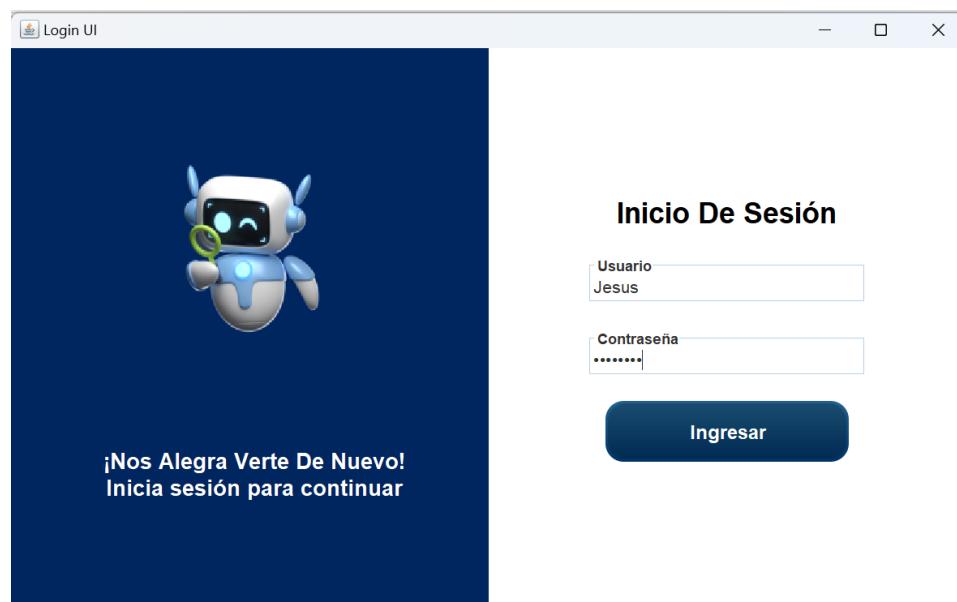
```
166
167     // Atajos de teclado
168     addKeyboardShortcuts();
169 }
170
171 private void addKeyboardShortcuts() {
172     txtEntrada.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW)
173         .put(KeyStroke.getKeyStroke(KeyEvent.VK_S, InputEvent.CTRL_DOWN_MASK), "save");
174     txtEntrada.getActionMap().put("save", new AbstractAction() {
175         @Override
176         public void actionPerformed(ActionEvent e) {
177             statusBar.setText("Archivo guardado.");
178         }
179     });
180
181     txtEntrada.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW)
182         .put(KeyStroke.getKeyStroke(KeyEvent.VK_R, InputEvent.CTRL_DOWN_MASK), "analyze");
183     txtEntrada.getActionMap().put("analyze", new AbstractAction() {
184         @Override
185         public void actionPerformed(ActionEvent e) {
186             btnAnalizarActionPerformed(null);
187         }
188     });
189 }
190
191 private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {
192     statusBar.setText("Analizando...");
193     File archivo = new File("archivo.txt");
194     try (PrintWriter escribir = new PrintWriter(archivo)) {
195         escribir.print(txtEntrada.getText());
196     } catch (FileNotFoundException ex) {
197         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
198     }
199
200     try (Reader lector = new BufferedReader(new FileReader("archivo.txt"))) {
201         Lexer lexer = new Lexer(lector);
202         StringBuilder resultado = new StringBuilder();
203         while (true) {
204             Tokens tokens = lexer.yylex();
205             if (tokens == null) {
206                 resultado.append("FIN");
207                 txtResultado.setText(resultado.toString());
208                 statusBar.setText("Análisis completado.");
209                 return;
210             }
211             switch (tokens) {
212                 case ERROR:
213                     resultado.append("Simbolo no definido\n");
214                     break;
215                 case Identificador:
216                 case Numero:
217                 case Reservadas:
218                     resultado.append(lexer.lexeme).append(": Es un ").append(tokens).append("\n");
219                     break;
220                 default:
221                     resultado.append("Token: ").append(tokens).append("\n");
222                     break;
223             }
224         }
225     } catch (IOException ex) {
226         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
227     }
228 }
229
230 public static void main(String args[]) {
231     java.awt.EventQueue.invokeLater(() -> new FrmPrincipal().setVisible(true));
232 }
233
234 // Variables
235 private JPanel mainPanel;
236 private JPanel tabBar;
237 private JLabel tabl;
238 private JTextArea txtEntrada;
239 private JTextArea txtResultado;
240 private JTextArea lineNumberArea;
241 private JButton btnAnalizar;
242 private JLabel statusBar;
243 }
```

## PANTALLAS RESULTANTES CON PRUEBAS DEL RECONOCIMIENTO DE LOS TOKENS

### Pantalla Splash



### Pantalla Login





## Pantalla Principal

Analizador Léxico - IDE Simulado

archivo.txt

Explorador archivo.txt output.log README.md

```
1 // Escribe tu código aquí...
2
3
4
5
```

// Los resultados del análisis aparecerán aquí...

Modo: Edición

Analizar Código

## Pantalla principal con demostración de la verificación de los tokens

Analizador Léxico - IDE Simulado

archivo.txt

Explorador archivo.txt output.log README.md

```
1 // Escribe tu código aquí...
2 Clase hola(){
3 Ent edad=> 5;-
4 Ent edad2=>8;-
5 Ent edad=> 5;-
6 Ent soy =>25;-
7 Flot ed => 7.7 ;-
8
9 suma + suma2 ;-
10 ciclo ( hola => 5 ;-; Boo ;-; hola++ ){
11 Ent hola ++ ;-
12 }
13 }
```

Token: clase  
hola: Es un Identificador  
Token: parentizq  
Token: parentDer  
Token: llaveIzq  
Token: valorEntero  
edad: Es un Identificador  
Token: asignacion

Analisis completado.

Analizar Código

## ANALIZADOR SINTACTICO

### DESCRIPCIÓN DEL PROBLEMA

La construcción de un analizador sintáctico (parser) en Java es una tarea compleja que forma una parte crucial del proceso de compilación o interpretación de un lenguaje de programación. Su función es analizar una secuencia de tokens, previamente generada por un analizador léxico, para verificar que estos sigan las reglas gramaticales de un lenguaje específico. En caso de encontrar una estructura inválida, el analizador debe identificar y reportar el error de manera clara. Este proceso es esencial para garantizar que el código fuente sea interpretable o ejecutable.

Uno de los principales desafíos al construir un analizador sintáctico es definir y manejar la gramática del lenguaje. La gramática es un conjunto de reglas que define cómo los tokens pueden combinarse para formar estructuras válidas, como expresiones aritméticas, declaraciones de variables o estructuras de control. Para poder analizar correctamente un programa, es necesario crear un modelo de estas reglas de manera precisa, y asegurar que el analizador las respete al procesar la entrada.

Un aspecto desafiante es el manejo de errores sintácticos. Los errores deben ser detectados de manera precisa y reportados con información útil para el usuario. Además, es fundamental que el analizador sea capaz de recuperarse de algunos errores y continuar el análisis de las partes restantes del programa, lo que complica aún más la implementación.

El proyecto consiste en la construcción de un **analizador sintáctico** para un lenguaje de programación específico, utilizando el lenguaje de programación **Java**. El objetivo principal de este proyecto es desarrollar un sistema que reciba un conjunto de **tokens** generados por un analizador léxico y los procese conforme a las reglas de la gramática del lenguaje, verificando si la secuencia de tokens cumple con las reglas sintácticas. El analizador deberá ser capaz de identificar estructuras válidas y, en caso de encontrar un error, reportarlo de manera adecuada, señalando la ubicación y naturaleza del mismo.

### Objetivo del Proyecto

El propósito de este proyecto es implementar un analizador sintáctico que sea capaz de analizar la entrada de un lenguaje de programación, detectar errores sintácticos y, en algunos casos, recuperar el flujo del análisis para continuar con el procesamiento del código. Este sistema forma parte fundamental de un compilador o intérprete, y su correcta implementación garantiza que el código fuente se pueda transformar en una estructura semántica interpretable o ejecutable. La correcta implementación del analizador sintáctico es crucial, ya que asegura que el código siga las reglas gramaticales del lenguaje y facilita la detección de errores en etapas tempranas del proceso de compilación.

La construcción de un analizador sintáctico en Java es una tarea compleja que implica una comprensión profunda de las estructuras gramaticales y el manejo eficiente de errores. El éxito de este proyecto permitirá contar con una herramienta fundamental para el análisis de código en un entorno de desarrollo, la cual ayudará a verificar la validez de las estructuras sintácticas y proporcionará una base sólida para etapas posteriores de compilación o interpretación.

## DESCRIPCIÓN DE CADA UNO DE LOS ARCHIVOS GENERADOS EN EL PROYECTO

### Archivo Lexer.flex

El archivo Lexer.flex es un componente clave en el desarrollo de un compilador o intérprete. Este archivo define el analizador léxico, una herramienta que descompone el código fuente en unidades fundamentales llamadas *tokens*. Un *token* es una representación simbólica de elementos como operadores, palabras reservadas, identificadores, números, o símbolos especiales.

El propósito principal de este archivo es procesar y clasificar el código fuente de manera sistemática, eliminando elementos irrelevantes como espacios y comentarios, para facilitar el análisis sintáctico en las etapas posteriores del procesamiento del lenguaje. Además, al manejar errores mediante el token ERROR, el analizador contribuye a detectar entradas no válidas en el código, mejorando la eficiencia del sistema.

Este archivo actúa como el primer filtro del procesamiento del lenguaje, estableciendo una base sólida para que el resto del sistema interprete o compile el código fuente de manera eficiente y precisa. Es esencial para automatizar el reconocimiento de los elementos del lenguaje y garantizar que el código fuente sea correctamente comprendido, minimizando errores y facilitando el desarrollo general del proyecto.

### Código Realizado:

```
1 package codigo;
2 import static codigo.Tokens.*;
3 /**
4  *class Lexer
5  *type Tokens
6  L=[a-zA-Z_]+
7  D=[0-9]+
8  espacio=[ ,\t,\r]+
9  {
10     public String lexeme;
11 }
12 /**
13
14 /* Espacios en blanco */
15 {espacio} /*Ignore*/
16
17 /* Comentarios */
18 "/*.*" /*Ignore*/
19
20 /* Salto de linea */
21 "\n" {return Linea;}
22
23 /* Comillas */
24 "\"\"" {lexeme=yytext(); return comillaDoble;}
25 "''" {lexeme=yytext(); return comillaSimple;}
26
27 /* Tipos de datos */
28 "Ent" {lexeme=yytext(); return valorEntero;}
29 "Flot" {lexeme=yytext(); return valorFlotante;}
30 "Boo" {lexeme=yytext(); return valorBooleano;}
31 "Cad" {lexeme=yytext(); return cadena;}
32 "Cars" {lexeme=yytext(); return carácter;}
```



```
34 /* Palabras reservadas */
35 "si" {lexeme=yytext(); return condicionalIf;}
36 "siNo" {lexeme=yytext(); return condicionalElse;}
37 "ciclo" {lexeme=yytext(); return cicloFor;}
38 "mientras" {lexeme=yytext(); return cicloWhile;}
39 "hacer" {lexeme=yytext(); return cicloDo;}
40 "segun" {lexeme=yytext(); return funcionSwitch;}
41 "caso" {lexeme=yytext(); return funcionCase;}
42 "salir" {lexeme=yytext(); return funcionSalirCase;}
43 "predeterminado" {lexeme=yytext(); return predeterminado;}
44 "Detener" {lexeme=yytext(); return detener;}
45 "Constante" {lexeme=yytext(); return constante;}
46 "Variable" {lexeme=yytext(); return variable;}
47 "funcion" {lexeme=yytext(); return funcion;}
48 "Clase" {lexeme=yytext(); return clase;}
49 "Imprimir" {lexeme=yytext(); return imprimir;}
50 "Leer" {lexeme=yytext(); return funcionLeer;}
51 "IntroducirD" {lexeme=yytext(); return funcionIntroducirDatos;}
52
53 /* Operadores matemáticos */
54 "+" {lexeme=yytext(); return operadorSuma;}
55 "-" {lexeme=yytext(); return operadorResta;}
56 "*" {lexeme=yytext(); return operadorMultiplicacion;}
57 "/" {lexeme=yytext(); return operadorDivision;}
58 "%" {lexeme=yytext(); return operadorModulo;}
59 "^" {lexeme=yytext(); return operadorPotencia;}
60 "Rcuad" {lexeme=yytext(); return operadorRaiz;}
61
62 /* Operadores de asignación y comparación */
63 ">=" {lexeme=yytext(); return asignacion;}
64 "==" {lexeme=yytext(); return comparacionIgualdad;}
65 ">" {lexeme=yytext(); return mayorQue;}
66 "<" {lexeme=yytext(); return menorQue;}
67
68 "<" {lexeme=yytext(); return menorQue;}
69 ">=<" {lexeme=yytext(); return mayorIgual;}
70 "<=>" {lexeme=yytext(); return menorIgual;}
71
72 /* Operadores lógicos */
73 "s" {lexeme=yytext(); return operadorY;}
74 "o" {lexeme=yytext(); return operadorO;}
75 "no" {lexeme=yytext(); return operadorNo;}
76
77 /* Operadores incrementales */
78 "++" {lexeme=yytext(); return incremento;}
79 "-+" {lexeme=yytext(); return decremento;}
80
81 /* Caracteres especiales */
82 "(" {lexeme=yytext(); return parentIzq;}
83 ")" {lexeme=yytext(); return parentDer;}
84 "{" {lexeme=yytext(); return llaveIzq;}
85 "}" {lexeme=yytext(); return llaveDer;}
86 "[" {lexeme=yytext(); return corchetIzq;}
87 "]" {lexeme=yytext(); return corchetDer;}
88 ";" {lexeme=yytext(); return finLinea;}
89 "<<" {lexeme=yytext(); return inicioTexto;}
90 ">>" {lexeme=yytext(); return finalTexto;}
91 ".," {lexeme=yytext(); return puntoDecimal;}
92 ":" {lexeme=yytext(); return dosPuntos;}
93 " " {lexeme=yytext(); return espacio;}
94
95 /* Identificadores y números */
96 ({L}|{D})* {lexeme=yytext(); return Identificador;}
97 ("-"{D}+"){D}+ {lexeme=yytext(); return Número;}
98
99 /* Errores */
100 . {return ERROR;}
```

## Archivo Lexer.java (Generado automáticamente)

El archivo Lexer.java es una clase generada automáticamente por la herramienta JFlex, diseñada para construir un analizador léxico o escáner. Este archivo se encarga de identificar los distintos componentes léxicos o tokens de un lenguaje definido en una especificación, en este caso el archivo Lexer.flex. Los tokens pueden representar palabras clave, identificadores, operadores, literales, y otros elementos sintácticos básicos del lenguaje.

El analizador léxico es una parte esencial de un compilador o intérprete, ya que actúa como el primer paso en el análisis del código fuente. Este archivo tiene la responsabilidad de leer el flujo de entrada de caracteres y convertirlo en una secuencia organizada de tokens que el analizador sintáctico puede entender. Su importancia radica en garantizar que el código fuente se procese de manera eficiente y que cualquier error léxico se detecte desde etapas tempranas del análisis. Este proceso es fundamental para garantizar la correcta interpretación y ejecución del código fuente.

### Código Generado:

```
1  /* The following code was generated by JFlex 1.4.3 on 01/01/25, 12:03 */
2
3  package codigo;
4  import static codigo.Tokens.*;
5
6  /**
7   * This class is a scanner generated by
8   * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9   * on 01/01/25, 12:03 from the specification file
10  * <tt>C:/Users/jesus/Downloads/AnalizadorLexico/src/codigo/Lexer.flex</tt>
11  */
12  class Lexer {
13
14  /** This character denotes the end of file */
15  public static final int YYEOF = -1;
16
17  /** initial size of the lookahead buffer */
18  private static final int ZZ_BUFSIZE = 16384;
19
20  /** lexical states */
21  public static final int YYINITIAL = 0;
22
23  /**
24   * ZZ_LEXSTATE[l] is the state in the DFA for the lexical state l
25   * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
26   * at the beginning of a line
27   * l is of the form l = 2*k, k a non negative integer
28   */
29  private static final int ZZ_LEXSTATE[] = {
30      0, 0
31  };
32
33  /**
```



## **MANUAL DE PRÁCTICAS**



```
34     * Translates characters to character classes
35     */
36     private static final String ZZ_CMAP_PACKED =
37         "\u11\u01\u31\u16\u20\u11\u32\u01\u47\u14\u21\u65\u171+" +
38         "\u11\u60\u11\u22\u14\u40\u11\u66\u14\u45\u11\u50\u11\u21\u11\u17\u13\u120+" +
39         "\u11\u64\u11\u15\u12\u11\u46\u11\u57\u11\u37\u11\u31\u132\u20\u11\u44+" +
40         "\u11\u54\u11\u55\u11\u74\u11\u52\u11\u53\u24\u44\u11\u77\u21\u44\u11\u100\u11\u44+" +
41         "\u11\u35\u11\u44\u11\u33\u11\u44\u11\u24\u3\u44\u11\u75\u4\u44\u11\u62\u11\u67+" +
42         "\u11\u63\u11\u23\u11\u11\u01\u27\u11\u76\u11\u25\u11\u30\u11\u10\u11\u7+" +
43         "\u11\u36\u11\u14\u11\u2\u4\u2\u43\u11\u11\u17\u2\u11\u5\u11\u34\u11\u73\u11\u43+" +
44         "\u11\u56\u11\u12\u11\u6\u11\u26\u11\u43\u11\u13\u3\u43\u11\u61\u11\u41\u11\u51+" +
45         "\u11\u43\u01\u11\u70\uuff5e\u10";
46
47     /**
48     * Translates characters to character classes
49     */
50     private static final char [] ZZ_CMAP = zzUnpackCMap(ZZ_CMAP_PACKED);
51
52     /**
53     * Translates DFA states to action switch labels.
54     */
55     private static final int [] ZZ_ACTION = zzUnpackAction();
56
57     private static final String ZZ_ACTION_PACKED_0 =
58         "\u10\u11\u12\u13\u14\u15\u16\u17+" +
59         "\u11\u10\u11\u12\u13\u14\u15\u11\u15+" +
60         "\u12\u14\u11\u16\u17\u12\u20\u11\u21\u11\u14\u11\u22\u11\u23+" +
61         "\u11\u24\u11\u25\u11\u14\u11\u11\u26\u11\u27\u11\u30\u11\u31+" +
62         "\u11\u32\u11\u33\u11\u34\u11\u35\u11\u36\u11\u37\u2\u54\u11\u14+" +
63         "\u11\u40\u11\u41\u11\u40\u11\u42\u2\u40\u11\u43\u4\u40\u11\u4+" +
64         "\u11\u44\u11\u45\u11\u40\u11\u46\u11\u47\u11\u01\u11\u50\u11\u40+" +
65         "\u11\u51\u11\u40\u11\u01\u11\u52\u11\u01\u11\u53\u11\u06\u40+" +
66         "\u11\u01\u54\u11\u01\u21\u40\u11\u55\u11\u56\u11\u57\u11\u60+" +
67
68         "\u11\u01\u61\u11\u62\u11\u63\u11\u64\u11\u40\u11\u65\u11\u40+" +
69         "\u11\u66\u2\u40\u11\u67\u11\u01\u10\u40\u11\u70\u5\u40\u11\u71+" +
70         "\u11\u41\u11\u72\u11\u40\u11\u73\u11\u40\u11\u01\u7\u40\u11\u74+" +
71         "\u11\u40\u11\u75\u11\u76\u11\u77\u11\u100\u11\u101\u11\u102\u11\u40+" +
72         "\u11\u103\u20\u40\u11\u104\u3\u11\u40\u11\u105\u5\u40\u11\u106\u11\u40+" +
73         "\u11\u107\u11\u40\u11\u110\u11\u111\u11\u112\u5\u40\u11\u113\u2\u40+" +
74         "\u11\u114";
75
76     private static int [] zzUnpackAction() {
77         int [] result = new int[208];
78         int offset = 0;
79         offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);
80         return result;
81     }
82
83     private static int zzUnpackAction(String packed, int offset, int [] result) {
84         int i = 0; /* index in packed string */
85         int j = offset; /* index in unpacked array */
86         int l = packed.length();
87         while (i < l) {
88             int count = packed.charAt(i++);
89             int value = packed.charAt(i++);
90             do result[j++] = value; while (--count > 0);
91         }
92         return j;
93     }
94
95     /**
96     * Translates a state to a row index in the transition table
97     */
98     private static final int [] ZZ_ROWMAP = zzUnpackRowMap();
```



```
100 private static final String ZZ_ROWMAP_PACKED_0 =
101     "\u00a0\u00a0\u101\u00a0\u202\u00a0\u303\u00a0\u00a0\u0104\u00a0\u00a0\u0145\u00a0\u202\u00a0\u00a0\u0186"+
102     "\u00a0\u01c7\u00a0\u00a0\u208\u00a0\u00a0\u0249\u00a0\u00a0\u028a\u00a0\u00a0\u02cb\u00a0\u00a0\u030c\u00a0\u00a0\u034d\u00a0\u101"+
103     "\u00a0\u101\u00a0\u101\u00a0\u00a0\u038e\u00a0\u00a0\u03cf\u00a0\u00a0\u0410\u00a0\u00a0\u0451\u00a0\u00a0\u0492\u00a0\u00a0\u04d3"+
104     "\u00a0\u0514\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u202\u00a0\u00a0\u0555\u00a0\u00a0\u0596\u00a0\u101"+
105     "\u00a0\u05d7\u00a0\u00a0\u0618\u00a0\u00a0\u0659\u00a0\u00a0\u069a\u00a0\u00a0\u06db\u00a0\u00a0\u071c\u00a0\u00a0\u075d\u00a0\u101"+
106     "\u00a0\u079e\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101\u00a0\u101"+
107     "\u00a0\u07df\u00a0\u00a0\u0820\u00a0\u00a0\u0861\u00a0\u00a0\u0882\u00a0\u00a0\u08e3\u00a0\u00a0\u0924\u00a0\u202\u00a0\u303"+
108     "\u00a0\u0965\u00a0\u202\u00a0\u00a0\u09a6\u00a0\u00a0\u09e7\u00a0\u00a0\u0a28\u00a0\u00a0\u0aa\u00a0\u00a0\u0aeb"+
109     "\u00a0\u0b2c\u00a0\u00a0\u0b6d\u00a0\u101\u00a0\u101\u00a0\u00a0\u0bae\u00a0\u00a0\u0bef\u00a0\u00a0\u0c30\u00a0\u101"+
110     "\u00a0\u101\u00a0\u0c71\u00a0\u101\u00a0\u00a0\u0cb2\u00a0\u202\u00a0\u00a0\u0cf3\u00a0\u00a0\u0d34\u00a0\u101"+
111     "\u00a0\u0d75\u00a0\u00a0\u0db6\u00a0\u00a0\u0df7\u00a0\u101\u00a0\u00a0\u0e38\u00a0\u00a0\u0eba\u00a0\u00a0\u0efb"+
112     "\u00a0\u0f3c\u00a0\u00a0\u0fd7\u00a0\u00a0\u0fbef\u00a0\u00a0\u101\u00a0\u00a0\u1040\u00a0\u00a0\u081\u00a0\u10c2"+
113     "\u00a0\u1103\u00a0\u1144\u00a0\u1185\u00a0\u00a0\u11c6\u00a0\u00a0\u1207\u00a0\u00a0\u1248\u00a0\u00a0\u1289\u00a0\u12ca"+
114     "\u00a0\u130b\u00a0\u134c\u00a0\u138d\u00a0\u00a0\u13ce\u00a0\u00a0\u140f\u00a0\u00a0\u1450\u00a0\u00a0\u1491\u00a0\u101"+
115     "\u00a0\u202\u00a0\u202\u00a0\u101\u00a0\u00a0\u14d2\u00a0\u101\u00a0\u00a0\u101\u00a0\u00a0\u202"+
116     "\u00a0\u1513\u00a0\u202\u00a0\u00a0\u1554\u00a0\u202\u00a0\u00a0\u1599\u00a0\u00a0\u15d6\u00a0\u101\u00a0\u00a0\u1617"+
117     "\u00a0\u1658\u00a0\u00a0\u1699\u00a0\u00a0\u16da\u00a0\u00a0\u171b\u00a0\u00a0\u175c\u00a0\u00a0\u179d\u00a0\u00a0\u17de\u00a0\u00a0\u181f"+
118     "\u00a0\u202\u00a0\u1860\u00a0\u00a0\u18a1\u00a0\u00a0\u18e2\u00a0\u00a0\u1923\u00a0\u00a0\u1964\u00a0\u202\u00a0\u101"+
119     "\u00a0\u202\u00a0\u19a5\u00a0\u202\u00a0\u00a0\u199e\u00a0\u00a0\u1a27\u00a0\u00a0\u1a68\u00a0\u00a0\u1aa9\u00a0\u00a0\u1aea"+
120     "\u00a0\u1b2b\u00a0\u00a0\u1b6c\u00a0\u00a0\u1bad\u00a0\u00a0\u1bee\u00a0\u202\u00a0\u00a0\u1c2f\u00a0\u00a0\u202\u00a0\u202"+
121     "\u00a0\u202\u00a0\u202\u00a0\u00a0\u1c70\u00a0\u00a0\u101\u00a0\u00a0\u1cb1\u00a0\u00a0\u1cf2"+
122     "\u00a0\u01d3\u00a0\u00a0\u1d74\u00a0\u00a0\u1fdb5\u00a0\u00a0\u1df6\u00a0\u00a0\u1e37\u00a0\u00a0\u1e78\u00a0\u00a0\u1e99\u00a0\u00a0\u1efa"+
123     "\u00a0\u1f3b\u00a0\u00a0\u1f7c\u00a0\u00a0\u1fbf\u00a0\u00a0\u1ffe\u00a0\u00a0\u203f\u00a0\u00a0\u2080\u00a0\u202\u00a0\u20c1"+
124     "\u00a0\u2102\u00a0\u2143\u00a0\u202\u00a0\u2184\u00a0\u00a0\u21c5\u00a0\u00a0\u2206\u00a0\u00a0\u2247\u00a0\u00a0\u2288"+
125     "\u00a0\u202\u00a0\u22c9\u00a0\u202\u00a0\u230a\u00a0\u00a0\u202\u00a0\u00a0\u202\u00a0\u00a0\u234b"+
126     "\u00a0\u238c\u00a0\u00a0\u23cd\u00a0\u00a0\u240e\u00a0\u00a0\u244f\u00a0\u00a0\u202\u00a0\u00a0\u2490\u00a0\u00a0\u24d1\u00a0\u00a0\u202";
127
128     private static int [] zzUnpackRowMap() {
129         int [] result = new int[208];
130         int offset = 0;
131         offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);
132         return result;
133     }
134
135     private static int zzUnpackRowMap(String packed, int offset, int [] result) {
136         int i = 0; /* index in packed string */
137         int j = offset; /* index in unpacked array */
138         int l = packed.length();
139         while (i < l) {
140             int high = packed.charAt(i++) << 16;
141             result[j++] = high | packed.charAt(i++);
142         }
143         return j;
144     }
145
146     /**
147      * The transition table of the DFA
148     */
149     private static final int [] ZZ_TRANS = zzUnpackTrans();
150
151     private static final String ZZ_TRANS_PACKED_0 =
152     "\u1\u2\u1\u3\u1\u4\u1\u5\u1\u6\u2\u7\u1\u10\u1\u11"+
153     "\u1\u7\u1\u12\u1\u13\u1\u14\u1\u15\u1\u16\u1\u17"+
154     "\u1\u20\u1\u21\u1\u22\u1\u23\u1\u24\u1\u25\u1\u26"+
155     "\u1\u27\u1\u28\u1\u29\u1\u2a\u1\u2b\u1\u2c\u1\u2d"+
156     "\u1\u2\u1\u3\u1\u36\u1\u37\u1\u5\u1\u40\u1\u41\u1\u42"+
157     "\u1\u43\u1\u44\u1\u45\u1\u46\u1\u47\u1\u50\u1\u51"+
158     "\u1\u52\u1\u53\u1\u54\u1\u55\u1\u56\u1\u57\u1\u60\u1\u61"+
159     "\u1\u62\u1\u63\u1\u64\u1\u65\u1\u66\u1\u67\u1\u68"+
160     "\u1\u69\u1\u6a\u1\u6b\u1\u6c\u1\u6d\u1\u6e\u1\u6f"+
161     "\u5\u0\u5\u67\u1\u3\u0\u7\u67\u2\u0\u1\u70\u1\u0\u1\u5"+
162     "\u1\u2\u0\u1\u3\u1\u30\u0\u1\u5\u32\u0\u2\u67\u1\u0\u1\u67"+
163     "\u1\u71\u1\u67\u1\u72\u5\u67\u7\u0\u5\u67\u2\u0\u4\u67"+
164     "\u4\u0\u2\u67\u5\u0\u5\u67\u1\u3\u0\u7\u67\u1\u0\u2\u67"+
165     "\u1\u0\u1\u67\u1\u70\u2\u67\u1\u73\u2\u67\u2\u0\u4\u67"+
```



# MANUAL DE PRÁCTICAS



166 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\0\2\67"+  
167 "1\1\0\5\67\1\74\3\67\7\0\5\67\2\0\4\67"+  
168 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\0\2\67"+  
169 "1\1\0\1\1\75\3\67\1\1\76\4\67\7\1\0\3\67\1\1\77"+  
170 "1\1\67\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\0"+  
171 "1\7\67\1\1\0\2\67\1\0\1\0\67\1\1\100\7\0\5\67"+  
172 "1\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\7\67"+  
173 "1\1\0\2\67\1\1\0\1\1\67\7\0\3\67\1\1\101\1\67"+  
174 "1\2\0\4\67\4\0\2\67\5\0\5\67\1\3\0\7\67"+  
175 "1\1\5\0\1\1\102\1\0\1\1\103\1\0\1\1\104\61\0\2\67"+  
176 "1\1\0\1\1\1\67\1\0\1\1\67\1\1\105\3\67\2\0\1\4\67"+  
177 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
178 "1\1\0\1\1\106\1\0\1\67\7\0\3\67\1\1\107\1\67\2\0\0"+  
179 "1\4\0\67\4\0\2\67\5\0\5\67\1\3\0\7\67\3\1\0\0"+  
180 "1\1\110\1\1\111\77\0\1\112\1\113\4\7\1\0\2\67\1\1\0\0"+  
181 "1\1\1\67\7\0\5\67\1\2\0\1\67\1\1\114\2\1\67\4\0\0"+  
182 "1\2\1\7\5\0\5\67\1\3\0\5\67\1\1\115\1\1\67\1\1\0\0"+  
183 "1\2\1\67\1\1\0\4\67\1\1\116\4\67\7\0\5\67\1\2\0\0"+  
184 "1\4\0\67\4\0\2\67\5\0\5\67\1\3\0\7\67\3\1\0\0"+  
185 "1\1\117\5\0\1\1\120\61\0\1\1\121\25\0\1\1\122\52\5\0\0"+  
186 "1\1\123\27\0\1\1\124\61\0\1\1\125\50\0\1\2\67\1\1\0\0"+  
187 "1\1\67\1\1\126\7\0\5\67\1\2\0\4\67\1\4\1\0\0"+  
188 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
189 "1\5\0\67\1\1\127\3\1\67\7\0\5\67\1\2\0\4\67\1\4\0\0"+  
190 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
191 "1\1\1\67\7\0\5\67\1\2\0\1\67\1\1\130\2\1\67\4\0\0"+  
192 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
193 "1\5\0\67\1\1\131\3\1\67\7\0\3\67\1\1\132\1\67\2\0\0"+  
194 "1\1\67\1\1\133\2\1\67\4\0\1\2\67\5\0\5\67\1\3\0\0"+  
195 "1\7\67\1\2\0\0\1\134\7\1\0\1\1\135\152\0\1\1\136\16\0\0"+  
196 "1\2\1\67\1\0\1\1\137\10\67\7\0\5\67\2\0\1\4\67"+  
197 "1\4\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
198 "1\1\0\1\1\67\7\0\5\67\2\0\4\67\1\0\2\67"+

199 "1\5\0\4\67\1\1\140\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
200 "1\4\1\67\1\1\141\4\1\67\7\0\5\67\2\0\4\1\67\4\1\0\0"+  
201 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
202 "1\1\1\67\7\7\0\3\67\1\1\142\3\67\2\0\4\1\67\4\1\0\0"+  
203 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\0"+  
204 "1\1\1\67\1\1\143\7\67\7\0\5\67\2\0\4\1\67\4\1\0\0"+  
205 "1\2\1\67\5\0\5\67\1\3\0\1\144\6\1\67\1\1\0\2\67"+  
206 "1\1\0\4\67\1\1\145\4\67\7\0\5\67\1\2\0\4\1\67"+  
207 "1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
208 "1\1\0\2\67\1\1\12\6\6\67\1\7\0\5\67\2\0\1\4\1\67"+  
209 "1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
210 "1\1\0\1\1\67\1\1\146\7\67\7\0\5\67\2\0\4\1\67"+  
211 "1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
212 "1\1\0\6\67\1\1\147\2\67\7\0\5\67\1\2\0\4\1\67"+  
213 "1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
214 "1\1\0\1\1\67\7\0\5\67\1\2\0\2\67\1\1\150\1\67"+  
215 "1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67"+  
216 "1\1\0\1\1\67\7\0\5\67\1\2\0\3\67\1\1\151\1\4\0\0"+  
217 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67\1\1\0\0"+  
218 "1\1\1\67\1\1\152\3\67\7\0\5\67\2\0\4\1\67\4\1\0\0"+  
219 "1\2\1\67\5\0\5\67\1\3\0\7\67\1\1\0\2\67\1\1\0\0"+  
220 "1\1\1\53\1\0\67\7\0\5\67\1\2\0\4\1\67\4\1\0\2\67"+  
221 "1\1\0\5\67\1\3\0\7\67\1\1\10\2\67\1\1\0\1\1\67"+  
222 "1\1\0\2\67\1\1\154\3\67\2\0\4\1\67\4\1\0\2\67"+  
223 "1\1\0\5\67\1\3\0\7\67\1\1\16\102\1\1\0\62\1\0\2\1\0\0"+  
224 "1\2\1\67\1\1\0\1\1\67\7\0\2\67\1\1\155\2\1\67\2\0\0"+  
225 "1\1\1\67\1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\0"+  
226 "1\2\1\67\1\1\0\1\1\67\7\0\1\1\67\1\1\156\3\67\2\0\0"+  
227 "1\1\1\67\1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\0"+  
228 "1\2\1\67\1\1\0\1\1\67\1\1\157\2\67\7\0\5\67\2\0\0"+  
229 "1\1\1\67\1\1\0\2\67\5\0\5\67\1\3\0\7\67\1\1\0\0"+  
230 "1\1\1\60\42\0\1\2\67\1\1\0\1\6\67\1\1\161\2\67\7\0\0"+  
231 "1\1\1\67\1\2\0\4\67\1\1\0\2\67\5\0\5\67\1\3\0\0"+



# MANUAL DE PRÁCTICAS



232 "x\1\67\1\0\2\67\1\0\1\1\67\1\0\5\67\1\2\0" +  
233 "x\3\1\67\1\1\62\4\0\2\67\1\5\0\5\1\67\1\3\0\1\7\67" +  
234 "x\3\2\0\1\1\163\5\0\1\1\164\145\0\1\1\165\7\0\1\1\166" +  
235 "x\1\0\2\0\1\1\67\1\3\1\0\2\67\1\1\0\1\2\67\1\1\170\6\67" +  
236 "x\7\0\1\5\67\1\2\0\4\1\67\1\0\2\67\1\3\0\1\5\1\67" +  
237 "x\1\3\0\1\7\1\67\1\1\1\1\2\67\1\1\0\1\1\67\1\1\0\5\67" +  
238 "x\1\0\1\1\67\1\1\1\1\2\67\1\4\0\1\2\67\1\5\0\5\1\67" +  
239 "x\1\3\0\1\7\1\67\1\1\0\2\67\1\1\0\1\1\67\1\1\0\5\67" +  
240 "x\2\0\1\1\67\1\1\1\2\67\1\4\0\1\2\67\1\5\0\5\1\67" +  
241 "x\1\3\0\1\7\1\67\1\1\0\2\67\1\1\0\1\1\67\1\1\0\3\67" +  
242 "x\1\1\73\1\1\67\1\2\0\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67" +  
243 "x\1\3\0\1\7\1\67\1\1\0\2\67\1\1\0\1\1\67\1\7\0\4\1\67" +  
244 "x\1\1\74\1\2\0\4\1\67\1\1\0\2\67\1\5\0\5\1\67\1\1\175" +  
245 "x\1\3\0\1\7\1\67\1\1\0\2\67\1\1\0\1\1\67\1\1\176\7\67" +  
246 "x\7\0\1\5\67\1\2\0\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67" +  
247 "x\1\3\0\1\7\1\67\1\5\7\1\0\1\1\7\1\4\1\0\1\2\200\61\0\2\1\67" +  
248 "x\1\1\0\1\4\1\67\1\1\201\1\4\1\67\1\7\0\5\1\67\1\2\0\4\1\67" +  
249 "x\4\0\1\2\67\1\5\0\5\1\67\1\3\1\0\1\7\67\1\1\0\1\2\67" +  
250 "x\1\1\0\1\4\1\67\1\1\202\1\4\1\67\1\7\0\5\1\67\1\2\0\4\1\67" +  
251 "x\4\0\1\2\67\1\5\0\5\1\67\1\3\1\0\1\7\67\1\1\0\1\2\67" +  
252 "x\1\1\0\1\2\67\1\1\203\1\6\1\7\0\5\1\67\1\2\0\4\1\67" +  
253 "x\4\0\1\2\67\1\5\0\5\1\67\1\3\1\0\1\7\67\1\1\0\1\2\67" +  
254 "x\1\1\0\1\1\1\67\1\1\0\5\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
255 "x\5\0\1\4\1\67\1\1\204\1\3\1\0\1\7\67\1\1\0\1\2\67\1\3\1\0" +  
256 "x\2\1\67\1\1\205\1\6\1\67\1\7\0\5\67\1\2\0\1\4\1\67\1\4\0\1\0" +  
257 "x\2\1\67\1\5\1\0\5\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
258 "x\1\1\67\1\7\0\1\5\67\1\2\0\1\4\1\67\1\4\0\1\2\67\1\5\0\1\0" +  
259 "x\5\1\67\1\1\3\1\0\1\67\1\1\206\1\5\1\67\1\1\0\1\2\67\1\1\0" +  
260 "x\4\1\67\1\1\207\1\4\1\67\1\7\0\5\1\67\1\2\0\1\4\1\67\1\4\0\1\0" +  
261 "x\2\1\67\1\5\1\0\5\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
262 "x\1\1\67\1\7\0\1\1\67\1\1\210\1\3\1\67\1\2\0\1\4\1\67\1\4\0\1\0" +  
263 "x\2\1\67\1\5\1\0\5\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
264 "x\4\1\67\1\1\72\1\4\1\67\1\7\0\5\1\67\1\2\0\1\4\1\67\1\4\0\1\0" +

265 "x\2\1\67\1\5\1\0\5\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
266 "x\1\1\67\1\7\0\1\5\67\1\2\0\1\4\1\67\1\1\1\21\1\2\1\67\1\4\0\1\0" +  
267 "x\2\1\67\1\5\1\0\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
268 "x\1\1\67\1\7\0\1\2\67\1\1\212\1\2\67\1\2\0\1\4\1\67\1\4\0\1\0" +  
269 "x\2\1\67\1\5\1\0\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0" +  
270 "x\1\1\21\1\1\0\67\1\7\0\1\5\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
271 "x\5\1\0\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0\1\5\1\67" +  
272 "x\1\1\147\1\3\1\67\1\7\0\1\5\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
273 "x\5\1\0\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0\1\4\1\67" +  
274 "x\1\1\21\1\4\1\67\1\7\0\1\5\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
275 "x\5\1\0\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0\1\1\1\67" +  
276 "x\7\0\1\3\1\67\1\1\215\1\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
277 "x\5\1\0\1\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0\1\5\1\67" +  
278 "x\1\1\21\1\3\1\67\1\7\0\1\5\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67" +  
279 "x\5\1\0\1\5\1\67\1\1\3\1\0\1\7\67\1\1\0\1\2\67\1\1\0\1\1\1\67" +  
280 "x\7\0\1\5\1\67\1\2\0\1\1\67\1\1\217\1\2\1\67\1\1\0\1\2\67" +  
281 "x\5\1\0\1\5\1\67\1\1\3\1\0\1\7\67\1\2\0\1\1\164\1\5\1\0\1\2\20" +  
282 "x\3\1\0\1\2\1\67\1\1\0\1\2\67\1\1\221\1\6\1\67\1\7\0\1\5\1\67" +  
283 "x\2\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\7\67" +  
284 "x\1\1\0\1\2\67\1\1\0\1\0\1\67\1\1\222\1\2\1\67\1\7\0\1\5\1\67" +  
285 "x\2\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\7\67" +  
286 "x\1\1\0\1\2\67\1\1\0\1\0\1\67\1\1\223\1\2\1\67\1\7\0\1\5\1\67" +  
287 "x\2\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\7\67" +  
288 "x\1\1\0\1\2\67\1\1\0\1\0\1\67\1\1\224\1\2\1\67\1\7\0\1\5\1\67" +  
289 "x\2\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\7\67" +  
290 "x\6\2\0\1\1\225\1\7\0\1\2\67\1\1\0\1\0\1\67\1\1\226\1\7\67" +  
291 "x\7\0\1\5\1\67\1\2\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67" +  
292 "x\1\3\0\1\7\1\1\0\1\2\67\1\1\0\1\1\1\67\1\7\0\1\4\1\67" +  
293 "x\1\1\22\1\2\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\0" +  
294 "x\7\0\1\7\1\1\0\1\2\67\1\1\0\1\1\1\67\1\1\230\1\4\1\67\1\7\0\1\0" +  
295 "x\5\1\67\1\2\0\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\0" +  
296 "x\7\0\1\7\1\1\0\1\2\67\1\1\0\1\1\1\67\1\1\231\1\1\0\1\67\1\7\0\1\5\1\67" +  
297 "x\2\1\0\1\4\1\67\1\4\0\1\2\67\1\5\0\5\1\67\1\1\3\1\0\1\7\67"



# MANUAL DE PRÁCTICAS



298 "11026710116717056720467"+  
299 "41026715014671232130767110"+  
300 "21671101116717056712011671233"+  
301 "2167141026715014671234130767"+  
302 "11026711011167170567201467"+  
303 "41026715014671235130767110"+  
304 "216711011236106710567201467"+  
305 "4102671501467123107671101267"+  
306 "110167123717671705671201467"+  
307 "4102167150146711307671101267"+  
308 "110111671705672014671401267"+  
309 "51046712401307671101267110"+  
310 "1116717056712014671401267150"+  
311 "41671241130767110126711011167"+  
312 "710467124212014671401267150"+  
313 "51671310767110126711011167170"+  
314 "516712014671124312671401267150"+  
315 "516713107671101267110146711244"+  
316 "41671705672014671401267150"+  
317 "51671310767110126711012671245"+  
318 "61671705672014671401267150"+  
319 "5167131076716310112461610267110"+  
320 "2167124716170567201467110"+  
321 "2167130567130767110267110"+  
322 "41671212501467170567201467110"+  
323 "2167130567130767110267110"+  
324 "11012511767170567201467140"+  
325 "2167130567130767110267110"+  
326 "11167170367125211671201467140"+  
327 "2167130567130767110267110"+  
328 "111671705671201167112532167140"+  
329 "216715015671307167110267110"+  
330 "1116717056712014671401267150"+  
  
331 "41671125413071671102671301255"+  
332 "11016717056712014671401267150"+  
333 "51671310767110126711011167170"+  
334 "5167121011671125612671401267150"+  
335 "51671310767110126711011167170"+  
336 "3167121257116712014671401267150"+  
337 "51671310767110126711011167170"+  
338 "5167121014671401267150146711260"+  
339 "113017671201267110267112611667"+  
340 "71056720146714012671501567"+  
341 "1130176712012671101467112621467"+  
342 "7101567120146714012671501567"+  
343 "1130176712101267110111671701567"+  
344 "2101467140126715015671301467"+  
345 "112632167110267110111671701467"+  
346 "11264201467140126715015671300"+  
347 "71671102671101267112651667170"+  
348 "516712101467140126715015671300"+  
349 "71671102671101116717005671200"+  
350 "41671401267150156713012661667"+  
351 "110126711011167126717671101567"+  
352 "2101167110126715015671300767"+  
353 "110126711011167127017671101567"+  
354 "2101467140126715015671300767"+  
355 "1101267110111671700367112711367"+  
356 "2101467140126715015671300767"+  
357 "11012671101467127214671701567"+  
358 "2101467140126715015671300767"+  
359 "11012671101116717005671200467"+  
360 "4101267150146712731307671100"+  
361 "21671305671274136717005671200"+  
362 "41671401267150156713007671100"+  
363 "2167110111671702167127521671200"+



```
364 "14\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
365 "\2\67\1\0\11\67\7\0\3\67\1\276\1\67\2\0"+
366 "\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
367 "\2\67\1\0\1\277\10\67\7\0\5\67\2\0\4\67"+
368 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
369 "\1\0\2\67\1\300\6\67\7\0\5\67\2\0\4\67"+
370 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
371 "\1\0\6\67\1\301\2\67\7\0\5\67\2\0\4\67"+
372 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
373 "\1\0\11\67\7\0\5\67\2\0\4\67\4\0\2\67"+
374 "\5\0\4\67\1\302\13\0\7\67\1\0\2\67\1\0"+
375 "\4\67\1\1\303\4\67\7\0\5\67\2\0\4\67\4\0"+
376 "\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+
377 "\1\67\7\0\1\67\1\304\3\67\2\0\4\67\4\0"+
378 "\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+
379 "\1\1\67\7\0\5\67\12\0\4\67\1\0\2\67\5\0"+
380 "\4\67\1\1\305\13\0\7\67\1\0\2\67\1\0\11\67"+
381 "\7\0\5\67\2\0\4\67\1\0\2\67\5\0\4\67"+
382 "\1\306\13\0\7\67\1\0\2\67\1\0\4\67\1\307"+
383 "\4\67\1\0\5\67\2\0\4\67\4\0\2\67\5\0"+
384 "\5\67\13\0\7\67\1\0\2\67\1\0\11\67\7\0"+
385 "\5\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+
386 "\1\310\6\67\1\0\2\67\1\0\1\311\10\67\7\0"+
387 "\5\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+
388 "\7\67\1\0\2\67\1\0\1\312\10\67\7\0\5\67"+
389 "\2\0\4\67\4\0\2\67\5\0\5\67\13\0\7\67"+
390 "\1\0\2\67\1\0\11\67\7\0\5\67\2\0\4\67"+
391 "\4\0\2\67\5\0\4\67\1\313\13\0\7\67\1\0"+
392 "\2\67\1\0\1\67\1\314\7\67\7\0\5\67\2\0"+
393 "\4\67\1\0\2\67\5\0\5\67\13\0\7\67\1\0"+
394 "\2\67\1\0\11\67\7\0\5\67\2\0\4\67\4\0"+
395 "\2\67\5\0\5\67\13\0\2\67\1\315\4\67\1\0"+
396 "\2\67\1\0\11\67\7\0\3\67\1\316\1\67\2\0"+
```

```
397 "\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+
398 "\2\67\1\0\11\67\7\0\4\67\1\317\2\0\4\67"+
399 "\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+
400 "\1\0\11\67\7\0\5\67\2\0\4\67\1\320\2\67"+
401 "\4\0\2\67\5\0\5\67\13\0\7\67"+
402 private static int [] zzUnpackTrans() {
403     int [] result = new int[9490];
404     int offset = 0;
405     offset = zzUnpackTrans(SE_TRANS_PACKED_0, offset, result);
406     return result;
407 }
408
409 private static int zzUnpackTrans(String packed, int offset, int [] result) {
410     int i = 0; /* index in packed string */
411     int j = offset; /* index in unpacked array */
412     int l = packed.length();
413     while (i < l) {
414         int count = packed.charAt(i++);
415         int value = packed.charAt(i++);
416         value--;
417         do result[j++] = value; while (--count > 0);
418     }
419     return j;
420 }
421
422
423 /* error codes */
424 private static final int ZZ_UNKNOWN_ERROR = 0;
425 private static final int ZZ_NO_MATCH = 1;
426 private static final int ZZ_PUSHBACK_2BIG = 2;
427
428 /* error messages for the codes above */
```



## **MANUAL DE PRÁCTICAS**



```
430 private static final String ZZ_ERROR_MSG[] = {
431     "Unknown internal scanner error",
432     "Error: could not match input",
433     "Error: pushback value was too large"
434 };
435
436 /**
437 * ZZ_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
438 */
439 private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();
440
441 private static final String ZZ_ATTRIBUTE_PACKED_0 =
442     "\\" + '0' + '1' + '1' + '1' + '5' + '1' + '3' + '1' + '1' + '7' + '1' + '3' + '1' + '3' + '1' + '1' + '1' +
443     "\\" + '7' + '1' + '1' + '1' + '1' + '1' + '1' + '1' + '1' + '2' + '2' + '1' + '2' + '1' + '3' + '1' + '2' + '1' +
444     "\\" + '1' + '0' + '1' + '1' + '1' + '3' + '1' + '1' + '0' + '1' + '1' + '3' + '0' + '1' + '1' + '1' + '0' +
445     "\\" + '6' + '1' + '1' + '0' + '1' + '1' + '1' + '0' + '2' + '1' + '1' + '1' + '2' + '1' + '3' + '1' + '1' +
446     "\\" + '1' + '0' + '3' + '1' + '1' + '7' + '1' + '1' + '1' + '1' + '0' + '1' + '7' + '1' + '1' + '1' + '1' + '1' +
447     "\\" + '1' + '0' + '2' + '0' + '1' + '1' + '1' + '5' + '2' + '1" ;
448
449 private static int [] zzUnpackAttribute() {
450     int [] result = new int[208];
451     int offset = 0;
452     zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
453     return result;
454 }
455
456 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
457     int i = 0; /* index in packed string */
458     int j = offset; /* index in unpacked array */
459     int l = packed.length();
460     while (i < l) {
461         int count = packed.charAt(i++);
462         int value = packed.charAt(i++);
463
464         do result[j++] = value; while (--count > 0);
465     }
466     return j;
467 }
468
469 /**
470 * the input device */
471 private java.io.Reader zzReader;
472
473 /**
474 * the current state of the DFA */
475 private int zzState;
476
477 /**
478 * this buffer contains the current text to be matched and is
479 * the source of the yytext() string */
480 private char zzBuffer[] = new char[ZZ_BUFSIZE];
481
482 /**
483 * the textposition at the last accepting state */
484 private int zzMarkedPos;
485
486 /**
487 * the current text position in the buffer */
488 private int zzCurrentPos;
489
490 /**
491 * startRead marks the beginning of the yytext() string in the buffer */
492 private int zzStartRead;
493
494 /**
495 * endRead marks the last character in the buffer, that has been read
496 * from input */
497 private int zzEndRead;
498
499 /**
500 * number of newlines encountered up to the start of the matched text */
501 private int yyline;
```



```
496
497     /** the number of characters up to the start of the matched text */
498     private int yychar;
499
500     /**
501      * the number of characters from the last newline up to the start of the
502      * matched text
503      */
504     private int yycolumn;
505
506     /**
507      * zzAtBOL == true <=> the scanner is currently at the beginning of a line
508      */
509     private boolean zzAtBOL = true;
510
511     /**
512      * zzAtEOF == true <=> the scanner is at the EOF
513      */
514     private boolean zzAtEOF;
515
516     /**
517      * denotes if the user-EOF-code has already been executed
518      */
519     private boolean zEOFDone;
520
521     /**
522      * Creates a new scanner
523      * There is also a java.io.InputStream version of this constructor.
524      *
525      * @param in the java.io.Reader to read input from.
526      */
527     Lexer(java.io.Reader in) {
528         this.zzReader = in;
529     }
530
531     /**
532      * Creates a new scanner.
533      * There is also java.io.Reader version of this constructor.
534      *
535      * @param in the java.io.InputStream to read input from.
536      */
537     Lexer(java.io.InputStream in) {
538         this(new java.io.InputStreamReader(in));
539     }
540
541     /**
542      * Unpacks the compressed character translation table.
543      *
544      * @param packed the packed character translation table
545      * @return the unpacked character translation table
546      */
547     private static char[] zzUnpackCMap(String packed) {
548         char[] map = new char[0x10000];
549         int i = 0; /* index in packed string */
550         int j = 0; /* index in unpacked array */
551         while (i < 166) {
552             int count = packed.charAt(i++);
553             char value = packed.charAt(i++);
554             do map[j++] = value; while (--count > 0);
555         }
556         return map;
557     }
558
559
560     /**
561      * Refills the input buffer.
```



# MANUAL DE PRÁCTICAS



```
562      *
563      * @return      <code>false</code>, iff there was new input.
564      *
565      * @exception    java.io.IOException  if any I/O-Error occurs
566      */
567  private boolean zzRefill() throws java.io.IOException {
568
569      /* first: make room (if you can) */
570      if (zzStartRead > 0) {
571          System.arraycopy(zzBuffer, zzStartRead,
572                          zzBuffer, 0,
573                          zzEndRead-zzStartRead);
574
575          /* translate stored positions */
576          zzEndRead-= zzStartRead;
577          zzCurrentPos-= zzStartRead;
578          zzMarkedPos-= zzStartRead;
579          zzStartRead = 0;
580      }
581
582      /* is the buffer big enough? */
583      if (zzCurrentPos >= zzBuffer.length) {
584          /* if not: blow it up */
585          char newBuffer[] = new char[zzCurrentPos*2];
586          System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
587          zzBuffer = newBuffer;
588      }
589
590      /* finally: fill the buffer with new input */
591      int numRead = zzReader.read(zzBuffer, zzEndRead,
592                                  zzBuffer.length-zzEndRead);
593
594      if (numRead > 0) {
595          zzEndRead+= numRead;
596          return false;
597      }
598      // unlikely but not impossible: read 0 characters, but not at end of stream
599      if (numRead == 0) {
600          int c = zzReader.read();
601          if (c == -1) {
602              return true;
603          } else {
604              zzBuffer[zzEndRead++] = (char) c;
605              return false;
606          }
607      }
608
609      // numRead < 0
610      return true;
611  }
612
613
614  /**
615   * Closes the input stream.
616   */
617  public final void yclose() throws java.io.IOException {
618      zzAtEOF = true;           /* indicate end of file */
619      zzEndRead = zzStartRead; /* invalidate buffer */
620
621      if (zzReader != null)
622          zzReader.close();
623  }
624
625
626  /**
627   * Resets the scanner to read from a new input stream.
```



```
628     * Does not close the old reader.  
629     *  
630     * All internal variables are reset, the old input stream  
631     * <b>cannot</b> be reused (internal buffer is discarded and lost).  
632     * Lexical state is set to <tt>ZZ_INITIAL</tt>.  
633     *  
634     * @param reader    the new input stream  
635     */  
636     public final void yyreset(java.io.Reader reader) {  
637         zzReader = reader;  
638         zzAtBOL = true;  
639         zzAtEOF = false;  
640         zzEOFDone = false;  
641         zzEndRead = zzStartRead = 0;  
642         zzCurrentPos = zzMarkedPos = 0;  
643         yyline = yychar = yycolumn = 0;  
644         zzLexicalState = YYINITIAL;  
645     }  
646  
647  
648     /**  
649      * Returns the current lexical state.  
650      */  
651     public final int yystate() {  
652         return zzLexicalState;  
653     }  
654  
655  
656     /**  
657      * Enters a new lexical state  
658      *  
659      * @param newState the new lexical state  
660      */
```

```
661     public final void yybegin(int newState) {  
662         zzLexicalState = newState;  
663     }  
664  
665  
666     /**  
667      * Returns the text matched by the current regular expression.  
668      */  
669     public final String yytext() {  
670         return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );  
671     }  
672  
673  
674     /**  
675      * Returns the character at position <tt>pos</tt> from the  
676      * matched text.  
677      *  
678      * It is equivalent to yytext().charAt(pos), but faster  
679      *  
680      * @param pos the position of the character to fetch.  
681      *          A value from 0 to yylength()-1.  
682      *  
683      * @return the character at position pos  
684      */  
685     public final char yycharat(int pos) {  
686         return zzBuffer[zzStartRead+pos];  
687     }  
688  
689  
690     /**  
691      * Returns the length of the matched text region.  
692      */  
693     public final int yylength() {
```



```
694     return zzMarkedPos-zzStartRead;
695 }
696
697
698 /**
699 * Reports an error that occurred while scanning.
700 *
701 * In a wellformed scanner (no or only correct usage of
702 * yypushback(int) and a match-all fallback rule) this method
703 * will only be called with things that "Can't Possibly Happen".
704 * If this method is called, something is seriously wrong
705 * (e.g. a JFlex bug producing a faulty scanner etc.).
706 *
707 * Usual syntax/scanner level error handling should be done
708 * in error fallback rules.
709 *
710 * @param errorCode the code of the errormessage to display
711 */
712 private void zzscanError(int errorCode) {
713     String message;
714     try {
715         message = ZZ_ERROR_MSG[errorCode];
716     }
717     catch (ArrayIndexOutOfBoundsException e) {
718         message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
719     }
720
721     throw new Error(message);
722 }
723
724
725 /**
726 * Pushes the specified amount of characters back into the input stream.
727
728 * They will be read again by then next call of the scanning method
729 *
730 * @param number the number of characters to be read again.
731 *                This number must not be greater than yylength()!
732 */
733 public void yypushback(int number) {
734     if (number > yylength())
735         zzScanError(ZZ_PUSHBACK_2BIG);
736
737     zzMarkedPos -= number;
738 }
739
740
741 /**
742 * Resumes scanning until the next regular expression is matched,
743 * the end of input is encountered or an I/O-Error occurs.
744 *
745 * @return the next token
746 * @exception java.io.IOException if any I/O-Error occurs
747 */
748 public Tokens yylex() throws java.io.IOException {
749     int zzInput;
750     int zzAction;
751
752     // cached fields:
753     int zzCurrentPosL;
754     int zzMarkedPosL;
755     int zzEndReadL = zzEndRead;
756     char [] zzBufferL = zzBuffer;
757     char [] zzCMapL = ZZ_CMAP;
758
759     int [] zzTransL = ZZ_TRANS;
```



```
760 int [] zzRowMapL = ZZ_ROWMAP;
761 int [] zzAttrL = ZZ_ATTRIBUTE;
762
763 while (true) {
764     zzMarkedPosL = zzMarkedPos;
765
766     zzAction = -1;
767
768     zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
769
770     zzState = ZZ_LEXSTATE[zzLexicalState];
771
772     zzForAction: {
773         while (true) {
774
775             if (zzCurrentPosL < zzEndReadL)
776                 zzInput = zzBufferL[zzCurrentPosL++];
777             else if (zzAtEOF) {
778                 zzInput = YYEOF;
779                 break zzForAction;
780             }
781             else {
782                 // store back cached positions
783                 zzCurrentPos = zzCurrentPosL;
784                 zzMarkedPos = zzMarkedPosL;
785                 boolean eof = zzRefill();
786                 // get translated positions and possibly new buffer
787                 zzCurrentPosL = zzCurrentPos;
788                 zzMarkedPosL = zzMarkedPos;
789                 zzBufferL = zzBuffer;
790                 zzEndReadL = zzEndread;
791                 if (eof) {
792
793                     zzInput = YYEOF;
794                     break zzForAction;
795                 }
796                 else {
797                     zzInput = zzBufferL[zzCurrentPosL++];
798                 }
799             }
800             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
801             if (zzNext == -1) break zzForAction;
802             zzState = zzNext;
803
804             int zzAttributes = zzAttrL[zzState];
805             if ( (zzAttributes & 1) == 1 ) {
806                 zzAction = zzState;
807                 zzMarkedPosL = zzCurrentPosL;
808                 if ( (zzAttributes & 8) == 8 ) break zzForAction;
809             }
810
811         }
812     }
813
814     // store back cached position
815     zzMarkedPos = zzMarkedPosL;
816
817     switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
818         case 20:
819             { return parentDer; }
820         case 77: break;
821         case 21:
822             { return llaveDer; }
823         case 78: break;
824     }
825 }
```



# MANUAL DE PRÁCTICAS



```
826         case 10:
827             { return operadorModulo;
828         }
829         case 79: break;
830         case 32:
831             { lexeme=yytext(); return Identificador;
832         }
833         case 80: break;
834         case 22:
835             { return pesos;
836         }
837         case 81: break;
838         case 54:
839             { return cadena;
840         }
841         case 82: break;
842         case 69:
843             { return detener;
844         }
845         case 83: break;
846         case 31:
847             { return gato;
848         }
849         case 84: break;
850         case 3:
851             { return numero;
852         }
853         case 85: break;
854         case 2:
855             { return guionBajo;
856         }
857         case 86: break;
858         case 67:
```

```
859             { return arreglo;
860         }
861         case 87: break;
862         case 11:
863             { return operadorPotencia;
864         }
865         case 88: break;
866         case 72:
867             { return importar;
868         }
869         case 89: break;
870         case 34:
871             { lexeme=yytext(); return Reservadas;
872         }
873         case 90: break;
874         case 75:
875             { return funcionIntroducirDatos;
876         }
877         case 91: break;
878         case 36:
879             { return incremento;
880         }
881         case 92: break;
882         case 24:
883             { return corchetIzq;
884         }
885         case 93: break;
886         case 17:
887             { return operadorNo;
888         }
889         case 94: break;
890         case 4:
891             { /*Ignore*/
```



```
892 }  
893 case 95: break;  
894 case 8:  
895 { return operadorResta;  
896 }  
897 case 96: break;  
898 case 60:  
899 { return funcionLeer;  
900 }  
901 case 97: break;  
902 case 46:  
903 { return operadorPositivo;  
904 }  
905 case 98: break;  
906 case 14:  
907 { return menorQue;  
908 }  
909 case 99: break;  
910 case 5:  
911 { return letrasMin;  
912 }  
913 case 100: break;  
914 case 70:  
915 { return cicloWhile; }  
916 }  
917 case 101: break;  
918 case 53:  
919 { return valorBooleano;  
920 }  
921 case 102: break;  
922 case 65:  
923 { return cicloFor;  
924 }
```

```
925 case 103: break;  
926 case 42:  
927 { return inicioTexto;  
928 }  
929 case 104: break;  
930 case 66:  
931 { return clase;  
932 }  
933 case 105: break;  
934 case 1:  
935 { return ERROR;  
936 }  
937 case 106: break;  
938 case 19:  
939 { return dosPuntos;  
940 }  
941 case 107: break;  
942 case 28:  
943 { return comillaSimple;  
944 }  
945 case 108: break;  
946 case 76:  
947 { return predeterminado;  
948 }  
949 case 109: break;  
950 case 50:  
951 { return concatenar;  
952 }  
953 case 110: break;  
954 case 56:  
955 { return condicionalElse;  
956 }  
957 case 111: break;
```



```
958     case 39:  
959     { return asignacion;  
960     }  
961     case 112: break;  
962     case 44:  
963     { return saltoLinea;  
964     }  
965     case 113: break;  
966     case 63:  
967     { return cicloDo;  
968     }  
969     case 114: break;  
970     case 49:  
971     { return inicioComentarioMult;  
972     }  
973     case 115: break;  
974     case 37:  
975     { return decremento;  
976     }  
977     case 116: break;  
978     case 13:  
979     { return mayorQue;  
980     }  
981     case 117: break;  
982     case 64:  
983     { return operadorRaiz;  
984     }  
985     case 118: break;  
986     case 35:  
987     { return condicionalIf;  
988     }  
989     case 119: break;  
990     case 57:
```

```
991     { return funcionCase;  
992     }  
993     case 120: break;  
994     case 16:  
995     { return operadorO;  
996     }  
997     case 121: break;  
998     case 43:  
999     { return finalComentarioMult;  
1000    }  
1001    case 122: break;  
1002    case 74:  
1003    { return constante;  
1004    }  
1005    case 123: break;  
1006    case 25:  
1007    { return corchetDer;  
1008    }  
1009    case 124: break;  
1010    case 58:  
1011    { return valorFlotante;  
1012    }  
1013    case 125: break;  
1014    case 51:  
1015    { return comentarioLinea;  
1016    }  
1017    case 126: break;  
1018    case 47:  
1019    { return operadorNegativo;  
1020    }  
1021    case 127: break;  
1022    case 26:  
1023    { return puntoAcceso;
```



```
1024     }
1025     case 128: break;
1026     case 18:
1027     {
1028         return parentIzq;
1029     }
1030     case 129: break;
1031     case 23:
1032     {
1033         return llaveIzq;
1034     }
1035     case 130: break;
1036     case 40:
1037     {
1038         return finalTexto;
1039     }
1040     case 131: break;
1041     case 33:
1042     {
1043         lexeme=yytext(); return Numero;
1044     }
1045     case 132: break;
1046     case 48:
1047     {
1048         return menorIgual;
1049     }
1050     case 133: break;
1051     case 27:
1052     {
1053         return comillaDoble;
1054     }
1055     case 134: break;
1056     case 12:
1057     {
1058         return operadorSuma;
1059     }
1060     case 135: break;
1061     case 136: break;
1062     case 45:
1063     {
1064         return mayorIgual;
1065     }
1066     case 137: break;
1067     case 62:
1068     {
1069         return funcionSalirCase;
1070     }
1071     case 138: break;
1072     case 68:
1073     {
1074         return funcion;
1075     }
1076     case 139: break;
1077     case 29:
1078     {
1079         return diagonalInvertido;
1080     }
1081     case 140: break;
1082     case 30:
1083     {
1084         return signoAdmiracionAbre;
1085     }
1086     case 141: break;
1087     case 71:
1088     {
1089         return variable;
1090     }
1091     case 142: break;
1092     case 73:
1093     {
1094         return imprimir;
1095     }
1096     case 143: break;
1097     case 15:
1098     {
1099         return operadorY;
1100     }
1101     case 144: break;
```

```
1102     case 145: break;
1103     case 146:
1104     {
1105         return operadorX;
1106     }
1107     case 147: break;
1108     case 148:
1109     {
1110         return operadorMultiplicacion;
1111     }
1112     case 149: break;
1113     case 150:
1114     {
1115         return operadorDivision;
1116     }
1117     case 151: break;
1118     case 152:
1119     {
1120         return operadorModulo;
1121     }
1122     case 153: break;
1123     case 154:
1124     {
1125         return operadorPotencia;
1126     }
1127     case 155: break;
1128     case 156:
1129     {
1130         return operadorComplemento;
1131     }
1132     case 157: break;
1133     case 158:
1134     {
1135         return operadorNegacion;
1136     }
1137     case 159: break;
1138     case 160:
1139     {
1140         return operadorAsignacion;
1141     }
1142     case 161: break;
1143     case 162:
1144     {
1145         return operadorRelacional;
1146     }
1147     case 163: break;
1148     case 164:
1149     {
1150         return operadorLogico;
1151     }
1152     case 165: break;
1153     case 166:
1154     {
1155         return operadorAritmetico;
1156     }
1157     case 167: break;
1158     case 168:
1159     {
1160         return operadorComparacion;
1161     }
1162     case 169: break;
1163     case 170:
1164     {
1165         return operadorRelacional;
1166     }
1167     case 171: break;
1168     case 172:
1169     {
1170         return operadorLogico;
1171     }
1172     case 173: break;
1173     case 174:
1174     {
1175         return operadorAritmetico;
1176     }
1177     case 178: break;
1178     case 179:
1179     {
1180         return operadorComparacion;
1181     }
1182     case 180: break;
1183     case 181:
1184     {
1185         return operadorRelacional;
1186     }
1187     case 188: break;
1188     case 189:
1189     {
1190         return operadorLogico;
1191     }
1192     case 190: break;
1193     case 191:
1194     {
1195         return operadorAritmetico;
1196     }
1197     case 198: break;
1198     case 199:
1199     {
1200         return operadorComparacion;
1201     }
1202     case 200: break;
1203     case 201:
1204     {
1205         return operadorRelacional;
1206     }
1207     case 208: break;
1208     case 209:
1209     {
1210         return operadorLogico;
1211     }
1212     case 209: break;
1213     case 210:
1214     {
1215         return operadorAritmetico;
1216     }
1217     case 218: break;
1218     case 219:
1219     {
1220         return operadorComparacion;
1221     }
1222     case 220: break;
1223     case 221:
1224     {
1225         return operadorRelacional;
1226     }
1227     case 228: break;
1228     case 229:
1229     {
1230         return operadorLogico;
1231     }
1232     case 229: break;
1233     case 230:
1234     {
1235         return operadorAritmetico;
1236     }
1237     case 238: break;
1238     case 239:
1239     {
1240         return operadorComparacion;
1241     }
1242     case 240: break;
1243     case 241:
1244     {
1245         return operadorRelacional;
1246     }
1247     case 248: break;
1248     case 249:
1249     {
1250         return operadorLogico;
1251     }
1252     case 249: break;
1253     case 250:
1254     {
1255         return operadorAritmetico;
1256     }
1257     case 258: break;
1258     case 259:
1259     {
1260         return operadorComparacion;
1261     }
1262     case 260: break;
1263     case 261:
1264     {
1265         return operadorRelacional;
1266     }
1267     case 268: break;
1268     case 269:
1269     {
1270         return operadorLogico;
1271     }
1272     case 269: break;
1273     case 270:
1274     {
1275         return operadorAritmetico;
1276     }
1277     case 278: break;
1278     case 279:
1279     {
1280         return operadorComparacion;
1281     }
1282     case 280: break;
1283     case 281:
1284     {
1285         return operadorRelacional;
1286     }
1287     case 288: break;
1288     case 289:
1289     {
1290         return operadorLogico;
1291     }
1292     case 289: break;
1293     case 290:
1294     {
1295         return operadorAritmetico;
1296     }
1297     case 298: break;
1298     case 299:
1299     {
1300         return operadorComparacion;
1301     }
1302     case 300: break;
1303     case 301:
1304     {
1305         return operadorRelacional;
1306     }
1307     case 308: break;
1308     case 309:
1309     {
1310         return operadorLogico;
1311     }
1312     case 309: break;
1313     case 310:
1314     {
1315         return operadorAritmetico;
1316     }
1317     case 318: break;
1318     case 319:
1319     {
1320         return operadorComparacion;
1321     }
1322     case 320: break;
1323     case 321:
1324     {
1325         return operadorRelacional;
1326     }
1327     case 328: break;
1328     case 329:
1329     {
1330         return operadorLogico;
1331     }
1332     case 329: break;
1333     case 330:
1334     {
1335         return operadorAritmetico;
1336     }
1337     case 338: break;
1338     case 339:
1339     {
1340         return operadorComparacion;
1341     }
1342     case 340: break;
1343     case 341:
1344     {
1345         return operadorRelacional;
1346     }
1347     case 348: break;
1348     case 349:
1349     {
1350         return operadorLogico;
1351     }
1352     case 349: break;
1353     case 350:
1354     {
1355         return operadorAritmetico;
1356     }
1357     case 358: break;
1358     case 359:
1359     {
1360         return operadorComparacion;
1361     }
1362     case 360: break;
1363     case 361:
1364     {
1365         return operadorRelacional;
1366     }
1367     case 368: break;
1368     case 369:
1369     {
1370         return operadorLogico;
1371     }
1372     case 369: break;
1373     case 370:
1374     {
1375         return operadorAritmetico;
1376     }
1377     case 378: break;
1378     case 379:
1379     {
1380         return operadorComparacion;
1381     }
1382     case 380: break;
1383     case 381:
1384     {
1385         return operadorRelacional;
1386     }
1387     case 388: break;
1388     case 389:
1389     {
1390         return operadorLogico;
1391     }
1392     case 389: break;
1393     case 390:
1394     {
1395         return operadorAritmetico;
1396     }
1397     case 398: break;
1398     case 399:
1399     {
1400         return operadorComparacion;
1401     }
1402     case 400: break;
1403     case 401:
1404     {
1405         return operadorRelacional;
1406     }
1407     case 408: break;
1408     case 409:
1409     {
1410         return operadorLogico;
1411     }
1412     case 409: break;
1413     case 410:
1414     {
1415         return operadorAritmetico;
1416     }
1417     case 418: break;
1418     case 419:
1419     {
1420         return operadorComparacion;
1421     }
1422     case 420: break;
1423     case 421:
1424     {
1425         return operadorRelacional;
1426     }
1427     case 428: break;
1428     case 429:
1429     {
1430         return operadorLogico;
1431     }
1432     case 429: break;
1433     case 430:
1434     {
1435         return operadorAritmetico;
1436     }
1437     case 438: break;
1438     case 439:
1439     {
1440         return operadorComparacion;
1441     }
1442     case 440: break;
1443     case 441:
1444     {
1445         return operadorRelacional;
1446     }
1447     case 448: break;
1448     case 449:
1449     {
1450         return operadorLogico;
1451     }
1452     case 449: break;
1453     case 450:
1454     {
1455         return operadorAritmetico;
1456     }
1457     case 458: break;
1458     case 459:
1459     {
1460         return operadorComparacion;
1461     }
1462     case 460: break;
1463     case 464:
1464     {
1465         return operadorRelacional;
1466     }
1467     case 468: break;
1468     case 469:
1469     {
1470         return operadorLogico;
1471     }
1472     case 469: break;
1473     case 470:
1474     {
1475         return operadorAritmetico;
1476     }
1477     case 478: break;
1478     case 479:
1479     {
1480         return operadorComparacion;
1481     }
1482     case 480: break;
1483     case 481:
1484     {
1485         return operadorRelacional;
1486     }
1487     case 488: break;
1488     case 489:
1489     {
1490         return operadorLogico;
1491     }
1492     case 489: break;
1493     case 490:
1494     {
1495         return operadorAritmetico;
1496     }
1497     case 498: break;
1498     case 499:
1499     {
1500         return operadorComparacion;
1501     }
1502     case 500: break;
1503     case 501:
1504     {
1505         return operadorRelacional;
1506     }
1507     case 508: break;
1508     case 509:
1509     {
1510         return operadorLogico;
1511     }
1512     case 509: break;
1513     case 510:
1514     {
1515         return operadorAritmetico;
1516     }
1517     case 518: break;
1518     case 519:
1519     {
1520         return operadorComparacion;
1521     }
1522     case 520: break;
1523     case 521:
1524     {
1525         return operadorRelacional;
1526     }
1527     case 528: break;
1528     case 529:
1529     {
1530         return operadorLogico;
1531     }
1532     case 529: break;
1533     case 530:
1534     {
1535         return operadorAritmetico;
1536     }
1537     case 538: break;
1538     case 539:
1539     {
1540         return operadorComparacion;
1541     }
1542     case 540: break;
1543     case 541:
1544     {
1545         return operadorRelacional;
1546     }
1547     case 548: break;
1548     case 549:
1549     {
1550         return operadorLogico;
1551     }
1552     case 549: break;
1553     case 550:
1554     {
1555         return operadorAritmetico;
1556     }
1557     case 558: break;
1558     case 559:
1559     {
1560         return operadorComparacion;
1561     }
1562     case 560: break;
1563     case 564:
1564     {
1565         return operadorRelacional;
1566     }
1567     case 568: break;
1568     case 569:
1569     {
1570         return operadorLogico;
1571     }
1572     case 569: break;
1573     case 570:
1574     {
1575         return operadorAritmetico;
1576     }
1577     case 578: break;
1578     case 579:
1579     {
1580         return operadorComparacion;
1581     }
1582     case 580: break;
1583     case 581:
1584     {
1585         return operadorRelacional;
1586     }
1587     case 588: break;
1588     case 589:
1589     {
1590         return operadorLogico;
1591     }
1592     case 589: break;
1593     case 590:
1594     {
1595         return operadorAritmetico;
1596     }
1597     case 598: break;
1598     case 599:
1599     {
1600         return operadorComparacion;
1601     }
1602     case 600: break;
1603     case 601:
1604     {
1605         return operadorRelacional;
1606     }
1607     case 608: break;
1608     case 609:
1609     {
1610         return operadorLogico;
1611     }
1612     case 609: break;
1613     case 610:
1614     {
1615         return operadorAritmetico;
1616     }
1617     case 618: break;
1618     case 619:
1619     {
1620         return operadorComparacion;
1621     }
1622     case 620: break;
1623     case 621:
1624     {
1625         return operadorRelacional;
1626     }
1627     case 628: break;
1628     case 629:
1629     {
1630         return operadorLogico;
1631     }
1632     case 629: break;
1633     case 630:
1634     {
1635         return operadorAritmetico;
1636     }
1637     case 638: break;
1638     case 639:
1639     {
1640         return operadorComparacion;
1641     }
1642     case 640: break;
1643     case 641:
1644     {
1645         return operadorRelacional;
1646     }
1647     case 648: break;
1648     case 649:
1649     {
1650         return operadorLogico;
1651     }
1652     case 649: break;
1653     case 650:
1654     {
1655         return operadorAritmetico;
1656     }
1657     case 658: break;
1658     case 659:
1659     {
1660         return operadorComparacion;
1661     }
1662     case 660: break;
1663     case 664:
1664     {
1665         return operadorRelacional;
1666     }
1667     case 668: break;
1668     case 669:
1669     {
1670         return operadorLogico;
1671     }
1672     case 669: break;
1673     case 670:
1674     {
1675         return operadorAritmetico;
1676     }
1677     case 678: break;
1678     case 679:
1679     {
1680         return operadorComparacion;
1681     }
1682     case 680: break;
1683     case 681:
1684     {
1685         return operadorRelacional;
1686     }
1687     case 688: break;
1688     case 689:
1689     {
1690         return operadorLogico;
1691     }
1692     case 689: break;
1693     case 690:
1694     {
1695         return operadorAritmetico;
1696     }
1697     case 698: break;
1698     case 699:
1699     {
1700         return operadorComparacion;
1701     }
1702     case 700: break;
1703     case 701:
1704     {
1705         return operadorRelacional;
1706     }
1707     case 708: break;
1708     case 709:
1709     {
1710         return operadorLogico;
1711     }
1712     case 709: break;
1713     case 710:
1714     {
1715         return operadorAritmetico;
1716     }
1717     case 718: break;
1718     case 719:
1719     {
1720         return operadorComparacion;
1721     }
1722     case 720: break;
1723     case 721:
1724     {
1725         return operadorRelacional;
1726     }
1727     case 728: break;
1728     case 729:
1729     {
1730         return operadorLogico;
1731     }
1732     case 729: break;
1733     case 730:
1734     {
1735         return operadorAritmetico;
1736     }
1737     case 738: break;
1738     case 739:
1739     {
1740         return operadorComparacion;
1741     }
1742     case 740: break;
1743     case 741:
1744     {
1745         return operadorRelacional;
1746     }
1747     case 748: break;
1748     case 749:
1749     {
1750         return operadorLogico;
1751     }
1752     case 749: break;
1753     case 750:
1754     {
1755         return operadorAritmetico;
1756     }
1757     case 758: break;
1758     case 759:
1759     {
1760         return operadorComparacion;
1761     }
1762     case 760: break;
1763     case 764:
1764     {
1765         return operadorRelacional;
1766     }
1767     case 768: break;
1768     case 769:
1769     {
1770         return operadorLogico;
1771     }
1772     case 769: break;
1773     case 770:
1774     {
1775         return operadorAritmetico;
1776     }
1777     case 778: break;
1778     case 779:
1779     {
1780         return operadorComparacion;
1781     }
1782     case 780: break;
1783     case 781:
1784     {
1785         return operadorRelacional;
1786     }
1787     case 788: break;
1788     case 789:
1789     {
1790         return operadorLogico;
1791     }
1792     case 789: break;
1793     case 790:
1794     {
1795         return operadorAritmetico;
1796     }
1797     case 798: break;
1798     case 799:
1799     {
1800         return operadorComparacion;
1801     }
1802     case 800: break;
1803     case 801:
1804     {
1805         return operadorRelacional;
1806     }
1807     case 808: break;
1808     case 809:
1809     {
1810         return operadorLogico;
1811     }
1812     case 809: break;
1813     case 810:
1814     {
1815         return operadorAritmetico;
1816     }
1817     case 818: break;
1818     case 819:
1819     {
1820         return operadorComparacion;
1821     }
1822     case 820: break;
1823     case 821:
1824     {
1825         return operadorRelacional;
1826     }
1827     case 828: break;
1828     case 829:
1829     {
1830         return operadorLogico;
1831     }
1832     case 829: break;
1833     case 830:
1834     {
1835         return operadorAritmetico;
1836     }
1837     case 838: break;
1838     case 839:
1839     {
1840         return operadorComparacion;
1841     }
1842     case 840: break;
1843     case 841:
1844     {
1845         return operadorRelacional;
1846     }
1847     case 848: break;
1848     case 849:
1849     {
1850         return operadorLogico;
1851     }
1852     case 849: break;
1853     case 850:
1854     {
1855         return operadorAritmetico;
1856     }
1857     case 858: break;
1858     case 859:
1859     {
1860         return operadorComparacion;
1861     }
1862     case 860: break;
1863     case 864:
1864     {
1865         return operadorRelacional;
1866     }
1867     case 868: break;
1868     case 869:
1869     {
1870         return operadorLogico;
1871     }
1872     case 869: break;
1873     case 870:
1874     {
1875         return operadorAritmetico;
1876     }
1877     case 878: break;
1878     case 879:
1879     {
1880         return operadorComparacion;
1881     }
1882     case 880: break;
1883     case 884:
1884     {
1885         return operadorRelacional;
1886     }
1887     case 888: break;
1888     case 889:
1889     {
1890         return operadorLogico;
1891     }
1892     case 889: break;
1893     case 890:
1894     {
1895         return operadorAritmetico;
1896     }
1897     case 898: break;
1898     case 899:
1899     {
1900         return operadorComparacion;
1901     }
1902     case 1900: break;
1903     case 1901:
1904     {
1905         return operadorRelacional;
1906     }
1907     case 1908: break;
1908     case 1909:
1909     {
1910         return operadorLogico;
1911     }
1912     case 1911: break;
1913     case 1914:
1914     {
1915         return operadorAritmetico;
1916     }
1917     case 1918: break;
1918     case 1919:
1919     {
1920         return operadorComparacion;
1921     }
1922     case 1923: break;
1923     case 1924:
1924     {
1925         return operadorRelacional;
1926     }
1927     case 1928: break;
1928     case 1929:
1929     {
1930         return operadorLogico;
1931     }
1932     case 1931: break;
1933     case 1934:
1934     {
1935         return operadorAritmetico;
1936     }
1937     case 1938: break;
1938     case 1939:
1939     {
1940         return operadorComparacion;
1941     }
1942     case 1943: break;
1943     case 1944:
1944     {
1945         return operadorRelacional;
1946     }
1947     case 1948: break;
1948     case 1949:
1949     {
1950         return operadorLogico;
1951     }
1952     case 1951: break;
1953     case 1954:
1954     {
1955         return operadorAritmetico;
1956     }
1957     case 1958: break;
1958     case 1959:
1959     {
1960         return operadorComparacion;
1961     }
1962     case 1963: break;
1963     case 1964:
1964     {
1965         return operadorRelacional;
1966     }
1967     case 1968: break;
1968     case 1969:
1969     {
1970         return operadorLogico;
1971     }
1972     case 1971: break;
1973     case 1974:
1974     {
1975         return operadorAritmetico;
1976     }
1977     case 1978: break;
1978     case 1979:
1979     {
1980         return operadorComparacion;
1981     }
1982     case 1983: break;
1983     case 1984:
1984     {
1985         return operadorRelacional;
1986     }
1987     case 1988: break;
1988     case 1989:
1989     {
1990         return operadorLogico;
1991     }
1992     case 1991: break;
1993     case 1994:
1994     {
1995         return operadorAritmetico;
1996     }
1997     case 1998: break;
1998     case 1999:
1999     {
2000         return operadorComparacion;
2001     }
2002     case 2003: break;
2003     case 2004:
2004     {
2005         return operadorRelacional;
2006     }
2007     case 2008: break;
2008     case 2009:
2009     {
2010         return operadorLogico;
2011     }
2012     case 2011: break;
2013     case 2014:
2014     {
2015         return operadorAritmetico;
2016     }
2017     case 2018: break;
2018     case 2019:
2019     {
2020         return operadorComparacion;
2021     }
2022     case 2023: break;
2023     case 2024:
2024     {
2025         return operadorRelacional;
2026     }
2027     case 2028: break;
2028     case 2029:
2029     {
2030         return operadorLogico;
2031     }
2032     case 2031: break;
2033     case 2034:
2034     {
2035         return operadorAritmetico;
2036     }
2037     case 2038: break;
2038     case 2039:
2039     {
2040         return operadorComparacion;
2041     }
2042     case 2043: break;
2043     case 2044:
2044     {
2045         return operadorRelacional;
2046     }
2047     case 2048: break;
2048     case 2049:
2049     {
2050         return operadorLogico;
2051     }
2052     case 2051: break;
2053     case 2054:
2054     {
2055         return operadorAritmetico;
2056     }
2057     case 2058: break;
2058     case 2059:
2059     {
2060         return operadorComparacion;
2061     }
2062     case 2063: break;
2063     case 2064:
2064     {
2065         return operadorRelacional;
2066     }
2067     case 2068: break;
2068     case 2069:
2069     {
2070         return operadorLogico;
2071     }
2072     case 2071: break;
2073     case 2074:
2074     {
2075         return operadorAritmetico;
2076     }
2077     case 2078: break;
2078     case 2079:
2079     {
2080         return operadorComparacion;
2081     }
2082     case 2083: break;
2083     case 2084:
2084     {
2085         return operadorRelacional;
2086     }
2087     case 2088: break;
2088     case 2089:
2089     {
2090         return operadorLogico;
2091     }
2092     case 2091: break;
2093     case 2094:
2094     {
2095         return operadorAritmetico;
2096     }
2097     case 2098: break;
2098     case 2099:
2099     {
2100         return operadorComparacion;
2101     }
2102     case 2103: break;
2103     case 2104:
2104     {
2105         return operadorRelacional;
2106     }
2107     case 2108: break;
2108     case 2109:
2109     {
2110         return operadorLogico;
2111     }
2112     case 2111: break;
2113     case 2114:
2114     {
2115         return operadorAritmetico;
2116     }
2117     case 2118: break;
2118     case 2119:
2119     {
2120         return operadorComparacion;
2121     }
2122     case 2123: break;
2123     case 2124:
2124     {
2125         return operadorRelacional;
2126     }
2127     case 2128: break;
2128     case 2129:
2129     {
2130         return operadorLogico;
2131     }
2132     case 2131: break;
2133     case 2134:
2134     {
2135         return operadorAritmetico;
2136     }
2137     case 2138: break;
2138     case 2139:
2139     {
2140         return operadorComparacion;
2141     }
2142     case 2143: break;
2143     case 2144:
2144     {
2145         return operadorRelacional;
2146     }
2147     case 2148: break;
2148     case 2149:
2149     {
2150         return operadorLogico;
2151     }
2152     case 2151: break;
2153     case 2154:
2154     {
2155         return operadorAritmetico;
2156     }
2157     case 2158: break;
2158     case 2159:
2159     {
2160         return operadorComparacion;
2161     }
2162     case 2163: break;
2163     case 2164:
2164     {
2165         return operadorRelacional;
2166     }
2167     case 2168: break;
2168     case 2169:
2169     {
2170         return operadorLogico;
2171     }
2172     case 2171: break;
2173     case 2174:
2174     {
2175         return operadorAritmetico;
2176     }
2177     case 2178: break;
2178     case 2179:
2179     {
2180         return operadorComparacion;
2181     }
2182     case 2183: break;
2183     case 2184:
2184     {
2185         return operadorRelacional;
2186     }
2187     case 2188: break;
2188     case 2189:
2189     {
2190         return
```



```
1090      case 59:
1091      { return carácter;
1092      }
1093      case 145: break;
1094      case 38:
1095      { return comparacionIgualdad;
1096      }
1097      case 146: break;
1098      case 41:
1099      { return pi;
1100      }
1101      case 147: break;
1102      case 6:
1103      { return operadorDivision;
1104      }
1105      case 148: break;
1106      case 9:
1107      { return operadorMultiplicacion;
1108      }
1109      case 149: break;
1110      case 61:
1111      { return funcionSwitch;
1112      }
1113      case 150: break;
1114      case 52:
1115      { return valorEntero;
1116      }
1117      case 151: break;
1118      case 55:
1119      { return finLinea;
1120      }
1121      case 152: break;
1122      default:
```

```
1123      if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
1124          zzAtEOF = true;
1125          return null;
1126      }
1127      else {
1128          zzScanError(ZZ_NO_MATCH);
1129      }
1130  }
1131 }
1132 }
1133
1134
1135 }
1136 }
```

## Archivo Tokens.java

El archivo Tokens.java es una pieza fundamental en la implementación de este analizador léxico, ya que define una enumeración que clasifica los diferentes tipos de tokens que el analizador puede identificar en el código fuente. Los tokens representan las unidades léxicas básicas del lenguaje, como palabras reservadas, operadores, identificadores, números y símbolos especiales, así como elementos estructurales como llaves, paréntesis y comentarios. Además, incluye un token genérico para errores, lo que permite gestionar y reportar entradas no reconocidas.

Este archivo actúa como una referencia central para el análisis y clasificación de los elementos léxicos, estableciendo una relación clara entre el código fuente y los componentes del lenguaje que se está interpretando o compilando. Esta enumeración facilita la implementación de reglas de análisis, mejora la legibilidad del código y contribuye a mantener una estructura ordenada y extensible para futuras actualizaciones o ampliaciones del lenguaje soportado por el analizador.



## Código Realizado:

```
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 codigo;  
7  
8  
9 public enum Tokens {  
10    Linea,  
11    Reservadas,  
12    operadorSuma,  
13    operadorResta,  
14    operadorMultiplicacion,  
15    operadorDivision,  
16    operadorModulo,  
17    operadorPotencia,  
18    operadorRaiz,  
19    asignacion,  
20    incremento,  
21    decremento,  
22    operadorPositivo,  
23    operadorNegativo,  
24    comparacionIgualdad,  
25    mayorQue,  
26    menorQue,  
27    mayorIgual,  
28    menorIgual,  
29    operadorY,  
30    operadorO,  
31    operadorNo,  
32    letrasMin,  
33    letrasMay,  
34    numero,  
35    inicioComentarioMult,  
36    finalComentarioMult,  
37    comentarioLinea,  
38    valorEntero,  
39    valorFlotante,  
40    valorBooleano,  
41    cadena,  
42    caracter,  
43    finLinea,  
44  
45    inicioTexto,  
46    finalTexto,  
47    parentIzq,  
48    parentDer,  
49    llaveIzq,  
50    llaveDer,  
51    corcheteIzq,  
52    corcheteDer,  
53    concatenar,  
54    arreglo,  
55  
56    puntoDecimal,  
57    dosPuntos,  
58    espacio,  
59    tabulacion,  
60    comillaDoble,  
61    comillaSimple,  
62    coma,  
63    guionBajo,  
64    guionMedio,  
65    diagonalInvertido,  
66    diagonal ,
```



```
67 signoAdmirationAbre,
68 signoAdmirationCierra,
69 gato,
70 pesos,
71 condicionalIf,
72 condicionalElse,
73 cicloFor,
74 cicloWhile,
75 cicloDo,
76 funcionesSwitch,
77 funcionCase,
78 funcionesSalirCase,
79 predeterminado,
80 detener,
81 constante,
82 variable,
83 funcion,
84 clase,
85 imprimir,
86 importar,
87 funcionLeer,
88 funcionIntroducirDatos,
89 pi,
90 euler,
91 Numero,
92 Identificador,
93     ERROR
94 }
```

### Archivo LexerCup.flex

El archivo LexerCup.flex es un componente esencial de un analizador léxico, que forma parte del proceso de compilación o interpretación de un lenguaje de programación. Utiliza JFlex, una herramienta de generación de analizadores léxicos, y está diseñado para convertir una secuencia de caracteres en una secuencia de tokens, los cuales son identificados según expresiones regulares definidas en el archivo. Cada token tiene una categoría (como identificador, número, operador, palabra reservada, etc.) y se asocia a información adicional, como la línea y la columna del código fuente. Este lexer es la primera etapa en el análisis de un programa, y su propósito es dividir el código en partes manejables que luego serán procesadas por un analizador sintáctico, como Java CUP.

La importancia de este archivo radica en su capacidad para transformar un código fuente en tokens estructurados que permiten al analizador sintáctico y semántico entender y procesar correctamente el lenguaje de programación. Sin un lexer como el definido en LexerCup.flex, el análisis del código sería prácticamente imposible, ya que no habría una forma clara de identificar las unidades básicas del lenguaje, el lexer no solo facilita la traducción del código fuente, sino que también contribuye a una correcta ejecución del proceso de compilación o interpretación, mejorando la eficiencia y precisión del sistema en su conjunto.



## Código Realizado:

```
1 package codigo;
2 import java_cup.runtime.Symbol;
3 /**
4 %class LexerCup
5 %type java_cup.runtime.Symbol
6 %cup
7 %full
8 %line
9 %char
10 L=[a-zA-Z_]+
11 D=[0-9]+
12 espacio=[ ,\t,\r,\n]+
13 ^{
14     private Symbol symbol(int type, Object value){
15         return new Symbol(type, yyline, yycolumn, value);
16     }
17     private Symbol symbol(int type){
18         return new Symbol(type, yyline, yycolumn);
19     }
20 }
21 /**
22 /* Espacios en blanco */
23 {espacio} {/*Ignore*/}
24
25 /* Comentarios */
26 /**/* /*Ignore*/
27 /* Comillas */
28 "\" {return new Symbol(sym.comillaDoble, yychar, yyline, yytext());}
29 '\'' {return new Symbol(sym.comillaSimple, yychar, yyline, yytext());}
30
31 /* Tipos de datos */
32 "Ent" {return new Symbol(sym.valorEntero, yychar, yyline, yytext());}
33
34 "Flot" {return new Symbol(sym.valorFlotante, yychar, yyline, yytext());}
35 "Boo" {return new Symbol(sym.valorBooleano, yychar, yyline, yytext());}
36 "Cad" {return new Symbol(sym.cadena, yychar, yyline, yytext());}
37 "Cars" {return new Symbol(sym.carácter, yychar, yyline, yytext());}
38
39 /* Palabras reservadas */
40 "si" {return new Symbol(sym.condicionalIf, yychar, yyline, yytext());}
41 "siNo" {return new Symbol(sym.condicionalElse, yychar, yyline, yytext());}
42 "ciclo" {return new Symbol(sym.cicloFor, yychar, yyline, yytext());}
43 "mientras" {return new Symbol(sym.cicloWhile, yychar, yyline, yytext());}
44 "hacer" {return new Symbol(sym.cicloDo, yychar, yyline, yytext());}
45 "segun" {return new Symbol(sym.funcionSwitch, yychar, yyline, yytext());}
46 "caso" {return new Symbol(sym.funcionCase, yychar, yyline, yytext());}
47 "salir" {return new Symbol(sym.funcionSalirCase, yychar, yyline, yytext());}
48 "predeterminado" {return new Symbol(sym.predeterminado, yychar, yyline, yytext());}
49 "Detener" {return new Symbol(sym.detener, yychar, yyline, yytext());}
50 "Constante" {return new Symbol(sym.constante, yychar, yyline, yytext());}
51 "Variable" {return new Symbol(sym.variable, yychar, yyline, yytext());}
52 "funcion" {return new Symbol(sym.funcion, yychar, yyline, yytext());}
53 "Clase" {return new Symbol(sym.clase, yychar, yyline, yytext());}
54 "Imprimir" {return new Symbol(sym.imprimir, yychar, yyline, yytext());}
55 "Leer" {return new Symbol(sym.funcionLeer, yychar, yyline, yytext());}
56 "IntroducirD" {return new Symbol(sym.funcionIntroducirDatos, yychar, yyline, yytext());}
57
58 /* Operadores matemáticos */
59 "+" {return new Symbol(sym.operadorSuma, yychar, yyline, yytext());}
60 "-" {return new Symbol(sym.operadorResta, yychar, yyline, yytext());}
61 "*" {return new Symbol(sym.operadorMultiplicacion, yychar, yyline, yytext());}
62 "/" {return new Symbol(sym.operadorDivision, yychar, yyline, yytext());}
63 "%" {return new Symbol(sym.operadorModulo, yychar, yyline, yytext());}
64 "^" {return new Symbol(sym.operadorPotencia, yychar, yyline, yytext());}
65 "Rcuad" {return new Symbol(sym.operadorRaiz, yychar, yyline, yytext());}
```



```
67  /* Operadores de asignación y comparación */
68  ">=" {return new Symbol(sym.asignacion, yychar, yyline, yytext());}
69  "==" {return new Symbol(sym.comparacionIgualdad, yychar, yyline, yytext());}
70  ">" {return new Symbol(sym.mayorQue, yychar, yyline, yytext());}
71  "<" {return new Symbol(sym.menorQue, yychar, yyline, yytext());}
72  ">=<" {return new Symbol(sym.mayorIgual, yychar, yyline, yytext());}
73  "<=>" {return new Symbol(sym.menorIgual, yychar, yyline, yytext());}
74
75  /* Operadores lógicos */
76  "&" {return new Symbol(sym.operadorY, yychar, yyline, yytext());}
77  "!" {return new Symbol(sym.operador), yychar, yyline, yytext());}
78  "!=" {return new Symbol(sym.operadorNo, yychar, yyline, yytext());}
79
80  /* Operadores incrementales */
81  "++" {return new Symbol(sym.increemento, yychar, yyline, yytext());}
82  "--" {return new Symbol(sym.decreemento, yychar, yyline, yytext());}
83
84  /* Caracteres especiales */
85  "(" {return new Symbol(sym.parentIzq, yychar, yyline, yytext());}
86  ")" {return new Symbol(sym.parentDer, yychar, yyline, yytext());}
87  "{" {return new Symbol(sym.llaveIzq, yychar, yyline, yytext());}
88  ")" {return new Symbol(sym.llaveDer, yychar, yyline, yytext());}
89  "[" {return new Symbol(sym.corchetIzq, yychar, yyline, yytext());}
90  "]" {return new Symbol(sym.corchetDer, yychar, yyline, yytext());}
91  ";" {return new Symbol(sym.finLinea, yychar, yyline, yytext());}
92
93  "<<" {return new Symbol(sym.inicioTexto, yychar, yyline, yytext());}
94  ">>" {return new Symbol(sym.finalTexto, yychar, yyline, yytext());}
95
96  "." {return new Symbol(sym.puntoDecimal, yychar, yyline, yytext());}
97  ":" {return new Symbol(sym.dosPuntos, yychar, yyline, yytext());}
98
99  /* Identificadores y números */
100 /* Identificadores y números */
101 "[L]([L]|{D})* {return new Symbol(sym.Identificador, yychar, yyline, yytext());}
102 ("-({D}+)"|{D}+ {return new Symbol(sym.Numero, yychar, yyline, yytext());}
103
104 /* Errores */
105 . {return new Symbol(sym.ERROR, yychar, yyline, yytext());}
```

## Archivo LexerCup.java (Generado automáticamente)

El archivo LexerCup.java es un escáner generado automáticamente por JFlex, una herramienta de análisis léxico. Este archivo convierte un flujo de texto en una secuencia de tokens, que son las unidades léxicas que comprenden el lenguaje a analizar. El propósito de este escáner es facilitar la tarea de la primera etapa de procesamiento en compiladores y analizadores de lenguajes, permitiendo identificar componentes básicos como palabras clave, operadores, identificadores y otros símbolos dentro del código fuente.

Este archivo tiene la capacidad para manejar entradas de texto de manera eficiente, utilizando un autómata finito determinista (DFA) para identificar patrones en el texto. Esto permite que el código sea optimizado en términos de velocidad de ejecución, siendo capaz de realizar un análisis léxico de manera rápida y confiable. Además, el archivo se genera automáticamente a partir de una especificación que describe los tokens del lenguaje, lo que facilita la construcción de analizadores léxicos sin necesidad de escribir manualmente el código complejo para cada patrón.



## Código Generado:

```
1 | 1 /* The following code was generated by JFlex 1.4.3 on 31/12/24, 15:24 */
2 |
3 | package codigo;
4 | import java_cup.runtime.Symbol;
5 |
6 | /**
7 | * This class is a scanner generated by
8 | * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9 | * on 31/12/24, 15:24 from the specification file
10 | * <tt>src/codigo/LexerCup.flex</tt>
11 | */
12 | class LexerCup implements java_cup.runtime.Scanner {
13 |
14 | /**
15 | * This character denotes the end of file
16 | */
17 | public static final int YYEOF = -1;
18 |
19 | /**
20 | * initial size of the lookahead buffer
21 | */
22 | private static final int ZZ_BUFSIZE = 16384;
23 |
24 | /**
25 | * ZZ_LEXSTATE[l] is the state in the DFA for the lexical state l
26 | * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
27 | * at the beginning of a line
28 | * l is of the form l = 2*k, k a non negative integer
29 | */
30 | private static final int ZZ_LEXSTATE[] = {
31 |     0, 0
32 | };
33 |
34 | /**
35 | * Translates characters to character classes
36 | */
37 | private static final char [] ZZ_CMAP = {
38 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 5, 0, 0, 0, 3, 0, 0,
39 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
40 |     3, 46, 6, 0, 0, 38, 44, 7, 47, 48, 37, 35, 3, 36, 54, 4,
41 |     2, 2, 2, 2, 2, 2, 2, 2, 2, 55, 53, 43, 41, 42, 0,
42 |     0, 1, 14, 15, 29, 8, 11, 1, 1, 33, 1, 1, 34, 1, 21, 1,
43 |     1, 1, 40, 1, 1, 1, 30, 1, 1, 1, 51, 0, 52, 39, 1,
44 |     0, 16, 31, 22, 17, 24, 32, 26, 25, 20, 1, 1, 12, 23, 9, 13,
45 |     28, 1, 18, 19, 10, 27, 1, 1, 1, 49, 45, 50, 0, 0,
46 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
47 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
48 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
49 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
50 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
51 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
52 |     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
53 | };
54 |
55 | /**
56 | * Translates DFA states to action switch labels.
57 | */
58 | private static final int [] ZZ_ACTION = zzUnpackAction();
59 |
60 | private static final String ZZ_ACTION_PACKED_0 =
61 |     "\\\\"0\\\\1\\\\1\\\\1\\\\2\\\\3\\\\1\\\\4\\\\1\\\\5\\\\1\\\\6\\\\1\\\\7\\\\" +
62 |     "\\\\"1\\\\6\\\\2\\\\1\\\\10\\\\1\\\\11\\\\1\\\\2\\\\3\\\\1\\\\1\\\\4\\\\1\\\\2\\\\1\\\\1\\\\" +
63 |     "\\\\"1\\\\15\\\\1\\\\16\\\\1\\\\17\\\\1\\\\20\\\\1\\\\21\\\\1\\\\22\\\\1\\\\23\\\\1\\\\24\\\\" +
64 |     "\\\\"1\\\\25\\\\1\\\\26\\\\1\\\\27\\\\1\\\\1\\\\1\\\\30\\\\1\\\\31\\\\1\\\\4\\\\7\\\\2\\\\" +
65 |     "\\\\"1\\\\32\\\\1\\\\4\\\\2\\\\1\\\\33\\\\1\\\\34\\\\1\\\\2\\\\1\\\\35\\\\1\\\\36\\\\1\\\\0\\\\" +
66 |     "\\\\"1\\\\37\\\\1\\\\0\\\\1\\\\40\\\\2\\\\0\\\\1\\\\41\\\\1\\\\2\\\\1\\\\42\\\\2\\\\2\\\\"
```



# **MANUAL DE PRACTICAS**





```
133     while (i < l) {
134         int high = packed.charAt(i++) << 16;
135         result[j++] = high | packed.charAt(i++);
136     }
137     return j;
138 }
139
140 /**
141 * The transition table of the DFA
142 */
143 private static final int [] ZZ_TRANS = zzUnpackTrans();
144
145 private static final String ZZ_TRANS_PACKED_0 =
146     "1\2\1\3\1\4\1\5\1\6\1\5\1\7\1\10"+
147     "1\1\1\2\3\1\12\2\3\1\13\1\14\1\3\1\1\15"+
148     "2\3\1\16\1\17\1\3\1\20\2\3\1\21\1\22"+
149     "1\1\23\1\3\1\24\1\25\1\26\1\27\1\30\1\31"+
150     "1\1\32\1\33\1\34\1\35\1\36\1\37\1\40\1\41"+
151     "1\1\42\1\43\1\44\1\45\1\46\1\47\1\50\1\51"+
152     "1\1\52\1\53\1\57\1\0\1\3\5\1\0\33\3\5\1\0\1\3"+
153     "1\2\1\0\1\4\1\70\1\0\1\5\1\0\1\5\1\6\1\0\1\4"+
154     "1\6\1\0\1\2\3\5\1\0\1\3\1\55\1\3\1\3\5\1\0\1\3"+
155     "1\20\1\0\1\2\3\5\1\0\1\3\1\56\1\26\3\5\1\0\1\3"+
156     "1\20\1\0\1\2\3\5\1\0\1\5\1\1\57\1\25\3\1\5\1\0\1\3"+
157     "1\20\1\0\1\2\3\5\1\0\1\4\1\60\1\1\61\1\2\3\1\62"+
158     "1\22\3\1\5\1\0\1\1\3\1\20\0\1\2\3\5\1\0\1\0\1\3\1\63"+
159     "1\3\1\64\1\3\1\65\1\2\3\5\1\0\1\3\1\20\0"+
160     "1\2\3\1\5\1\0\1\0\1\3\1\1\66\1\3\1\1\67\1\1\6\1\3\5\1\0"+
161     "1\1\3\1\20\0\1\2\3\5\1\0\1\4\1\3\1\70\1\16\1\3\5\1\0"+
162     "1\1\3\1\20\0\1\2\3\5\1\0\1\3\1\1\71\1\22\1\3\5\1\0"+
163     "1\1\3\1\20\0\1\2\3\5\1\0\1\2\3\1\1\72\1\20\1\3\5\1\0"+
164     "1\1\3\1\20\0\1\2\3\5\1\0\1\2\3\1\1\73\1\12\1\3\5\1\0"+
165     "1\1\3\1\20\0\1\2\3\5\1\0\1\0\1\3\1\1\74\1\22\1\3\5\1\0"+
166     "1\1\3\1\20\0\1\2\3\5\1\0\1\23\1\3\1\75\1\7\1\3\5\1\0"+
167     "1\1\3\1\20\0\1\2\3\5\1\0\1\3\1\1\76\1\15\3\1\1\77"+
168     "1\13\3\1\5\1\0\1\3\1\20\0\1\2\3\5\1\0\1\20\3\1\100"+
169     "1\12\3\1\5\1\0\1\3\1\62\1\0\1\101\1\70\1\0\1\102\1\24\0"+
170     "1\2\3\1\5\1\0\1\16\1\3\1\1\103\1\103\1\3\1\5\1\0\1\3\1\70\1\0"+
171     "1\1\104\1\1\105\1\66\1\0\1\106\1\1\107\1\66\1\0\1\110\1\1\0"+
172     "1\1\111\1\60\1\0\1\112\1\67\1\0\1\113\1\23\1\0\1\54\1\0"+
173     "1\62\1\54\1\1\0\1\2\3\1\5\1\0\1\2\3\1\1\14\1\30\1\3\5\1\0"+
174     "1\1\3\1\20\0\1\2\3\1\5\1\0\1\5\1\3\1\1\115\1\25\1\3\5\1\0"+
175     "1\1\3\1\20\0\1\2\3\1\5\1\0\1\5\1\3\1\1\116\1\25\1\3\5\1\0"+
176     "1\1\3\1\20\0\1\2\3\1\5\1\0\1\0\1\3\1\1\117\1\22\1\3\5\1\0"+
177     "1\1\3\1\20\0\1\2\3\1\5\1\0\1\1\3\1\1\120\1\31\1\3\5\1\0"+
178     "1\1\3\1\20\0\1\2\3\1\5\1\0\1\1\3\1\1\121\1\1\122\1\20\1\0"+
179     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\4\1\3\1\1\123\1\26\1\3"+
180     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\5\1\3\1\1\124\1\15\3\1"+
181     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\22\1\3\1\1\125\1\10\3\1"+
182     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\3\1\1\126\1\17\3\1"+
183     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\16\1\3\1\1\127\1\14\3\1"+
184     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\20\1\3\1\1\130\1\12\3\1"+
185     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\16\1\3\1\1\131\1\14\3\1"+
186     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\20\1\3\1\1\132\1\12\3\1"+
187     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\2\3\1\1\133\1\30\3\1"+
188     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\12\1\3\1\1\134\1\20\3\1"+
189     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\13\1\1\135\1\31\3\1"+
190     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\23\1\3\1\1\136\1\30\3\1"+
191     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\24\1\3\1\1\137\1\6\3\1"+
192     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\20\1\3\1\1\140\1\12\3\1"+
193     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\23\1\3\1\1\141\1\7\3\1"+
194     "1\5\1\0\1\3\1\72\1\0\1\142\1\66\1\0\1\143\1\17\1\0\1\144"+
195     "1\152\1\0\1\145\1\3\1\0\1\2\3\1\5\1\0\1\2\3\1\1\146\1\30\3\1"+
196     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\13\1\1\147\1\17\3\1"+
197     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\13\1\1\150\1\17\3\1"+
198     "1\5\1\0\1\3\1\20\0\1\2\3\1\5\1\0\1\13\1\1\151\1\17\3\1"
```



```
199 "1\5\0\0\1\3\20\0\2\3\5\0\0\14\3\1\152\16\3"+  
200 "1\5\0\0\1\3\20\0\2\3\5\0\0\15\3\1\153\25\3"+  
201 "1\5\0\0\1\3\20\0\2\3\5\0\0\23\3\1\154\7\3"+  
202 "1\5\0\0\1\3\20\0\2\3\5\0\0\15\3\1\155\25\3"+  
203 "1\5\0\0\1\3\20\0\2\3\5\0\0\4\3\1\156\26\3"+  
204 "1\5\0\0\1\3\20\0\2\3\5\0\0\13\1\157\31\3"+  
205 "1\5\0\0\1\3\20\0\2\3\5\0\0\20\3\1\160\12\3"+  
206 "1\5\0\0\1\3\20\0\2\3\5\0\0\11\3\1\161\21\3"+  
207 "1\5\0\0\1\3\20\0\2\3\5\0\0\20\3\1\162\12\3"+  
208 "1\5\0\0\1\3\20\0\2\3\5\0\0\14\3\1\163\16\3"+  
209 "1\5\0\0\1\3\20\0\2\3\5\0\0\16\3\1\164\14\3"+  
210 "1\5\0\0\1\3\20\0\2\3\5\0\0\12\3\1\165\20\3"+  
211 "1\5\0\0\1\3\20\0\2\3\5\0\0\12\3\1\166\20\3"+  
212 "1\5\0\0\1\3\20\0\2\3\5\0\0\12\3\1\167\20\3"+  
213 "1\5\0\0\1\3\20\0\2\3\5\0\0\10\3\1\170\22\3"+  
214 "1\5\0\0\1\3\20\0\1\144\5\5\0\0\172\10\0\2\3"+  
215 "1\5\0\0\20\3\1\172\12\3\5\0\0\13\20\0\2\3"+  
216 "1\5\0\0\2\3\1\173\30\3\5\0\0\13\20\0\2\3"+  
217 "1\5\0\0\12\3\1\174\20\3\5\0\0\13\20\0\2\3"+  
218 "1\5\0\0\1\3\1\175\31\3\5\0\0\13\20\0\2\3"+  
219 "1\5\0\0\5\3\1\176\25\3\5\0\0\13\20\0\2\3"+  
220 "1\5\0\0\2\3\1\177\30\3\5\0\0\13\20\0\2\3"+  
221 "1\5\0\0\12\3\1\178\20\3\5\0\0\13\20\0\2\3"+  
222 "1\5\0\0\20\3\1\179\12\3\5\0\0\13\20\0\2\3"+  
223 "1\5\0\0\1\3\1\180\21\3\5\0\0\13\20\0\2\3"+  
224 "1\5\0\0\10\3\1\181\22\3\5\0\0\13\20\0\2\3"+  
225 "1\5\0\0\14\3\1\182\16\3\5\0\0\13\20\0\2\3"+  
226 "1\5\0\0\5\3\1\183\205\25\3\5\0\0\13\20\0\2\3"+  
227 "1\5\0\0\14\3\1\184\16\3\5\0\0\13\20\0\2\3"+  
228 "1\5\0\0\11\3\1\185\21\3\5\0\0\13\20\0\2\3"+  
229 "1\5\0\0\10\3\1\186\22\3\5\0\0\13\20\0\2\3"+  
230 "1\5\0\0\12\3\1\187\20\3\5\0\0\13\20\0\2\3"+  
231 "1\5\0\0\2\3\1\188\30\3\5\0\0\13\20\0\2\3"+  
  
232 "1\5\0\0\20\3\1\189\12\3\5\0\0\13\20\0\2\3"+  
233 "1\5\0\0\27\3\1\190\21\3\5\0\0\13\20\0\2\3"+  
234 "1\5\0\0\5\3\1\191\25\25\3\5\0\0\13\20\0\2\3"+  
235 "1\5\0\0\11\3\1\192\21\3\5\0\0\13\20\0\2\3"+  
236 "1\5\0\0\17\3\1\193\217\13\3\5\0\0\13\20\0\2\3"+  
237 "1\5\0\0\1\3\1\194\220\31\3\5\0\0\13\20\0\2\3"+  
238 "1\5\0\0\10\3\1\195\221\22\3\5\0\0\13\20\0\2\3"+  
239 "1\5\0\0\20\3\1\196\222\12\3\5\0\0\13\20\0\2\3"+  
240 "1\5\0\0\12\3\1\197\223\20\3\5\0\0\13\20\0\2\3"+  
241 "1\5\0\0\4\3\1\198\224\26\3\5\0\0\13\20\0\2\3"+  
242 "1\5\0\0\1\3\1\199\225\31\3\5\0\0\13\20\0\2\3"+  
243 "1\5\0\0\23\3\1\200\226\17\3\5\0\0\13\20\0\2\3"+  
244 "1\5\0\0\14\3\1\201\227\16\3\5\0\0\13\20\0\2\3"+  
245 "1\5\0\0\2\3\1\202\230\30\3\5\0\0\13\20\0\2\3"+  
246 "1\5\0\0\13\3\1\203\231\17\3\5\0\0\13\20\0\2\3"+  
247 "1\5\0\0\12\3\1\204\232\20\3\5\0\0\13\20\0\2\3"+  
248 "1\5\0\0\20\3\1\205\233\12\3\5\0\0\13\20\0\2\3"+  
249 "1\5\0\0\16\3\1\206\234\14\3\5\0\0\13\20\0\2\3"+  
250 "1\5\0\0\12\3\1\207\235\20\3\5\0\0\13\20\0\2\3"+  
251 "1\5\0\0\20\3\1\208\236\12\3\5\0\0\13\20\0\2\3"+  
252 "1\5\0\0\17\3\1\209\237\13\3\5\0\0\13\20\0\2\3"+  
253 "1\5\0\0\14\3\1\210\240\16\3\5\0\0\13\20\0\2\3"+  
254 "1\5\0\0\14\3\1\211\241\16\3\5\0\0\13\20\0\2\3"+  
255 "1\5\0\0\12\3\1\212\242\20\3\5\0\0\13\20\0\2\3"+  
256 "1\5\0\0\1\3\1\213\243\32\3\5\0\0\13\20\0\2\3"+  
257 "1\5\0\0\25\3\1\214\5\3\5\0\0\13\20\0\2\3"+  
258 "1\5\0\0\10\3\1\215\245\22\3\5\0\0\13\20\0\2\3"+  
259 "1\5\0\0\11\3\1\216\246\21\3\5\0\0\13\20\0\2\3"+  
260 "1\5\0\0\5\3\1\217\247\25\3\5\0\0\13\17\0";  
261  
262 private static int [] zzUnpackTrans() {  
263     int [] result = new int[6720];  
264     int offset = 0;  
}
```



```
266     offset = zzUnpackTrans(ZZ_TRANS_PACKED_0, offset, result);
267     return result;
268 }
269 
270 private static int zzUnpackTrans(String packed, int offset, int [] result) {
271     int i = 0; /* index in packed string */
272     int j = offset; /* index in unpacked array */
273     int l = packed.length();
274     while (i < l) {
275         int count = packed.charAt(i++);
276         int value = packed.charAt(i++);
277         value--;
278         do result[j++] = value; while (--count > 0);
279     }
280     return j;
281 }
282 
283 /* error codes */
284 private static final int ZZ_UNKNOWN_ERROR = 0;
285 private static final int ZZ_NO_MATCH = 1;
286 private static final int ZZ_PUSHBACK_2BIG = 2;
287 
288 /* error messages for the codes above */
289 private static final String ZZ_ERROR_MSG[] = {
290     "Unknown internal scanner error",
291     "Error: could not match input",
292     "Error: pushback value was too large"
293 };
294 
295 /**
296  * ZZ_ATTRIBUTE[astate] contains the attributes of state <code>aState</code>
297 */
298 
299 private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();
300 
301 private static final String ZZ_ATTRIBUTE_PACKED_0 =
302     "\\" + '0' + '\\' + '1' + '\\' + '4' + '\\' + '1' + '\\' + '2' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '0' + '\\' + '1' + '\\' + '3' + '\\' + '1' + '\\' + '4' + '\\' + '1' + '\\' + '3' + '\\' + '1' + '"';
303     "\\" + '1' + '\\' + '1' + '\\' + '5' + '\\' + '1' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '5' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '1' + '\\' + '1' + '"';
304     "\\" + '1' + '\\' + '0' + '\\' + '1' + '\\' + '1' + '\\' + '0' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '0' + '\\' + '2' + '\\' + '6' + '\\' + '1' + '\\' + '2' + '\\' + '1' + '\\' + '1' + '\\' + '0' + '"';
305     "\\" + '1' + '\\' + '1' + '\\' + '1' + '\\' + '2' + '\\' + '3' + '\\' + '1' + '\\' + '1' + '\\' + '1' + '\\' + '5' + '\\' + '1' + '"';
306 
307 private static int [] zzUnpackAttribute() {
308     int [] result = new int[167];
309     int offset = 0;
310     offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
311     return result;
312 }
313 
314 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
315     int i = 0; /* index in packed string */
316     int j = offset; /* index in unpacked array */
317     int l = packed.length();
318     while (i < l) {
319         int count = packed.charAt(i++);
320         int value = packed.charAt(i++);
321         value--;
322         do result[j++] = value; while (--count > 0);
323     }
324     return j;
325 }
326 
327 /**
328  * the input device */
329 private java.io.Reader zzReader;
330 
331 /**
332  * the current state of the DFA */
333 private int zzState;
```



```
331  /** the current lexical state */
332  private int zzLexicalState = YYINITIAL;
333
334  /** this buffer contains the current text to be matched and is
335  | the source of the yytext() string */
336  private char zzBuffer[] = new char[ZZ_BUFFERSIZE];
337
338  /** the textposition at the last accepting state */
339  private int zzMarkedPos;
340
341  /** the current text position in the buffer */
342  private int zzCurrentPos;
343
344  /** startRead marks the beginning of the yytext() string in the buffer */
345  private int zzStartRead;
346
347  /** endRead marks the last character in the buffer, that has been read
348  | from input */
349  private int zzEndRead;
350
351  /** number of newlines encountered up to the start of the matched text */
352  private int yyline;
353
354  /** the number of characters up to the start of the matched text */
355  private int ychar;
356
357  /**
358  * the number of characters from the last newline up to the start of the
359  * matched text
360  */
361  private int yycolumn;
362
363  /**
364  * zzAtBOL == true <=> the scanner is currently at the beginning of a line
365  */
366  private boolean zzAtBOL = true;
367
368  /**
369  * zzAtEOF == true <=> the scanner is at the EOF */
370  private boolean zzAtEOF;
371
372  /** denotes if the user-EOF-code has already been executed */
373  private boolean zzEOFDone;
374
375  /**
376   * user code:
377   * private Symbol symbol(int type, Object value){
378   *     return new Symbol(type, yyline, yycolumn, value);
379   * }
380   * private Symbol symbol(int type){
381   *     return new Symbol(type, yyline, yycolumn);
382   * }
383
384  /**
385   * Creates a new scanner
386   * There is also a java.io.InputStream version of this constructor.
387   *
388   * @param in the java.io.Reader to read input from.
389  */
390  LexerCup(java.io.Reader in) {
391    this.zzReader = in;
392  }
393
394  /**
395   * Creates a new scanner.
396   * There is also java.io.Reader version of this constructor.
397  */
```



```
397     * @param   in  the java.io.InputStream to read input from.  
398     */  
399     LexerCup(java.io.InputStream in) {  
400         this(new java.io.InputStreamReader(in));  
401     }  
402  
403     /**  
404      * Refills the input buffer.  
405      *  
406      * @return      <code>false</code>, iff there was new input.  
407      *  
408      * @exception  java.io.IOException  if any I/O-Error occurs  
409      */  
410     private boolean zzRefill() throws java.io.IOException {  
411  
412         /* first: make room (if you can) */  
413         if (zzStartRead > 0) {  
414             System.arraycopy(zzBuffer, zzStartRead,  
415                             zzBuffer, 0,  
416                             zzEndRead-zzStartRead);  
417  
418             /* translate stored positions */  
419             zzEndRead-= zzStartRead;  
420             zzCurrentPos-= zzStartRead;  
421             zzMarkedPos-= zzStartRead;  
422             zzStartRead = 0;  
423         }  
424  
425         /* is the buffer big enough? */  
426         if (zzCurrentPos >= zzBuffer.length) {  
427             /* if not: blow it up */  
428             char newBuffer[] = new char[zzCurrentPos*2];  
429  
430             System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);  
431             zzBuffer = newBuffer;  
432         }  
433  
434         /* finally: fill the buffer with new input */  
435         int numRead = zzReader.read(zzBuffer, zzEndRead,  
436                                     zzBuffer.length-zzEndRead);  
437  
438         if (numRead > 0) {  
439             zzEndRead+= numRead;  
440             return false;  
441         }  
442         // unlikely but not impossible: read 0 characters, but not at end of stream  
443         if (numRead == 0) {  
444             int c = zzReader.read();  
445             if (c == -1) {  
446                 return true;  
447             } else {  
448                 zzBuffer[zzEndRead++] = (char) c;  
449                 return false;  
450             }  
451         }  
452  
453         // numRead < 0  
454         return true;  
455     }  
456  
457     /**  
458      * Closes the input stream.  
459      */  
460     public final void yclose() throws java.io.IOException {  
461         zzAtEOF = true;          /* indicate end of file */  
462     }
```



```
463     zzEndRead = zzStartRead; /* invalidate buffer */
464
465     if (zzReader != null)
466         zzReader.close();
467 }
468
469
470 /**
471 * Resets the scanner to read from a new input stream.
472 * Does not close the old reader.
473 *
474 * All internal variables are reset, the old input stream
475 * <b>cannot</b> be reused (internal buffer is discarded and lost).
476 * Lexical state is set to <tt>ZZ_INITIAL</tt>.
477 *
478 * @param reader the new input stream
479 */
480 public final void yyreset(java.io.Reader reader) {
481     zzReader = reader;
482     zzAtBOL = true;
483     zzAtEOF = false;
484     zzEOFDone = false;
485     zzEndRead = zzstartRead = 0;
486     zzCurrentPos = zzMarkedPos = 0;
487     yyline = ychar = ycolumn = 0;
488     zzLexicalState = YYINITIAL;
489 }
490
491
492 /**
493 * Returns the current lexical state.
494 */
495 public final int yystate() {
496     return zzLexicalState;
497 }
498
499
500 /**
501 * Enters a new lexical state
502 *
503 * @param newState the new lexical state
504 */
505 public final void yybegin(int newState) {
506     zzLexicalState = newState;
507 }
508
509
510 /**
511 * Returns the text matched by the current regular expression.
512 */
513 public final String yytext() {
514     return new String(zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
515 }
516
517
518 /**
519 * Returns the character at position <tt>pos</tt> from the
520 * matched text.
521 *
522 * It is equivalent to yytext().charAt(pos), but faster
523 *
524 * @param pos the position of the character to fetch.
525 *          A value from 0 to yylength()-1.
526 *
527 * @return the character at position pos
528 */
```



```
529  public final char yycharat(int pos) {
530      return zzBuffer[zzStartRead+pos];
531  }
532
533
534  /**
535   * Returns the length of the matched text region.
536   */
537  public final int yylength() {
538      return zzMarkedPos-zzStartRead;
539  }
540
541
542  /**
543   * Reports an error that occurred while scanning.
544   *
545   * In a wellformed scanner (no or only correct usage of
546   * yypushback(int) and a match-all fallback rule) this method
547   * will only be called with things that "Can't Possibly Happen".
548   * If this method is called, something is seriously wrong
549   * (e.g. a JFlex bug producing a faulty scanner etc.).
550   *
551   * Usual syntax/scanner level error handling should be done
552   * in error fallback rules.
553   *
554   * @param errorCode the code of the errormessage to display
555   */
556  private void zzScanError(int errorCode) {
557      String message;
558      try {
559          message = ZZ_ERROR_MSG[errorCode];
560      }
561      catch (ArrayIndexOutOfBoundsException e) {
562          message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
563      }
564
565      throw new Error(message);
566  }
567
568
569  /**
570   * Pushes the specified amount of characters back into the input stream.
571   *
572   * They will be read again by the next call of the scanning method
573   *
574   * @param number the number of characters to be read again.
575   *               This number must not be greater than yylength()!
576   */
577  public void yypushback(int number) {
578      if (number > yylength())
579          zzScanError(ZZ_PUSHBACK_2BIG);
580
581      zzMarkedPos -= number;
582  }
583
584
585  /**
586   * Contains user EOF-code, which will be executed exactly once,
587   * when the end of file is reached
588   */
589  private void zzDoEOF() throws java.io.IOException {
590      if (!zzEOFDone) {
591          zzEOFDone = true;
592          yyclose();
593      }
594  }
```



```
595
596
597 /**
598 * Resumes scanning until the next regular expression is matched,
599 * the end of input is encountered or an I/O-Error occurs.
600 *
601 * @return      the next token
602 * @exception   java.io.IOException if any I/O-Error occurs
603 */
604 public java_cup.runtime.Symbol next_token() throws java.io.IOException {
605     int zzInput;
606     int zzAction;
607
608     // cached fields:
609     int zzCurrentPosL;
610     int zzMarkedPosL;
611     int zzEndReadL = zzEndRead;
612     char [] zzBufferL = zzBuffer;
613     char [] zzCMapL = ZZ_CMAP;
614
615     int [] zzTransL = ZZ_TRANS;
616     int [] zzRowMapL = ZZ_ROWMAP;
617     int [] zzAttrL = ZZ_ATTRIBUTE;
618
619     while (true) {
620         zzMarkedPosL = zzMarkedPos;
621
622         yychar+= zzMarkedPosL-zzStartRead;
623
624         boolean zzR = false;
625         for (zzCurrentPosL = zzStartRead; zzCurrentPosL < zzMarkedPosL;
626              zzCurrentPosL++) {
627             switch (zzBufferL[zzCurrentPosL]) {
628
629                 case '\u000B':
630                 case '\u000C':
631                 case '\u0085':
632                 case '\u2028':
633                 case '\u2029':
634                     yyline++;
635                     zzR = false;
636                     break;
637                 case '\r':
638                     yyline++;
639                     zzR = true;
640                     break;
641                 case '\n':
642                     if (zzR)
643                         zzR = false;
644                     else {
645                         yyline++;
646                     }
647                     break;
648                 default:
649                     zzR = false;
650                 }
651             }
652             if (zzR) {
653                 // peek one character ahead if it is \n (if we have counted one line too much)
654                 boolean zzPeek;
655                 if (zzMarkedPosL < zzEndReadL)
656                     zzPeek = zzBufferL[zzMarkedPosL] == '\n';
657                 else if (zzAtEOF)
658                     zzPeek = false;
659                 else {
660                     boolean eof = zzRefill();
```



```
661     zzEndReadL = zzEndRead;
662     zzMarkedPosL = zzMarkedPos;
663     zzBufferL = zzBuffer;
664     if (eof)
665         zzPeek = false;
666     else
667         zzPeek = zzBufferL[zzMarkedPosL] == '\n';
668     }
669     if (zzPeek) yyline--;
670 }
671 zzAction = -1;
672
673 zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
674
675 zzState = ZZ_LEXSTATE[zzLexicalState];
676
677
678 zzForAction: {
679     while (true) {
680
681         if (zzCurrentPosL < zzEndReadL)
682             zzInput = zzBufferL[zzCurrentPosL++];
683         else if (zzAtEOF) {
684             zzInput = YYEOF;
685             break zzForAction;
686         }
687         else {
688             // store back cached positions
689             zzCurrentPos = zzCurrentPosL;
690             zzMarkedPos = zzMarkedPosL;
691             boolean eof = zzRefill();
692             // get translated positions and possibly new buffer
693             zzCurrentPosL = zzCurrentPos;
694
695             zzMarkedPosL = zzMarkedPos;
696             zzBufferL = zzBuffer;
697             zzEndReadL = zzEndRead;
698             if (eof) {
699                 zzInput = YYEOF;
700                 break zzForAction;
701             }
702             else {
703                 zzInput = zzBufferL[zzCurrentPosL++];
704             }
705             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
706             if (zzNext == -1) break zzForAction;
707             zzState = zzNext;
708
709             int zzAttributes = zzAttrL[zzState];
710             if ( (zzAttributes & 1) == 1 ) {
711                 zzAction = zzState;
712                 zzMarkedPosL = zzCurrentPosL;
713                 if ( (zzAttributes & 8) == 8 ) break zzForAction;
714             }
715
716         }
717     }
718
719     // store back cached position
720     zzMarkedPos = zzMarkedPosL;
721
722     switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
723         case 14:
724             { return new Symbol(sym.menorQue, yychar, yyline, yytext()); }
725         case 58: break;
```



```
727     case 7:
728         { return new Symbol(sym.comillaSimple, yychar, yyline, yytext());
729     }
730     case 59: break;
731     case 57:
732         { return new Symbol(sym.predeterminado, yychar, yyline, yytext());
733     }
734     case 60: break;
735     case 10:
736         { return new Symbol(sym.operadorMultiplicacion, yychar, yyline, yytext());
737     }
738     case 61: break;
739     case 5:
740         { return new Symbol(sym.operadorDivision, yychar, yyline, yytext());
741     }
742     case 62: break;
743     case 24:
744         { return new Symbol(sym.puntoDecimal, yychar, yyline, yytext());
745     }
746     case 63: break;
747     case 23:
748         { return new Symbol(sym.corchetDer, yychar, yyline, yytext());
749     }
750     case 64: break;
751     case 8:
752         { return new Symbol(sym.operadorSuma, yychar, yyline, yytext());
753     }
754     case 65: break;
755     case 35:
756         { return new Symbol(sym.cadena, yychar, yyline, yytext());
757     }
758     case 66: break;
759     case 52:
760         { return new Symbol(sym.cicloWhile, yychar, yyline, yytext());
761     }
762     case 67: break;
763     case 41:
764         { return new Symbol(sym.condicionalElse, yychar, yyline, yytext());
765     }
766     case 68: break;
767     case 27:
768         { return new Symbol(sym.incremento, yychar, yyline, yytext());
769     }
770     case 69: break;
771     case 48:
772         { return new Symbol(sym.cicloDo, yychar, yyline, yytext());
773     }
774     case 70: break;
775     case 40:
776         { return new Symbol(sym.carácter, yychar, yyline, yytext());
777     }
778     case 71: break;
779     case 50:
780         { return new Symbol(sym.detener, yychar, yyline, yytext());
781     }
782     case 72: break;
783     case 22:
784         { return new Symbol(sym.corchetIzq, yychar, yyline, yytext());
785     }
786     case 73: break;
787     case 33:
788         { return new Symbol(sym.valorEntero, yychar, yyline, yytext());
789     }
790     case 74: break;
791     case 32:
792         { return new Symbol(sym.inicioTexto, yychar, yyline, yytext());
```



# MANUAL DE PRÁCTICAS



```
793     }
794     case 75: break;
795     case 55:
796     {
797         return new Symbol(sym.constant, yychar, yyline, yytext());
798     }
799     case 76: break;
800     case 16:
801     {
802         return new Symbol(sym.operadorO, yychar, yyline, yytext());
803     }
804     case 77: break;
805     case 38:
806     {
807         return new Symbol(sym.finLinea, yychar, yyline, yytext());
808     }
809     case 78: break;
810     case 34:
811     {
812         return new Symbol(sym.valorBooleano, yychar, yyline, yytext());
813     }
814     case 80: break;
815     case 56:
816     {
817         return new Symbol(sym.funcionIntroducirDatos, yychar, yyline, yytext());
818     }
819     case 81: break;
820     case 54:
821     {
822         return new Symbol(sym.imprimir, yychar, yyline, yytext());
823     }
824     case 28:
825     {
826         return new Symbol(sym.deCREMENTO, yychar, yyline, yytext());
827     }
828     case 83: break;
829     case 37:
830     {
831         return new Symbol(sym.menorIgual, yychar, yyline, yytext());
832     }
833     case 84: break;
834     case 31:
835     {
836         return new Symbol(sym.finalTexto, yychar, yyline, yytext());
837     }
838     case 85: break;
839     case 13:
840     {
841         return new Symbol(sym.mayorQue, yychar, yyline, yytext());
842     }
843     case 86: break;
844     case 47:
845     {
846         return new Symbol(sym.cicloFor, yychar, yyline, yytext());
847     }
848     case 87: break;
849     case 21:
850     {
851         return new Symbol(sym.llaveDer, yychar, yyline, yytext());
852     }
853     case 88: break;
854     case 29:
855     {
856         return new Symbol(sym.comparacionIgualdad, yychar, yyline, yytext());
857     }
858     case 89: break;
859     case 25:
860     {
861         return new Symbol(sym.dosPuntos, yychar, yyline, yytext());
862     }
863     case 90: break;
864     case 46:
865     {
866         return new Symbol(sym.funcionSwitch, yychar, yyline, yytext());
867     }
868     case 91: break;
```



```
859     case 26:
860         { return new Symbol(sym.condicionalIf, yychar, yyline, yytext());
861         }
862         case 92: break;
863         case 12:
864             { return new Symbol(sym.operadorPotencia, yychar, yyline, yytext());
865             }
866         case 93: break;
867         case 45:
868             { return new Symbol(sym.funcionSalirCase, yychar, yyline, yytext());
869             }
870         case 94: break;
871         case 20:
872             { return new Symbol(sym.llaveIzq, yychar, yyline, yytext());
873             }
874         case 95: break;
875         case 2:
876             { return new Symbol(sym.Identificador, yychar, yyline, yytext());
877             }
878         case 96: break;
879         case 19:
880             { return new Symbol(sym.parentDer, yychar, yyline, yytext());
881             }
882         case 97: break;
883         case 51:
884             { return new Symbol(sym.funcion, yychar, yyline, yytext());
885             }
886         case 98: break;
887         case 44:
888             { return new Symbol(sym.clase, yychar, yyline, yytext());
889             }
890         case 99: break;
891         case 18:
892             { return new Symbol(sym.parentIzq, yychar, yyline, yytext());
893             }
894         case 100: break;
895         case 43:
896             { return new Symbol(sym.funcionLeer, yychar, yyline, yytext());
897             }
898         case 101: break;
899         case 1:
900             { return new Symbol(sym.ERROR, yychar, yyline, yytext());
901             }
902         case 102: break;
903         case 17:
904             { return new Symbol(sym.operadorNo, yychar, yyline, yytext());
905             }
906         case 103: break;
907         case 9:
908             { return new Symbol(sym.operadorResta, yychar, yyline, yytext());
909             }
910         case 104: break;
911         case 6:
912             { return new Symbol(sym.comillaDoble, yychar, yyline, yytext());
913             }
914         case 105: break;
915         case 36:
916             { return new Symbol(sym.mayorIgual, yychar, yyline, yytext());
917             }
918         case 106: break;
919         case 11:
920             { return new Symbol(sym.operadorModulo, yychar, yyline, yytext());
921             }
922         case 107: break;
923         case 49:
924             { return new Symbol(sym.operadorRaiz, yychar, yyline, yytext());
```



```
925     }
926     case 108: break;
927     case 53:
928     { return new Symbol(sym.variable, yychar, yyline, yytext());
929     }
930     case 109: break;
931     case 4:
932     { /*Ignore*/
933     }
934     case 110: break;
935     case 30:
936     { return new Symbol(sym.asignacion, yychar, yyline, yytext());
937     }
938     case 111: break;
939     case 39:
940     { return new Symbol(sym.valorFlotante, yychar, yyline, yytext());
941     }
942     case 112: break;
943     case 15:
944     { return new Symbol(sym.operadorY, yychar, yyline, yytext());
945     }
946     case 113: break;
947     case 42:
948     { return new Symbol(sym.funcionCase, yychar, yyline, yytext());
949     }
950     case 114: break;
951     default:
952     if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
953         zzAtEOF = true;
954         zzDoEOF();
955         { return new java_cup.runtime.Symbol(sym.EOF); }
956     }
957     else {
958         zzScanError(ZZ_NO_MATCH);
959     }
960 }
961 }
962 }
963
964
965 }
966 }
```

### Archivo Sintax.cup

El archivo sintax.cup es el componente encargado de definir el analizador sintáctico para el lenguaje en cuestión, utilizando Java CUP. Este archivo especifica la gramática del lenguaje, indicando cómo deben organizarse los tokens generados por el analizador léxico para formar estructuras válidas, como declaraciones, expresiones y control de flujo. El analizador sintáctico toma la secuencia de tokens producidos por el lexer y los agrupa en una jerarquía de símbolos, construyendo un árbol sintáctico que refleja la estructura del programa. A través de reglas como INICIO, SENTENCIA, y DECLARACION, el archivo define los patrones válidos que el código debe seguir, permitiendo detectar errores en la sintaxis, como paréntesis o llaves mal emparejadas.

Este archivo es fundamental para garantizar que el código fuente siga las reglas estructurales del lenguaje, lo que es esencial para la correcta ejecución de un compilador o intérprete. Al identificar y manejar correctamente las construcciones sintácticas y los errores, el analizador sintáctico ayuda a asegurar que el programa sea ejecutado de manera precisa. Además, permite detectar errores a nivel de estructura antes de que el código se ejecute, lo cual es crucial para el desarrollo de software robusto y confiable. Sin un archivo como sintax.cup, sería casi imposible validar la correcta formación de los programas en el lenguaje definido.



## Código Realizado:

```
1 package codigo;
2
3
4 import java_cup.runtime.Symbol;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 parser code
9 {
10     private java.util.List<String> errores = new java.util.ArrayList<>();
11     private String textoCompleto; // Para calcular las columnas por línea
12
13     public void syntax_error(Symbol s) {
14         int linea = s.right + 1;
15         int columnaReal = calcularColumnaReal(linea, s.left + 1);
16
17         errores.add("Error de sintaxis. Línea: " + linea +
18                     ", Columna: " + columnaReal + ", Texto: \""
19                     + s.value + "\"");
20     }
21
22     public java.util.List<String> getErrores() {
23         return errores;
24     }
25
26     // Configura el texto completo para cálculos de posición
27     public void setTextoCompleto(String textoCompleto) {
28         this.textoCompleto = textoCompleto;
29     }
30
31     // Calcula la columna real reiniciando en cada nueva línea
32     private int calcularColumnaReal(int linea, int posicionGlobal) {
33         int columna = 1;
34         int contadorLineas = 1;
35
36         for (int i = 0; i < posicionGlobal; i++) {
37             if (textoCompleto.charAt(i) == '\n') {
38                 contadorLineas++;
39                 if (contadorLineas == linea) {
40                     columna = 1; // Reinicia la columna al inicio de la nueva línea
41                 }
42                 else if (contadorLineas == linea) {
43                     columna++;
44                 }
45             }
46         }
47         return columna;
48     }
49
50     terminal Linea, comillaDoble, comillaSimple, valorEntero, valorFlotante,
51     valorBooleano, cadena, carácter, condicionalIf, condicionalElse, cicloFor,
52     cicloWhile, cicloDo, funcionSwitch, funcionCase, funcionSalirCase,
53     predeterminado, detener, constante, variable, funcion, clase, imprimir,
54     funcionLeer, funcionIntroducirDatos, operadorSuma, operadorResta,
55     operadorMultiplicacion, operadorDivision, operadorModulo, operadorPotencia,
56     operadorRaiz, asignacion, comparacionIgualdad, mayorQue, menorQue,
57     mayorIgual, menorIgual, operadorY, operadorO, operadorNo, incremento,
58     decremento, parentIzq, parentDer, llaveIzq, llaveDer, corchetIzq,
59     corchetDer, finLinea, inicioTexto, finalTexto, puntoDecimal,
60     dosPuntos, Identificador, Numero, ERROR;
61
62     non terminal INICIO, SENTENCIA, DECLARACION, DECLARACION_FOR, CONDICIONAL_IF,
63     CONDICIONAL_IF_ELSE, CICLO WHILE, CICLO_DO WHILE, CICLO_FOR,
64     SENTENCIA_BOOLEANA, SENTENCIA_FOR, OPERACION_ARITMETICA, FUNCION_SWITCH,
65     FUNCION_CASE, FUNCION_SALIR_CASE, SENTENCIA_CONSTANTE, SENTENCIA_VARIABLE,
66     SENTENCIA_IMPRIMIR, SENTENCIA_LEER;
```



# MANUAL DE PRÁCTICAS



```
67      start with INICIO;
68
69      INICIO ::= 
70          clase Identificador parentIzq parentDer llaveIzq SENTENCIA llaveDer |
71          error llaveDer { : System.out.println("Error en SENTENCIA"); :}
72      ;
73
74      SENTENCIA ::= 
75          SENTENCIA DECLARACION |
76          DECLARACION |
77          SENTENCIA CONDICIONAL_IF |
78          CONDICIONAL_IF |
79          SENTENCIA CONDICIONAL_IF_ELSE |
80          CONDICIONAL_IF_ELSE |
81          SENTENCIA CICLO_WHILE |
82          CICLO_WHILE |
83          SENTENCIA CICLO_DO_WHILE |
84          CICLO_DO_WHILE |
85          SENTENCIA CICLO_FOR |
86          CICLO_FOR |
87          SENTENCIA OPERACION_ARITMETICA |
88          OPERACION_ARITMETICA |
89          SENTENCIA_BOLEANA |
90          SENTENCIA_FUNCION_SWITCH |
91          FUNCION_SWITCH |
92          SENTENCIA SENTENCIA_IMPRIMIR |
93          SENTENCIA_IMPRIMIR |
94          SENTENCIA SENTENCIA_LEER |
95          SENTENCIA_LEER |
96          SENTENCIA SENTENCIA_CONSTANTE |
97          SENTENCIA_CONSTANTE |
98          SENTENCIA SENTENCIA_VARIABLE |
99
100         SENTENCIA_VARIABLE |
101         error finLinea { : System.out.println("Error en SENTENCIA"); :} |
102         error llaveDer { : System.out.println("Error en SENTENCIA"); :}
103     ;
104
105     DECLARACION ::= 
106         valorEntero Identificador finLinea |
107         valorEntero Identificador asignacion Numero finLinea |
108         valorFlotante Identificador asignacion Numero puntoDecimal Numero finLinea |
109         valorEntero Identificador incremento finLinea |
110         valorEntero Identificador decremento finLinea |
111         cadena Identificador asignacion comillaDoble Identificador comillaDoble finLinea
112     ;
113
114     OPERACION_ARITMETICA ::= 
115         Identificador operadorSuma Identificador finLinea |
116         Identificador operadorResta Identificador finLinea |
117         Identificador operadorMultiplicacion Identificador finLinea |
118         Identificador operadorDivision Identificador finLinea |
119         Identificador operadorModulo Identificador finLinea |
120         Identificador operadorPotencia Identificador finLinea |
121         Identificador operadorRaiz Identificador finLinea
122     ;
123
124     CONDICIONAL_IF ::= 
125         condicionalIf parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer
126     ;
127
128     SENTENCIA_BOLEANA ::= 
129         valorBooleano |
130         Identificador comparacionIgualdad valorBooleano |
131         Identificador comparacionIgualdad Numero |
132         Identificador comparacionIgualdad Identificador |
```



# MANUAL DE PRÁCTICAS



```
133     Identificador mayorQue Numero |
134     Identificador menorQue Numero |
135     Identificador mayorIgual Numero |
136     Identificador menorIgual Numero |
137     Identificador operadorY Identificador |
138     Identificador operadorO Identificador |
139     operadorNo Identificador
140 ;
141
142 CONDICIONAL_IF_ELSE ::=*
143     condicionalIf parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer
144     condicionalElse llaveIzq SENTENCIA llaveDer
145 ;
146
147 CICLO_WHILE ::=*
148     cicloWhile parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer
149 ;
150
151 CICLO_DO_WHILE ::=*
152     cicloDo llaveIzq SENTENCIA llaveDer cicloWhile parentIzq SENTENCIA_BOLEANA parentDer finLinea
153 ;
154
155 CICLO_FOR ::=*
156     cicloFor parentIzq SENTENCIA_FOR parentDer llaveIzq SENTENCIA llaveDer
157 ;
158
159 SENTENCIA_FOR ::=*
160     valorEntero Identificador asignacion Numero finLinea SENTENCIA_BOLEANA finLinea DECLARACION_FOR |
161     Identificador asignacion Numero finLinea SENTENCIA_BOLEANA finLinea DECLARACION_FOR
162 ;
163
164 DECLARACION_FOR ::=*
165     Identificador asignacion Numero |
166
167     Identificador incremento |
168     incremento Identificador
169 ;
170
171 FUNCION_SWITCH ::=*
172     funcionSwitch parentIzq Identificador parentDer llaveIzq FUNCION_CASE predeterminado dosPuntos SENTENCIA detener finLinea llaveDer
173 ;
174
175 FUNCION_CASE ::=*
176     funcionCase valorEntero dosPuntos SENTENCIA detener finLinea |
177     FUNCION_CASE funcionCase valorEntero dosPuntos SENTENCIA detener finLinea
178 ;
179
180 FUNCION_SALIR_CASE ::=*
181     funcionSalirCase finLinea
182 ;
183
184 SENTENCIA_CONSTANTE ::=*
185     constante Identificador asignacion Numero finLinea
186 ;
187
188 SENTENCIA_VARIABLE ::=*
189     variable Identificador asignacion Numero finLinea
190 ;
191
192 SENTENCIA_IMPRIMIR ::=*
193     imprimir parentIzq Identificador parentDer finLinea
194 ;
195
196 SENTENCIA_LEER ::=*
197     funcionLeer parentIzq Identificador parentDer finLinea
```

## Archivo Sintax.java (Generado automáticamente)

El archivo sintax.java es el archivo generado automáticamente por Java CUP a partir de las especificaciones de la gramática definidas en el archivo sintax.cup. Este archivo contiene la implementación del analizador sintáctico que se utiliza para procesar la secuencia de tokens generados por el analizador léxico. Específicamente, sintax.java maneja la lógica para recorrer y analizar la estructura de las cadenas de entrada, identificando las secuencias correctas de símbolos según la gramática definida y construyendo un árbol de sintaxis abstracta (AST) o un flujo de ejecución que puede ser utilizado por el compilador o intérprete. Este archivo también maneja los errores sintácticos, activando los métodos adecuados cuando se encuentra una secuencia de tokens que no corresponde a ninguna de las reglas de la gramática.

Sintax.java es el componente central del proceso de análisis sintáctico en el ciclo de compilación. Permite verificar que el código fuente sigue la estructura esperada del lenguaje, lo que facilita la detección de errores en una etapa temprana del proceso de desarrollo. Además, sintax.java es fundamental para transformar el flujo de tokens del lexer en una representación más comprensible y manipulable, ya sea para la posterior generación de código o para realizar optimizaciones. Sin este archivo, el proceso de interpretación o compilación del lenguaje no podría continuar de manera efectiva, pues la sintaxis del código no podría ser validada ni convertida en una estructura que el sistema pueda entender y ejecutar.

### Código Generado:

```
1 //-----
2 // The following code was generated by CUP v0.11a beta 20060608
3 // Tue Dec 31 15:24:08 CST 2024
4 //-----
5
6 package codigo;
7
8 import java_cup.runtime.Symbol;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 /**
13 * CUP v0.11a beta 20060608 generated parser.
14 * @version Tue Dec 31 15:24:08 CST 2024
15 */
16 public class Sintax extends java_cup.runtime.lr_parser {
17
18     /**
19      * Default constructor.
20      */
21     public Sintax() {super();}
22
23     /**
24      * Constructor which sets the default scanner.
25      */
26     public Sintax(java_cup.runtime.Scanner s) {super(s);}
27
28     /**
29      * Production table.
30      */
31     protected static final short _production_table[][] =
32         unpackFromStrings(new String[] {
33             "\u0001\u0000\u0020\u0040\u0000\u0020\u0020\u0010\u0000\u0020\u0020" +
34             "\u0040\u0000\u0020\u0030\u0040\u0000\u0020\u0030\u0000\u0020\u0030" +
35             "\u0000\u0020\u0030\u0000\u0020\u0030\u0040\u0000\u0020\u0030\u0000" +
36             "\u0020\u0030\u0040\u0000\u0020\u0030\u0000\u0020\u0030\u0000" +
```



# MANUAL DE PRÁCTICAS



```
34      "\003\003\000\002\003\004\000\002\003\003\000\002\003" +
35      "\004\000\002\003\003\000\002\003\000\002\003\004" +
36      "\000\002\003\003\000\002\003\004\000\002\003\000" +
37      "\002\003\004\000\002\003\003\000\002\003\004\000\002" +
38      "\003\003\000\002\003\004\000\002\003\003\000\002\003" +
39      "\004\000\002\003\004\000\002\004\005\000\002\004\007" +
40      "\000\002\004\011\000\002\004\006\000\002\004\006\000" +
41      "\002\004\011\000\002\015\006\000\002\015\006\000\002" +
42      "\015\006\000\002\015\006\000\002\015\006\000\002\015" +
43      "\006\000\002\015\006\000\002\006\011\000\002\013\003" +
44      "\002\002\013\005\000\002\013\005\000\002\013\005\000" +
45      "\002\013\005\000\002\013\005\000\002\013\005\000\002" +
46      "\013\005\000\002\013\005\000\002\013\005\000\002\013" +
47      "\004\000\002\007\015\000\002\010\011\000\002\011\013" +
48      "\000\002\012\011\000\002\014\012\000\002\014\011\000" +
49      "\002\005\005\000\002\005\004\000\002\005\004\000\002" +
50      "\016\016\000\002\017\010\000\002\017\011\000\002\020" +
51      "\004\000\002\021\007\000\002\022\007\000\002\023\007" +
52      "\000\002\024\007" );
53
54  /** Access to production table. */
55  public short[][] production_table() { return _production_table; }
56
57  /** Parse-action table. */
58  protected static final short[][] _action_table =
59  unpackFromStrings(new String[] {
60      "\000\316\000\006\003\004\031\006\001\002\000\004\062" +
61      "\320\001\002\000\004\002\317\001\002\000\004\072\007" +
62      "\001\002\000\004\057\010\001\002\000\004\060\011\001" +
63      "\002\000\004\061\012\001\002\000\042\003\044\007\041" +
64      "\010\032\011\034\012\013\014\014\016\036\017\050\020" +
65      "\027\021\031\026\025\027\045\032\021\033\043\054\016" +
66      "\072\030\001\002\000\004\072\311\001\002\000\004\057" +
67      "\277\001\002\000\004\007\ufff9\010\ufff9\012\ufff9\014\ufff9" +
68      "\016\ufff9\017\ufff9\020\ufff9\021\ufff9\025\ufff9\026\ufff9\027" +
69      "\ufff9\032\ufff9\033\ufff9\062\ufff9\072\ufff9\001\002\000\004" +
70      "\072\276\001\002\000\040\007\ufffd\010\ufffd\012\ufffd\014" +
71      "\ufffd\016\ufffd\017\ufffd\020\ufffd\021\ufffd\025\ufffd\026\ufffd" +
72      "\027\ufffd\032\ufffd\033\ufffd\062\ufffd\072\ufffd\001\002\000" +
73      "\036\007\041\010\032\012\013\014\014\016\036\017\050" +
74      "\020\027\021\031\026\025\027\045\032\021\033\043\062" +
75      "\0275\072\106\001\002\000\004\057\271\001\002\000\040" +
76      "\007\uffec\010\uffec\012\uffec\014\uffec\016\uffec\017\uffec\020" +
77      "\uffec\021\uffec\025\uffec\026\uffec\027\uffec\032\uffec\033\uffec" +
78      "\062\uffec\072\uffec\001\002\000\040\007\ufff3\010\ufff3\012" +
79      "\ufff3\014\ufff3\016\ufff3\017\ufff3\020\ufff3\021\ufff3\025\ufff3" +
80      "\026\ufff3\027\ufff3\032\ufff3\033\ufff3\062\ufff3\072\ufff3\001" +
81      "\002\000\040\007\ufffb\010\ufffb\012\ufffb\014\ufffb\016\ufffb" +
82      "\017\ufffb\020\ufffb\021\ufffb\025\ufffb\026\ufffb\027\ufffb\032" +
83      "\ufffb\033\ufffb\062\ufffb\072\ufffb\001\002\000\004\072\265" +
84      "\001\002\000\040\007\ufff5\010\ufff5\012\ufff5\014\ufff5\016" +
85      "\ufff5\017\ufff5\020\ufff5\021\ufff5\025\ufff5\026\ufff5\027\ufff5" +
86      "\032\ufff5\033\ufff5\062\ufff5\072\ufff5\001\002\000\004\061" +
87      "\255\001\002\000\036\035\116\036\117\037\115\040\121" +
88      "\041\123\042\120\043\122\045\060\046\057\047\056\050" +
89      "\054\051\055\052\062\053\061\001\002\000\004\057\226" +
90      "\001\002\000\004\072\220\001\002\000\040\007\uffea\010" +
91      "\uffea\012\uffea\014\uffea\016\uffea\017\uffea\020\uffea\021\uffea" +
92      "\025\uffea\026\uffea\027\uffea\032\uffea\033\uffea\062\uffea\072" +
93      "\uffea\001\002\000\044\007\uffd5\010\uffd5\012\uffd5\014\uffd5" +
94      "\016\uffd5\017\uffd5\020\uffd5\021\uffd5\025\uffd5\026\uffd5\027" +
95      "\uffd5\032\uffd5\033\uffd5\060\uffd5\062\uffd5\065\uffd5\072\uffd5" +
96      "\001\002\000\040\007\ufff7\010\ufff7\012\ufff7\014\ufff7\016" +
97      "\ufff7\017\ufff7\020\ufff7\021\ufff7\025\ufff7\026\ufff7\027\ufff7" +
98      "\032\ufff7\033\ufff7\062\ufff7\072\ufff7\001\002\000\004\057" +
99      "\165\001\002\000\040\007\uffe8\010\uffe8\012\uffe8\014\uffe8" +
```



# MANUAL DE PRÁCTICAS



```
"\016\uffe8\017\uffe8\020\uffe8\021\uffe8\025\uffe8\026\uffe8\027" +
"\uffe8\032\uffe8\033\uffe8\062\uffe8\072\uffe8\001\002\000\040" +
"\007\uffee\010\uffee\012\uffee\014\uffee\016\uffee\017\uffee\020" +
"\uffee\021\uffee\025\uffee\026\uffee\027\uffee\032\uffee\033\uffee" +
"\062\uffee\072\uffee\001\002\000\004\072\154\001\002\000" +
"\040\007\ufff0\010\ufff0\012\ufff0\014\ufff0\016\ufff0\017\ufff0" +
"\020\ufff0\021\ufff0\025\ufff0\026\ufff0\027\ufff0\032\ufff0\033" +
"\ufff0\062\ufff0\072\ufff0\001\002\000\004\057\150\001\002" +
"\000\006\062\146\065\147\001\002\000\004\072\142\001" +
"\002\000\040\007\ufff1\010\ufff1\012\ufff1\014\ufff1\016\ufff1" +
"\017\ufff1\020\ufff1\021\ufff1\025\ufff1\026\ufff1\027\ufff1\032" +
"\ufff1\033\ufff1\062\ufff1\072\ufff1\001\002\000\040\007\uffe6" +
"\010\uffe6\012\uffe6\014\uffe6\016\uffe6\017\uffe6\020\uffe6\021" +
"\uffe6\025\uffe6\026\uffe6\027\uffe6\032\uffe6\033\uffe6\062\uffe6" +
"\072\uffe6\001\002\000\004\057\051\001\002\000\010\011" +
"\034\054\016\072\053\001\002\000\004\060\074\001\002" +
"\000\020\045\060\046\057\047\056\050\054\051\055\052" +
"\062\053\061\001\002\000\004\073\073\001\002\000\004" +
"\073\072\001\002\000\004\073\071\001\002\000\004\073" +
"\070\001\002\000\010\011\066\072\067\073\065\001\002" +
"\000\004\072\064\001\002\000\004\072\063\001\002\000" +
"\044\007\uffcd\010\uffcd\012\uffcd\014\uffcd\016\uffcd\017\uffcd" +
"\020\uffcd\021\uffcd\025\uffcd\026\uffcd\027\uffcd\032\uffcd\033" +
"\uffcd\060\uffcd\062\uffcd\065\uffcd\072\uffcd\001\002\000\044" +
"\007\uffcc\010\uffcc\012\uffcc\014\uffcc\016\uffcc\017\uffcc\020" +
"\uffcc\021\uffcc\025\uffcc\026\uffcc\027\uffcc\032\uffcc\033\uffcc" +
"\060\uffcc\062\uffcc\065\uffcc\072\uffcc\001\002\000\044\007" +
"\uffd3\010\012\uffd3\014\uffd3\016\uffd3\017\uffd3\020\uffd3" +
"\021\uffd3\025\uffd3\026\uffd3\027\uffd3\032\uffd3\033\uffd3\060" +
"\uffd3\062\uffd3\065\uffd3\072\uffd3\001\002\000\044\007\uffd4" +
"\010\uffd4\012\uffd4\014\uffd4\016\uffd4\017\uffd4\020\uffd4\021" +
"\uffd4\025\uffd4\026\uffd4\027\uffd4\032\uffd4\033\uffd4\060\uffd4" +
"\062\uffd4\065\uffd4\072\uffd4\001\002\000\044\007\uffd2\010" +
"\uffd2\012\uffd2\014\uffd2\016\uffd2\017\uffd2\020\uffd2\021\uffd2" +
"\025\uffd2\026\uffd2\027\uffd2\032\uffd2\033\uffd2\060\uffd2\062" +
"\uffd2\065\uffd2\072\uffd2\001\002\000\044\007\uffd1\010\uffd1" +
"\012\uffd1\014\uffd1\016\uffd1\017\uffd1\020\uffd1\021\uffd1\025" +
"\uffd1\026\uffd1\027\uffd1\032\uffd1\033\uffd1\060\uffd1\062\uffd1" +
"\065\uffd1\072\uffd1\001\002\000\044\007\uffd0\010\uffd0\012" +
"\uffd0\014\uffd0\016\uffd0\017\uffd0\020\uffd0\021\uffd0\025\uffd0" +
"\026\uffd0\027\uffd0\032\uffd0\033\uffd0\060\uffd0\062\uffd0\065" +
"\uffd0\072\uffd0\001\002\000\044\007\uffce\010\uffce\012\uffce" +
"\014\uffce\016\uffce\017\uffce\020\uffce\021\uffce\025\uffce\026" +
"\uffce\027\uffce\032\uffce\033\uffce\060\uffce\062\uffce\065\uffce" +
"\072\uffce\001\002\000\044\007\uffcf\010\uffcf\012\uffcf\014" +
"\uffcf\016\uffcf\017\uffcf\020\uffcf\021\uffcf\025\uffcf\026\uffcf" +
"\027\uffcf\032\uffcf\033\uffcf\060\uffcf\062\uffcf\065\uffcf\072" +
"\uffcf\001\002\000\044\061\075\001\002\000\042\003\044" +
"\007\041\010\032\011\034\012\013\014\014\016\036\017" +
"\050\020\027\021\031\026\025\027\045\032\021\033\043" +
"\054\016\072\030\001\002\000\036\007\041\010\032\012" +
"\013\014\014\016\036\017\050\020\027\021\031\026\025" +
"\027\045\032\021\033\043\062\100\072\106\001\002\000" +
"\040\007\ufffa\010\ufffa\012\ufffa\014\ufffa\016\ufffa\017\ufffa" +
"\020\ufffa\021\ufffa\025\ufffa\026\ufffa\027\ufffa\032\ufffa\033" +
"\ufffa\062\ufffa\072\ufffa\001\002\000\040\007\uffc9\010\uffc9" +
"\012\uffc9\014\uffc9\016\uffc9\017\uffc9\020\uffc9\021\uffc9\025" +
"\uffc9\026\uffc9\027\uffc9\032\uffc9\033\uffc9\062\uffc9\072\uffc9" +
"\001\002\000\040\007\ufffe\010\ufffe\012\ufffe\014\ufffe\016" +
"\ufffe\017\ufffe\020\ufffe\021\ufffe\025\ufffe\026\ufffe\027\ufffe" +
"\032\ufffe\033\ufffe\062\ufffe\072\ufffe\001\002\000\040\007" +
"\ufff4\010\ufff4\012\ufff4\014\ufff4\016\ufff4\017\ufff4\020\ufff4" +
"\021\ufff4\025\ufff4\026\ufff4\027\ufff4\032\ufff4\033\ufff4\062" +
"\ufff4\072\ufff4\001\002\000\040\007\ufffc\010\ufffc\012\ufffc" +
"\014\ufffc\016\ufffc\017\ufffc\020\ufffc\021\ufffc\025\ufffc\026" +
"\ufffc\027\ufffc\032\ufffc\033\ufffc\062\ufffc\072\ufffc\001\002"
```



# MANUAL DE PRÁCTICAS



```
166      "\000\040\007\uffed\010\uffed\012\uffed\014\uffed\016\uffed\017" +
167      "\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed\032\uffed" +
168      "\033\uffed\062\uffed\072\uffed\001\002\000\040\007\uffed\010" +
169      "\uffed\012\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed" +
170      "\025\uffed\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072" +
171      "\uffed\001\002\000\020\035\116\036\117\037\115\040\121" +
172      "\041\123\042\120\043\122\001\002\000\040\007\uffed\010" +
173      "\uffed\012\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed" +
174      "\025\uffed\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072" +
175      "\uffed\001\002\000\040\007\uffed\010\uffed\014\uffed" +
176      "\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027" +
177      "\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001\002\000\040" +
178      "\007\uffed\010\uffed\012\uffed\014\uffed\016\uffed\017\uffed\020" +
179      "\uffed\021\uffed\025\uffed\026\uffed\027\uffed\032\uffed\033\uffed" +
180      "\062\uffed\072\uffed\001\002\000\040\007\uffed\010\uffed\012" +
181      "\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed" +
182      "\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001" +
183      "\002\000\040\007\uffed\020\010\uffed\021\uffed\014\uffed\016\uffed" +
184      "\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed\032" +
185      "\uffed\033\uffed\062\uffed\072\uffed\001\002\000\040\007\uffed" +
186      "\010\uffed\012\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021" +
187      "\uffed\025\uffed\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed" +
188      "\072\uffed\001\002\000\004\072\140\001\002\000\004\072" +
189      "\136\001\002\000\004\072\134\001\002\000\004\072\132" +
190      "\001\002\000\004\072\130\001\002\000\004\072\126\001" +
191      "\002\000\004\072\124\001\002\000\004\065\125\001\002" +
192      "\000\040\007\uffed\010\uffed\012\uffed\014\uffed\016\uffed\017" +
193      "\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed\032\uffed" +
194      "\033\uffed\062\uffed\072\uffed\001\002\000\004\065\127\001" +
195      "\002\000\040\007\uffed\010\uffed\012\uffed\014\uffed\016\uffed" +
196      "\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed\032" +
197      "\uffed\033\uffed\062\uffed\072\uffed\001\002\000\004\065\131" +
198      "\001\002\000\004\072\007\uffed\010\uffed\012\uffed\014\uffed\016" +
199      "\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed" +
200      "\032\uffed\033\uffed\062\uffed\072\uffed\001\002\000\004\065" +
201      "\133\001\002\000\040\007\uffed\010\uffed\012\uffed\014\uffed" +
202      "\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027" +
203      "\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001\002\000\004" +
204      "\065\135\001\002\000\040\007\uffed\010\uffed\012\uffed\014" +
205      "\uffed\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed" +
206      "\027\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001\002\000" +
207      "\004\065\137\001\002\000\040\007\uffed\010\uffed\012\uffed" +
208      "\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026" +
209      "\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001\002" +
210      "\000\004\065\141\001\002\000\040\007\uffed\010\uffed\012" +
211      "\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed" +
212      "\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001" +
213      "\002\000\004\044\143\001\002\000\004\073\144\001\002" +
214      "\000\004\065\145\001\002\000\040\007\uffed\010\uffed\012" +
215      "\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021\uffed\025\uffed" +
216      "\026\uffed\027\uffed\032\uffed\033\uffed\062\uffed\072\uffed\001" +
217      "\002\000\040\007\uffed\010\uffed\012\uffed\014\uffed\016\uffed" +
218      "\017\uffed\040\020\uffed\041\uffed\025\uffed\042\uffed\026\uffed" +
219      "\uffed\033\uffed\062\uffed\072\uffed\001\002\000\040\007\uffed" +
220      "\010\uffed\051\012\uffed\050\014\uffed\051\016\uffed\051\017\uffed" +
221      "\052\uffed\053\025\uffed\054\026\uffed\055\027\uffed\056\032\uffed" +
222      "\057\001\002\000\004\072\151\001\002\000\004\060" +
223      "\152\001\002\000\004\065\153\001\002\000\040\007\uffed" +
224      "\010\uffed\012\uffed\014\uffed\016\uffed\017\uffed\020\uffed\021" +
225      "\uffed\025\uffed\026\uffed\027\uffed\028\uffed\032\uffed\033\uffed" +
226      "\072\uffed\001\002\000\012\044\160\055\155\056\156\065" +
227      "\157\001\002\000\004\065\164\001\002\000\004\065\163" +
228      "\001\002\000\040\007\uffed\010\uffed\012\uffed\014\uffed\016" +
229      "\uffed\017\uffed\020\uffed\021\uffed\025\uffed\026\uffed\027\uffed" +
230      "\032\uffed\033\uffed\062\uffed\072\uffed\001\002\000\004\073" +
231      "\161\001\002\000\004\065\162\001\002\000\040\007\uffed"
```



# MANUAL DE PRÁCTICAS



```
232      "\010\uffe2\012\uffe2\014\uffe2\016\uffe2\017\uffe2\020\uffe2\021" +
233      "\uffe2\025\uffe2\026\uffe2\027\uffe2\032\uffe2\033\uffe2\062\uffe2" +
234      "\072\uffe2\001\002\000\040\007\uffdf\010\uffdf\012\uffdf\014" +
235      "\uffdf\016\uffdf\017\uffdf\020\uffdf\021\uffdf\025\uffdf\026\uffdf" +
236      "\027\uffdf\032\uffdf\033\uffdf\062\uffdf\072\uffdf\001\002\000" +
237      "\040\007\uffe0\010\uffe0\012\uffe0\014\uffe0\016\uffe0\017\uffe0" +
238      "\020\uffe0\021\uffe0\025\uffe0\026\uffe0\027\uffe0\032\uffe0\033" +
239      "\uffe0\062\uffe0\072\uffe0\001\002\000\006\007\167\072\170" +
240      "\001\002\000\004\060\214\001\002\000\004\072\205\001" +
241      "\002\000\004\044\171\001\002\000\004\073\172\001\002" +
242      "\000\004\065\173\001\002\000\010\011\034\054\016\072" +
243      "\053\001\002\000\004\065\175\001\002\000\006\055\177" +
244      "\072\200\001\002\000\004\060\uffc5\001\002\000\004\072" +
245      "\024\001\002\000\006\044\202\055\201\001\002\000\004" +
246      "\060\uffc3\001\002\000\004\073\203\001\002\000\004\060" +
247      "\uffc4\001\002\000\004\060\uffc2\001\002\000\004\044\206" +
248      "\001\002\000\004\073\207\001\002\000\004\065\210\001" +
249      "\002\000\010\011\034\054\016\072\053\001\002\000\004" +
250      "\065\212\001\002\000\006\055\177\072\200\001\002\000" +
251      "\004\060\uffc6\001\002\000\004\061\215\001\002\000\042" +
252      "\003\044\007\041\010\032\011\034\012\013\014\016" +
253      "\036\017\050\020\027\021\031\026\025\027\045\032\021" +
254      "\033\043\054\016\072\030\001\002\000\036\007\041\010" +
255      "\032\012\013\014\014\016\036\017\050\020\027\021\031" +
256      "\026\025\027\045\032\021\033\043\062\217\072\106\001" +
257      "\002\000\040\007\uffc7\010\uffc7\012\uffc7\014\uffc7\016\uffc7" +
258      "\017\uffc7\020\uffc7\021\uffc7\025\uffc7\026\uffc7\027\uffc7\032" +
259      "\uffc7\033\uffc7\062\uffc7\072\uffc7\001\002\000\004\044\221" +
260      "\001\002\000\004\073\222\001\002\000\004\070\223\001" +
261      "\002\000\004\073\224\001\002\000\004\065\225\001\002" +
262      "\000\040\007\uffe1\010\uffe1\012\uffe1\014\uffe1\016\uffe1\017" +
263      "\uffe1\020\uffe1\021\uffe1\025\uffe1\026\uffe1\027\uffe1\032\uffe1" +
264      "\033\uffe1\062\uffe1\072\uffe1\001\002\000\004\072\227\001" +
265      "\002\000\004\060\230\001\002\000\004\061\231\001\002" +
266      "\000\004\022\233\001\002\000\006\022\241\024\242\001" +
267      "\002\000\004\007\234\001\002\000\004\071\235\001\002" +
268      "\000\042\003\044\007\041\010\032\011\034\012\013\014" +
269      "\014\016\036\017\050\020\027\021\031\026\025\027\045" +
270      "\032\021\033\043\054\016\072\030\001\002\000\036\007" +
271      "\041\010\032\012\013\014\014\016\036\017\050\020\027" +
272      "\021\031\025\237\026\025\027\045\032\021\033\043\072" +
273      "\066\001\002\000\004\065\240\001\002\000\006\022\uffc0" +
274      "\024\uffc0\001\002\000\004\007\250\001\002\000\004\071" +
275      "\243\001\002\000\042\003\044\007\041\010\032\011\034" +
276      "\012\013\014\014\016\036\017\050\020\027\021\031\026" +
277      "\025\027\045\032\021\033\043\054\016\072\030\001\002" +
278      "\000\036\007\041\010\032\012\013\014\014\016\036\017" +
279      "\050\020\027\021\031\025\245\026\025\027\045\032\021" +
280      "\033\043\072\106\001\002\000\004\065\246\001\002\000" +
281      "\004\062\247\001\002\000\040\007\uffc1\010\uffc1\012\uffc1" +
282      "\014\uffc1\016\uffc1\017\uffc1\020\uffc1\021\uffc1\025\uffc1\026" +
283      "\uffc1\027\uffc1\032\uffc1\033\uffc1\062\uffc1\072\uffc1\001\002" +
284      "\000\004\071\251\001\002\000\042\003\044\007\041\010" +
285      "\032\011\034\012\013\014\014\016\036\017\050\020\027" +
286      "\021\031\026\025\027\045\032\021\033\043\054\016\072" +
287      "\030\001\002\000\036\007\041\010\032\012\013\014\014" +
288      "\016\036\017\050\020\027\021\031\025\253\026\025\027" +
289      "\045\032\021\033\043\072\106\001\002\000\004\065\254" +
290      "\001\002\000\006\022\uffbf\024\uffbf\001\002\000\042\003" +
291      "\044\007\041\010\032\011\034\012\013\014\014\016\036" +
292      "\017\050\020\027\021\031\026\025\027\045\032\021\033" +
293      "\043\054\016\072\030\001\002\000\036\007\041\010\032" +
294      "\012\013\014\014\016\036\017\050\020\027\021\031\026" +
295      "\025\027\045\032\021\033\043\062\257\072\106\001\002" +
296      "\000\004\017\260\001\002\000\004\057\261\001\002\000" +
297      "\010\011\034\054\016\072\053\001\002\000\004\060\263"
```



# MANUAL DE PRÁCTICAS



```
298      "\\"001\\002\\000\\004\\065\\264\\001\\002\\000\\040\\007\\uffc8\\010" +
299      "\uffc8\\012\\uffc8\\014\\uffc8\\016\\uffc8\\017\\uffc8\\020\\uffc8\\021\\uffc8" +
300      "\\"025\\uffc8\\026\\uffc8\\027\\uffc8\\032\\uffc8\\033\\uffc8\\062\\uffc8\\072" +
301      "\uffc8\\001\\002\\000\\004\\044\\266\\001\\002\\000\\004\\073\\267" +
302      "\\"001\\002\\000\\004\\065\\270\\001\\002\\000\\040\\007\\uffbd\\010" +
303      "\uffbd\\012\\uffbd\\014\\uffbd\\016\\uffbd\\017\\uffbd\\020\\uffbd\\021\\uffbd" +
304      "\\"025\\uffbd\\026\\uffbd\\027\\uffbd\\032\\uffbd\\033\\uffbd\\062\\uffbd\\072" +
305      "\uffbd\\001\\002\\000\\004\\072\\272\\001\\002\\000\\004\\060\\273" +
306      "\\"001\\002\\000\\004\\065\\274\\001\\002\\000\\040\\007\\uffbb\\010" +
307      "\uffbb\\012\\uffbb\\014\\uffbb\\016\\uffbb\\017\\uffbb\\020\\uffbb\\021\\uffbb" +
308      "\\"025\\uffbb\\026\\uffbb\\027\\uffbb\\032\\uffbb\\033\\uffbb\\062\\uffbb\\072" +
309      "\uffbb\\001\\002\\000\\004\\002\\000\\001\\002\\000\\044\\007\\uffcb" +
310      "\\"010\\uffcb\\012\\uffcb\\014\\uffcb\\016\\uffcb\\017\\uffcb\\020\\uffcb\\021" +
311      "\uffcb\\025\\uffcb\\026\\uffcb\\027\\uffcb\\032\\uffcb\\033\\uffcb\\060\\uffcb" +
312      "\\"062\\uffcb\\065\\uffcb\\072\\uffcb\\001\\002\\000\\010\\011\\034\\054" +
313      "\\"016\\072\\053\\001\\002\\000\\004\\060\\301\\001\\002\\000\\004" +
314      "\\"061\\302\\001\\002\\000\\042\\003\\044\\007\\041\\010\\032\\011" +
315      "\\"034\\012\\013\\014\\014\\016\\036\\017\\050\\020\\027\\021\\031" +
316      "\\"026\\025\\027\\045\\032\\021\\033\\043\\054\\016\\072\\030\\001" +
317      "\\"002\\000\\036\\007\\041\\010\\032\\012\\013\\014\\014\\016\\036" +
318      "\\"017\\050\\020\\027\\021\\031\\026\\025\\027\\045\\032\\021\\033" +
319      "\\"043\\062\\304\\072\\106\\001\\002\\000\\042\\007\\uffd6\\010\\uffd6" +
320      "\\"012\\uffd6\\014\\uffd6\\015\\305\\016\\uffd6\\017\\uffd6\\020\\uffd6\\021" +
321      "\uffd6\\025\\uffd6\\026\\uffd6\\027\\uffd6\\032\\uffd6\\033\\uffd6\\062\\uffd6" +
322      "\\"072\\uffd6\\001\\002\\000\\004\\061\\306\\001\\002\\000\\042\\003" +
323      "\\"044\\007\\041\\010\\032\\011\\034\\012\\013\\014\\014\\016\\036" +
324      "\\"017\\050\\020\\027\\021\\031\\026\\025\\027\\045\\032\\021\\033" +
325      "\\"043\\054\\016\\072\\030\\001\\002\\000\\036\\007\\041\\010\\032" +
326      "\\"012\\013\\014\\014\\016\\036\\017\\050\\020\\027\\021\\031\\026" +
327      "\\"025\\027\\045\\032\\021\\033\\043\\062\\310\\072\\106\\001\\002" +
328      "\\"000\\040\\007\\uffca\\010\\uffca\\012\\uffca\\014\\uffca\\016\\uffca\\017" +
329      "\uffca\\020\\021\\uffca\\025\\uffca\\026\\uffca\\027\\uffca\\032\\uffca" +
330      "\\"033\\uffca\\062\\uffca\\072\\uffca\\001\\002\\000\\004\\044\\312\\001" +
331      "\\"002\\000\\004\\005\\313\\001\\002\\000\\004\\072\\314\\001\\002" +
332      "\\"000\\004\\005\\315\\001\\002\\000\\004\\065\\316\\001\\002\\000" +
333      "\\"040\\007\\uffde\\010\\uffde\\012\\uffde\\014\\uffde\\016\\uffde\\017\\uffde" +
334      "\\"020\\uffde\\021\\uffde\\025\\uffde\\026\\uffde\\027\\uffde\\032\\uffde\\033" +
335      "\uffde\\062\\uffde\\072\\uffde\\001\\002\\000\\004\\002\\001\\001\\002" +
336      "\\"000\\004\\002\\uffff\\001\\002" );
337
338     /** Access to parse-action table. */
339     public short[][] action_table() { return _action_table; }
340
341     /** <code>reduce_goto</code> table. */
342     protected static final short[][] _reduce_table =
343         unpackFromStrings(new String[] {
344             "\\"000\\316\\000\\004\\002\\004\\001\\001\\000\\002\\001\\001\\000" +
345             "\\"002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002" +
346             "\'001\\001\\000\\002\\001\\001\\000\\036\\003\\017\\004\\016\\006" +
347             "\'023\\007\\014\\010\\034\\011\\025\\012\\022\\013\\041\\015\\045" +
348             "\'016\\037\\021\\036\\022\\046\\023\\021\\024\\032\\001\\001\\000" +
349             "\'002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002" +
350             "\'001\\001\\000\\002\\001\\001\\000\\032\\004\\100\\006\\102\\007" +
351             "\'076\\010\\107\\011\\104\\012\\101\\015\\112\\016\\110\\021\\111" +
352             "\'022\\113\\023\\103\\024\\106\\001\\001\\000\\002\\001\\001\\000" +
353             "\'002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002" +
354             "\'001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001" +
355             "\'001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001" +
356             "\'000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000" +
357             "\'002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002" +
358             "\'001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001" +
359             "\'001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001" +
360             "\'000\\004\\013\\051\\001\\001\\001\\000\\002\\001\\001\\000\\002\\001" +
361             "\'001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001" +
362             "\'000\\002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000" +
363             "\'002\\001\\001\\000\\002\\001\\001\\000\\002\\001\\001\\000\\002" +
```



# MANUAL DE PRÁCTICAS



```
364      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
365      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
366      "\000\002\001\001\000\036\003\075\004\016\006\023\007" +
367      "\014\010\034\011\025\012\022\013\041\015\045\016\037" +
368      "\021\036\022\046\023\021\024\032\001\001\000\032\004" +
369      "\100\006\102\007\076\010\107\011\104\012\101\015\112" +
370      "\016\110\021\111\022\113\023\103\024\106\001\001\000" +
371      "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
372      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
373      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
374      "\000\002\001\001\000\002\001\001\000\002\001\000" +
375      "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
376      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
377      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
378      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
379      "\002\001\001\000\002\001\001\000\002\001\000\002" +
380      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
381      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
382      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
383      "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
384      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
385      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
386      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
387      "\002\001\001\000\002\001\001\000\004\014\165\001\001" +
388      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
389      "\002\001\001\000\002\001\001\000\004\013\173\001\001" +
390      "\000\002\001\001\000\004\005\175\001\001\000\002\001" +
391      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
392      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
393      "\002\001\001\000\002\001\001\000\002\001\001\000\004" +
394      "\013\210\001\001\000\002\001\001\000\004\005\212\001" +
395      "\001\000\002\001\001\000\002\001\001\000\036\003\215" +
396      "\004\016\006\023\007\014\010\034\011\025\012\022\013" +
397      "\041\015\045\016\037\021\036\022\046\023\021\024\032" +
398      "\001\001\000\032\004\100\006\102\007\076\010\107\011" +
399      "\104\012\101\015\112\016\110\021\111\022\113\023\103" +
400      "\024\106\001\001\000\002\001\001\000\002\001\001\000" +
401      "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
402      "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
403      "\001\000\002\001\001\000\004\017\231\001\001\000\002" +
404      "\001\001\000\002\001\001\000\002\001\001\000\036\003" +
405      "\235\004\016\006\023\007\014\010\034\011\025\012\022" +
406      "\013\041\015\045\016\037\021\036\022\046\023\021\024" +
407      "\032\001\001\000\032\004\100\006\102\007\076\010\107" +
408      "\011\104\012\101\015\112\016\110\021\111\022\113\023" +
409      "\024\106\001\001\000\002\001\001\000\002\001\001\001" +
410      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
411      "\016\006\023\007\014\010\034\011\025\012\022\013\041" +
412      "\015\045\016\037\021\036\022\046\023\021\024\032\001" +
413      "\001\000\032\004\100\006\102\007\076\010\107\011\104" +
414      "\012\101\015\112\016\110\021\111\022\113\023\103\024" +
415      "\006\001\001\000\002\001\001\000\002\001\001\000\002" +
416      "\001\001\000\002\001\001\000\036\003\251\004\016\006" +
417      "\023\007\014\010\034\011\025\012\022\013\041\015\045" +
418      "\016\037\021\036\022\046\023\021\024\032\001\001\000" +
419      "\032\004\100\006\102\007\076\010\107\011\104\012\101" +
420      "\015\112\016\110\021\111\022\113\023\103\024\106\001" +
421      "\001\000\002\001\001\000\002\001\001\000\036\003\255" +
422      "\004\016\006\023\007\014\010\034\011\025\012\022\013" +
423      "\041\015\045\016\037\021\036\022\046\023\021\024\032" +
424      "\001\001\000\032\004\100\006\102\007\076\010\107\011" +
425      "\014\012\101\015\112\016\110\021\111\022\113\023\103" +
426      "\024\106\001\001\000\002\001\001\000\002\001\001\000" +
427      "\004\013\261\001\001\000\002\001\001\000\002\001\001" +
428      "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
429      "\002\001\001\000\002\001\001\000\002\001\001\000\002"
```



```
430      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
431      "\001\000\002\001\001\000\004\013\277\001\001\000\002" +
432      "\001\001\000\002\001\001\000\036\003\302\004\016\006" +
433      "\023\007\014\010\034\011\025\012\022\013\041\015\045" +
434      "\016\037\021\036\022\046\023\021\024\032\001\001\000" +
435      "\032\004\100\006\102\007\076\010\107\011\104\012\101" +
436      "\015\112\016\110\021\111\022\113\023\103\024\106\001" +
437      "\001\000\002\001\001\000\002\001\001\000\036\003\306" +
438      "\004\016\006\023\007\014\010\034\011\025\012\022\013" +
439      "\041\015\045\016\037\021\036\022\046\023\021\024\032" +
440      "\001\001\000\002\001\000\004\100\006\102\007\076\010\107\011" +
441      "\104\012\101\015\112\016\110\021\111\022\113\023\103" +
442      "\024\106\001\001\000\002\001\001\000\002\001\001\000" +
443      "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
444      "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
445      "\001" );
446
447  /** Access to <code>reduce_goto</code> table. */
448  public short[] reduce_table() {return _reduce_table;}
449
450  /** Instance of action encapsulation class. */
451  protected CUP$Syntax$actions action_obj;
452
453  /** Action encapsulation object initializer. */
454  protected void init_actions()
455  {
456      action_obj = new CUP$Syntax$actions(this);
457  }
458
459  /** Invoke a user supplied parse action. */
460  public java_cup.runtime.Symbol do_action(
461      int act_num,
462      java_cup.runtime.lr_parser parser,
463
464      java.util.Stack stack,
465      int top)
466  throws java.lang.Exception
467  {
468      /* call code in generated class */
469      return action_obj.CUP$Syntax$do_action(act_num, parser, stack, top);
470  }
471
472  /** Indicates start state. */
473  public int start_state() {return 0;}
474  /** Indicates start production. */
475  public int start_production() {return 0;}
476
477  /** <code>EOF</code> Symbol index. */
478  public int EOF_sym() {return 0;}
479
480  /** <code>error</code> Symbol index. */
481  public int error_sym() {return 1;}
482
483
484  private java.util.List<String> errores = new java.util.ArrayList<>();
485  private String textoCompleto; // Para calcular las columnas por linea
486
487  public void syntax_error(Symbol s) {
488      int linea = s.right + 1;
489      int columnaReal = calcularColumnaReal(linea, s.left + 1);
490
491      errores.add("Error de sintaxis. Linea: " + linea +
492                  ", Columna: " + columnaReal + ", Texto: \\" + s.value + "\\");
493  }
494
495  public java.util.List<String> getErrores() {
```



## **MANUAL DE PRACTICAS**





# MANUAL DE PRÁCTICAS



```
562     | CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_IMPRIMIR",17, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(17)));
563     |
564     | return CUP$Syntax$result;
565
566     /* . . . . . */
567     case 69: // SENTENCIA_VARIABLE ::= variable Identificador asignacion Numero finLinea
568     {
569         Object RESULT =null;
570
571         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_VARIABLE",16, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(16)));
572     }
573     return CUP$Syntax$result;
574
575     /* . . . . . */
576     case 68: // SENTENCIA_CONSTANTE ::= constante Identificador asignacion Numero finLinea
577     {
578         Object RESULT =null;
579
580         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_CONSTANTE",15, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(15)));
581     }
582     return CUP$Syntax$result;
583
584     /* . . . . . */
585     case 67: // FUNCION_SALIR_CASE ::= funcionesSalirCase finLinea
586     {
587         Object RESULT =null;
588
589         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FUNCION_SALIR_CASE",14, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(14)));
590     }
591     return CUP$Syntax$result;
592
593     /* . . . . . */
594     case 66: // FUNCION_CASE ::= FUNCION_CASE_funcionCase valorEntero dosPuntos SENTENCIA detener finLinea
595     {
596         Object RESULT =null;
597
598         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FUNCION_CASE",13, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(13)));
599     }
600     return CUP$Syntax$result;
601
602     /* . . . . . */
603     case 65: // FUNCION_CASE ::= funcionCase valorEntero dosPuntos SENTENCIA detener finLinea
604     {
605         Object RESULT =null;
606
607         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FUNCION_CASE",13, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(13)));
608     }
609     return CUP$Syntax$result;
610
611     /* . . . . . */
612     case 64: // FUNCION_SWITCH ::= funcionesSwitch parentIzq Identificador parentDer llaveIzq FUNCION_CASE predeterminado dosPuntos SENTENCIA
613     {
614         Object RESULT =null;
615
616         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FUNCION_SWITCH",12, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(12)));
617     }
618     return CUP$Syntax$result;
619
620     /* . . . . . */
621     case 63: // DECLARACION_FOR ::= incremento Identificador
622     {
623         Object RESULT =null;
624
625         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION_FOR",3, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(3)));
626     }
627     return CUP$Syntax$result;
```



# MANUAL DE PRÁCTICAS



```
628
629      /* . . . . . */
630      case 62: // DECLARACION_FOR ::= Identificador incremento
631      {
632          Object RESULT =null;
633
634          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION_FOR",3, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
635      }
636      return CUP$Sintax$result;
637
638      /* . . . . . */
639      case 61: // DECLARACION_FOR ::= Identificador asignacion Numero
640      {
641          Object RESULT =null;
642
643          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION_FOR",3, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
644      }
645      return CUP$Sintax$result;
646
647      /* . . . . . */
648      case 60: // SENTENCIA_FOR ::= Identificador asignacion Numero finLinea SENTENCIA_BOLEANA finLinea DECLARACION_FOR
649      {
650          Object RESULT =null;
651
652          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
653      }
654      return CUP$Sintax$result;
655
656      /* . . . . . */
657      case 59: // SENTENCIA_FOR ::= valorEntero Identificador asignacion Numero finLinea SENTENCIA_BOLEANA finLinea DECLARACION_FOR
658      {
659          Object RESULT =null;
660
661          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
662      }
663      return CUP$Sintax$result;
664
665      /* . . . . . */
666      case 58: // CICLO_FOR ::= cicloFor parentIzq SENTENCIA_FOR parentDer llaveIzq SENTENCIA llaveDer
667      {
668          Object RESULT =null;
669
670          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("CICLO_FOR",8, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
671      }
672      return CUP$Sintax$result;
673
674      /* . . . . . */
675      case 57: // CICLO_DO WHILE ::= cicloDo llaveIzq SENTENCIA llaveDer cicloWhile parentIzq SENTENCIA_BOLEANA parentDer finLinea
676      {
677          Object RESULT =null;
678
679          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("CICLO_DO WHILE",7, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
680      }
681      return CUP$Sintax$result;
682
683      /* . . . . . */
684      case 56: // CICLO_WHILE ::= cicloWhile parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer
685      {
686          Object RESULT =null;
687
688          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("CICLO WHILE",6, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.getTopIndex()-1)));
689      }
690      return CUP$Sintax$result;
691
692      /* . . . . . */
693      case 55: // CONDICIONAL_IF ELSE ::= condicionalIf parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer condicionalElse llaveDer
```



```
694     {
695         Object RESULT =null;
696
697         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("CONDICIONAL_IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(5));
698     }
699     return CUP$Sintax$result;
700
701     /*. . . . . */
702     case 54: // SENTENCIA_BOLEANA ::= operadorNo Identificador
703     {
704         Object RESULT =null;
705
706         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
707     }
708     return CUP$Sintax$result;
709
710     /*. . . . . */
711     case 53: // SENTENCIA_BOLEANA ::= Identificador operadorO Identificador
712     {
713         Object RESULT =null;
714
715         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
716     }
717     return CUP$Sintax$result;
718
719     /*. . . . . */
720     case 52: // SENTENCIA_BOLEANA ::= Identificador operadorY Identificador
721     {
722         Object RESULT =null;
723
724         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
725     }
726     return CUP$Sintax$result;
727
728     /*. . . . . */
729     case 51: // SENTENCIA_BOLEANA ::= Identificador menorIgual Numero
730     {
731         Object RESULT =null;
732
733         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
734     }
735     return CUP$Sintax$result;
736
737     /*. . . . . */
738     case 50: // SENTENCIA_BOLEANA ::= Identificador mayorIgual Numero
739     {
740         Object RESULT =null;
741
742         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
743     }
744     return CUP$Sintax$result;
745
746     /*. . . . . */
747     case 49: // SENTENCIA_BOLEANA ::= Identificador menorQue Numero
748     {
749         Object RESULT =null;
750
751         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(9));
752     }
753     return CUP$Sintax$result;
754
755     /*. . . . . */
756     case 48: // SENTENCIA_BOLEANA ::= Identificador mayorQue Numero
757     {
758         Object RESULT =null;
```



# MANUAL DE PRÁCTICAS



```
760 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA", 9, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(9)));
761 | }
762 | return CUP$Syntax$result;
763 |
764 | /* . . . . . */
765 | case 47: // SENTENCIA_BOLEANA ::= Identificador comparacionIgualdad Identificador
766 | {
767 |     Object RESULT =null;
768 |
769 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA", 9, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(9)));
770 | }
771 | return CUP$Syntax$result;
772 |
773 | /* . . . . . */
774 | case 46: // SENTENCIA_BOLEANA ::= Identificador comparacionIgualdad Numero
775 | {
776 |     Object RESULT =null;
777 |
778 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA", 9, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(9)));
779 | }
780 | return CUP$Syntax$result;
781 |
782 | /* . . . . . */
783 | case 45: // SENTENCIA_BOLEANA ::= Identificador comparacionIgualdad valorBooleano
784 | {
785 |     Object RESULT =null;
786 |
787 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA", 9, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(9)));
788 | }
789 | return CUP$Syntax$result;
790 |
791 | /* . . . . . */
792 | case 44: // SENTENCIA_BOLEANA ::= valorBooleano
793 | {
794 |     Object RESULT =null;
795 |
796 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA", 9, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()));
797 | }
798 | return CUP$Syntax$result;
799 |
800 | /* . . . . . */
801 | case 43: // CONDICIONAL_IF ::= condicionalIf parentIzq SENTENCIA_BOLEANA parentDer llaveIzq SENTENCIA llaveDer
802 | {
803 |     Object RESULT =null;
804 |
805 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("CONDICIONAL_IF", 4, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(4)));
806 | }
807 | return CUP$Syntax$result;
808 |
809 | /* . . . . . */
810 | case 42: // OPERACION_ARITMETICA ::= Identificador operadorRaiz Identificador finLinea
811 | {
812 |     Object RESULT =null;
813 |
814 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA", 11, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(11)));
815 | }
816 | return CUP$Syntax$result;
817 |
818 | /* . . . . . */
819 | case 41: // OPERACION_ARITMETICA ::= Identificador operadorPotencia Identificador finLinea
820 | {
821 |     Object RESULT =null;
822 |
823 |     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA", 11, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(11)));
824 | }
825 | return CUP$Syntax$result;
```



# MANUAL DE PRÁCTICAS



```
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
```

/\* . . . . . \*/  
case 40: // OPERACION\_ARITMETICA ::= Identificador operadorModulo Identificador finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("OPERACION\_ARITMETICA",11, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 39: // OPERACION\_ARITMETICA ::= Identificador operadorDivision Identificador finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("OPERACION\_ARITMETICA",11, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 38: // OPERACION\_ARITMETICA ::= Identificador operadorMultiplicacion Identificador finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("OPERACION\_ARITMETICA",11, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 37: // OPERACION\_ARITMETICA ::= Identificador operadorResta Identificador finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("OPERACION\_ARITMETICA",11, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 36: // OPERACION\_ARITMETICA ::= Identificador operadorSuma Identificador finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("OPERACION\_ARITMETICA",11, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 35: // DECLARACION ::= cadena Identificador asignacion comillaDoble Identificador comillaDoble finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 34: // DECLARACION ::= valorEntero Identificador decreemento finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}  
return CUP\$Syntax\$result;  
  
/\* . . . . . \*/  
case 33: // DECLARACION ::= valorEntero Identificador incremento finLinea  
{  
 Object RESULT =null;  
  
 CUP\$Syntax\$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java\_cup.runtime.Symbol)CUP\$Syntax\$stack.elementAt(CUP\$Syntax\$stack.getTopIndex()-1)));  
}



```
892     {
893         Object RESULT =null;
894
895         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
896         )
897         return CUP$Sintax$result;
898
899         /*. . . . . */
900         case 32: // DECLARACION ::= valorFlotante Identificador asignacion Numero puntoDecimal Numero finLinea
901         {
902             Object RESULT =null;
903
904             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
905             )
906             return CUP$Sintax$result;
907
908         /*. . . . . */
909         case 31: // DECLARACION ::= valorEntero Identificador asignacion Numero finLinea
910         {
911             Object RESULT =null;
912
913             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
914             )
915             return CUP$Sintax$result;
916
917         /*. . . . . */
918         case 30: // DECLARACION ::= valorEntero Identificador finLinea
919         {
920             Object RESULT =null;
921
922             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
923             )
924             return CUP$Sintax$result;
925
926         /*. . . . . */
927         case 29: // SENTENCIA ::= error llaveDer
928         {
929             Object RESULT =null;
930             System.err.println("Error en SENTENCIA");
931             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
932             )
933             return CUP$Sintax$result;
934
935         /*. . . . . */
936         case 28: // SENTENCIA ::= error finLinea
937         {
938             Object RESULT =null;
939             System.err.println("Error en SENTENCIA");
940             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$S
941             )
942             return CUP$Sintax$result;
943
944         /*. . . . . */
945         case 27: // SENTENCIA ::= SENTENCIA_VARIABLE
946         {
947             Object RESULT =null;
948
949             CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_c
950             )
951             return CUP$Sintax$result;
952
953         /*. . . . . */
954         case 26: // SENTENCIA ::= SENTENCIA SENTENCIA_VARIABLE
955         {
956             Object RESULT =null;
```



```
958     | CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)));
959     |
960     | return CUP$Syntax$result;
961
962     /*. . . . . */
963     case 25: // SENTENCIA ::= SENTENCIA_CONSTANTE
964     {
965         Object RESULT =null;
966
967         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)));
968     }
969     return CUP$Syntax$result;
970
971     /*. . . . . */
972     case 24: // SENTENCIA ::= SENTENCIA SENTENCIA_CONSTANTE
973     {
974         Object RESULT =null;
975
976         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 2)));
977     }
978     return CUP$Syntax$result;
979
980     /*. . . . . */
981     case 23: // SENTENCIA ::= SENTENCIA_LEER
982     {
983         Object RESULT =null;
984
985         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)));
986     }
987     return CUP$Syntax$result;
988
989     /*. . . . . */
990     case 22: // SENTENCIA ::= SENTENCIA SENTENCIA LEER
991     {
992         Object RESULT =null;
993
994         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 2)));
995     }
996     return CUP$Syntax$result;
997
998     /*. . . . . */
999     case 21: // SENTENCIA ::= SENTENCIA_IMPRIMIR
1000    {
1001        Object RESULT =null;
1002
1003        CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)));
1004    }
1005    return CUP$Syntax$result;
1006
1007    /*. . . . . */
1008    case 20: // SENTENCIA ::= SENTENCIA SENTENCIA_IMPRIMIR
1009    {
1010        Object RESULT =null;
1011
1012        CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 2)));
1013    }
1014    return CUP$Syntax$result;
1015
1016    /*. . . . . */
1017    case 19: // SENTENCIA ::= FUNCION_SWITCH
1018    {
1019        Object RESULT =null;
1020
1021        CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Sintax$stack.size() - 1)));
1022    }
1023    return CUP$Syntax$result;
```



```
1024
1025         /*. . . . . */
1026     case 18: // SENTENCIA ::= SENTENCIA_FUNCION_SWITCH
1027     {
1028         Object RESULT =null;
1029
1030         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1031     }
1032     return CUP$Sintax$result;
1033
1034     /*. . . . . */
1035     case 17: // SENTENCIA ::= SENTENCIA_BOLEANA
1036     {
1037         Object RESULT =null;
1038
1039         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1040     }
1041     return CUP$Sintax$result;
1042
1043     /*. . . . . */
1044     case 16: // SENTENCIA ::= OPERACION_ARITMETICA
1045     {
1046         Object RESULT =null;
1047
1048         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1049     }
1050     return CUP$Sintax$result;
1051
1052     /*. . . . . */
1053     case 15: // SENTENCIA ::= SENTENCIA_OPERACION_ARITMETICA
1054     {
1055         Object RESULT =null;
1056
1057         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1058     }
1059     return CUP$Sintax$result;
1060
1061     /*. . . . . */
1062     case 14: // SENTENCIA ::= CICLO_FOR
1063     {
1064         Object RESULT =null;
1065
1066         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1067     }
1068     return CUP$Sintax$result;
1069
1070     /*. . . . . */
1071     case 13: // SENTENCIA ::= SENTENCIA_CICLO_FOR
1072     {
1073         Object RESULT =null;
1074
1075         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1076     }
1077     return CUP$Sintax$result;
1078
1079     /*. . . . . */
1080     case 12: // SENTENCIA ::= CICLO_DO WHILE
1081     {
1082         Object RESULT =null;
1083
1084         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1085     }
1086     return CUP$Sintax$result;
1087
1088     /*. . . . . */
1089     case 11: // SENTENCIA ::= SENTENCIA_CICLO_DO WHILE
```



```
1090  {
1091      Object RESULT =null;
1092
1093      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1094  }
1095  return CUP$Sintax$result;
1096
1097  /*. . . . . */
1098  case 10: // SENTENCIA ::= CICLO WHILE
1099  {
1100      Object RESULT =null;
1101
1102      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1103  }
1104  return CUP$Sintax$result;
1105
1106  /*. . . . . */
1107  case 9: // SENTENCIA ::= SENTENCIA CICLO WHILE
1108  {
1109      Object RESULT =null;
1110
1111      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1112  }
1113  return CUP$Sintax$result;
1114
1115  /*. . . . . */
1116  case 8: // SENTENCIA ::= CONDICIONAL_IF_ELSE
1117  {
1118      Object RESULT =null;
1119
1120      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1121  }
1122  return CUP$Sintax$result;
1123
1124  /*. . . . . */
1125  case 7: // SENTENCIA ::= SENTENCIA CONDICIONAL_IF_ELSE
1126  {
1127      Object RESULT =null;
1128
1129      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1130  }
1131  return CUP$Sintax$result;
1132
1133  /*. . . . . */
1134  case 6: // SENTENCIA ::= CONDICIONAL_IF
1135  {
1136      Object RESULT =null;
1137
1138      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1139  }
1140  return CUP$Sintax$result;
1141
1142  /*. . . . . */
1143  case 5: // SENTENCIA ::= SENTENCIA CONDICIONAL_IF
1144  {
1145      Object RESULT =null;
1146
1147      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1148  }
1149  return CUP$Sintax$result;
1150
1151  /*. . . . . */
1152  case 4: // SENTENCIA ::= DECLARACION
1153  {
1154      Object RESULT =null;
1155  }
```



```
1156     |     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1157     |
1158     |     return CUP$Sintax$result;
1159
1160     /* . . . . . */
1161     case 3: // SENTENCIA ::= SENTENCIA DECLARACION
1162     {
1163         Object RESULT =null;
1164
1165         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1166     }
1167     return CUP$Sintax$result;
1168
1169     /* . . . . . */
1170     case 2: // INICIO ::= error llaveDer
1171     {
1172         Object RESULT =null;
1173         System.err.println("Error en SENTENCIA");
1174         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1175     }
1176     return CUP$Sintax$result;
1177
1178     /* . . . . . */
1179     case 1: // INICIO ::= clase Identificador parentIzq parentDer llaveIzq SENTENCIA llaveDer
1180     {
1181         Object RESULT =null;
1182
1183         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1184     }
1185     return CUP$Sintax$result;
1186
1187     /* . . . . . */
1188     case 0: // $START ::= INICIO EOF
1189     {
1190         Object RESULT =null;
1191         int start_valleft = ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).left;
1192         int start_valright = ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).right;
1193         Object start_val = (Object)((java_cup.runtime.Symbol) CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).value;
1194         RESULT = start_val;
1195         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("$START",0, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stack.size()-1)));
1196     }
1197     /* ACCEPT */
1198     CUP$Sintax$parser.done_parsing();
1199     return CUP$Sintax$result;
1200
1201     /* . . . . . */
1202     default:
1203         throw new Exception(
1204             "Invalid action number found in internal parse table");
1205
1206     }
1207 }
```

### Archivo sym.java (Generado automáticamente)

El archivo sym.java es generado automáticamente por Java CUP como parte del proceso de compilación. Este archivo contiene una lista de constantes que representan los diferentes símbolos terminales (tokens) definidos en la gramática del archivo sintax.cup. Cada constante es un identificador único asociado a un valor numérico, y su principal función es servir como referencia durante la ejecución del analizador sintáctico, facilitando la identificación de los tokens en el flujo de entrada.

sym.java actúa como una interfaz entre el analizador léxico y el sintáctico. Cuando el analizador léxico identifica un token en el código fuente, utiliza las constantes definidas en este archivo para comunicar al analizador sintáctico qué tipo de token ha encontrado. Esto asegura consistencia y evita errores de interpretación al estandarizar los valores asociados a cada símbolo.



El archivo sym.java es fundamental para mantener el flujo de datos entre las diferentes etapas del compilador de manera clara y eficiente. Sin este archivo, sería más complejo mantener la coherencia entre los nombres de los símbolos definidos en la gramática y los valores que utiliza el programa durante la ejecución. Además, facilita la depuración y el mantenimiento del código, ya que las constantes tienen nombres descriptivos que mejoran la legibilidad del programa.

## Código Generado:

```
1 //-----
2 // The following code was generated by CUP v0.11a beta 20060608
3 // Tue Dec 31 15:24:08 CST 2024
4 //-----
5 //-----
6
7 package codigo;
8
9 /**
10  * CUP generated class containing symbol constants. */
11 public class sym {
12     /* terminals */
13     public static final int operadorMultiplicacion = 29;
14     public static final int operadorResta = 28;
15     public static final int mayorQue = 36;
16     public static final int cicloWhile = 13;
17     public static final int finalTexto = 53;
18     public static final int operadorRaiz = 33;
19     public static final int valorBoleano = 7;
20     public static final int imprimir = 24;
21     public static final int operadorsuma = 27;
22     public static final int desPuntos = 55;
23     public static final int parentDer = 46;
24     public static final int detener = 19;
25     public static final int corcheteIzq = 49;
26     public static final int valorFletante = 6;
27     public static final int funcionLeer = 25;
28     public static final int puntoDecimal = 54;
29     public static final int inicioTexto = 52;
30     public static final int comillaDoble = 3;
31     public static final int caracter = 9;
32     public static final int condicionalElse = 11;
33     public static final int condicionalIf = 10;
34     public static final int menorIgual = 39;
```



```
40    public static final int menorQue = 37;
41    public static final int funcioneswitch = 15;
42    public static final int operadorNo = 42;
43    public static final int clase = 23;
44    public static final int operadorPotencia = 32;
45    public static final int cicloDo = 14;
46    public static final int llaveIzq = 47;
47    public static final int EOF = 0;
48    public static final int mayorIgual = 38;
49    public static final int error = 1;
50    public static final int funcionCase = 16;
51    public static final int linea = 2;
52    public static final int finlinea = 51;
53    public static final int cicloFor = 12;
54    public static final int funcion = 22;
55    public static final int ERROR = 58;
56    public static final int corcheteDer = 50;
57    public static final int operadorY = 40;
58    public static final int incremento = 43;
59    public static final int valorEntero = 5;
60    public static final int operadorDivision = 30;
61    public static final int comparacionIgualdad = 35;
62    public static final int variable = 21;
63    public static final int operadorO = 41;
64    public static final int parentizq = 45;
65    public static final int identificador = 56;
66    public static final int decremento = 44;
67    public static final int operadorModulo = 31;
68    public static final int comillaSimple = 4;
69    public static final int constante = 20;
70    public static final int llaveDer = 48;
71    public static final int Numero = 57;
72    public static final int predeterminado = 18;

    public static final int asignacion = 34;
    public static final int funcionesalircase = 17;
    public static final int cadena = 8;
    public static final int funcionIntroducirDatos = 26;
}
```

## Archivo Principal.java

El archivo **Principal.java** es el núcleo de ejecución del proyecto, encargado de automatizar la generación de los analizadores léxico y sintáctico, además de gestionar la interfaz gráfica inicial. Utiliza herramientas como **JFlex** y **Java CUP** para generar los componentes fundamentales del compilador a partir de archivos .flex y .cup, asegurando que estos se ubiquen correctamente en la estructura del proyecto. Además, el manejo de rutas relativas permite que el programa sea fácilmente adaptable a distintos entornos de desarrollo. La función `generar` se encarga de ejecutar estos procesos, mover los archivos generados a sus ubicaciones correspondientes y garantizar que no haya conflictos con versiones anteriores.



## Código Realizado:

```
1 package codigo;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.nio.file.Files;
6 import java.nio.file.Path;
7 import java.nio.file.Paths;
8 import java.util.logging.Level;
9 import java.util.logging.Logger;
10
11 public class Principal {
12
13     public static void main(String[] args) throws Exception {
14         // Usar rutas relativas basadas en la estructura del proyecto
15         String baseDir = "src/codigo"; // Asumiendo que los archivos están dentro de esta carpeta
16         String ruta1 = Paths.get(baseDir, "Lexer.flex").toString();
17         String ruta2 = Paths.get(baseDir, "LexerCup.flex").toString();
18         String[] rutas = {"-parser", "Sintax", Paths.get(baseDir, "Sintax.cup").toString()};
19
20         generar(ruta1, ruta2, rutas);
21         // Implementación del Runnable para el Splash y la pantalla principal
22         Runnable mRun = () -> {
23             // JFrame que funge como splash
24             ModernSplash mSplash = new ModernSplash();
25             mSplash.setVisible(true);
26
27             try {
28                 Thread.sleep(10000); // 5000 milisegundos equivale a 5 segundos.
29             } catch (InterruptedException ex) {
30                 Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
31             }
32
33             mSplash.dispose();
34
35             // JFrame que funge como pantalla principal
36             LoginUI mFrmPrincipal = new LoginUI(); // Renombrado de FrmBienvenida
37             mFrmPrincipal.setVisible(true);
38         };
39
40         // Creación e inicio del hilo para el Splash
41         Thread mHiloSplash = new Thread(mRun);
42         mHiloSplash.start();
43     }
44
45     public static void generar(String ruta1, String ruta2, String[] rutas) throws IOException, Exception {
46         File archivo;
47
48         // Usar JFlex para generar los archivos
49         archivo = new File(ruta1);
50         JFlex.Main.generate(archivo);
51
52         archivo = new File(ruta2);
53         JFlex.Main.generate(archivo);
54
55         // Ejecutar Java CUP para generar el parser
56         java_cup.Main.main(rutas);
57
58         // Rutas relativas para los archivos generados
59         Path rutaSym = Paths.get("src/codigo/sym.java");
60         if (Files.exists(rutaSym)) {
61             Files.delete(rutaSym);
62         }
63
64         // Mover el archivo sym.java generado
65         Files.move(
66             Paths.get("sym.java"),
```



```
67     rutaSym
68   );
69
70   // Rutas relativas para el archivo Sintax.java
71   Path rutaSin = Paths.get("src/codigo/Sintax.java");
72   if (Files.exists(rutaSin)) {
73     Files.delete(rutaSin);
74   }
75
76   // Mover el archivo Sintax.java generado
77   Files.move(
78     Paths.get("Sintax.java"),
79     rutaSin
80   );
81 }
82 }
```

## Archivo ModernSplash.java (Splash del proyecto)

El splash de este proyecto, llamado ModernSplash, es una pantalla inicial diseñada para ofrecer una experiencia visual atractiva mientras se cargan los recursos necesarios del sistema prácticamente es una ventana inicial moderna e interactiva que combina varias animaciones y efectos visuales para ofrecer una experiencia de bienvenida atractiva al usuario.. Este componente destaca por su diseño moderno, que incluye un fondo degradado en tonos azules, un título animado que realiza un efecto de caída y rebote simulado con física, y un ícono que flota suavemente gracias a una animación sinusoidal. Además, incorpora una barra de progreso estilizada con colores dinámicos y mensajes de estado que informan al usuario sobre el avance de la carga, como "Preparando recursos..." y "Cargando módulos...".

La finalidad principal de este splash es mejorar la interacción inicial con el usuario, proporcionando un entorno visual que no solo entretiene, sino que también comunica claramente el progreso mientras se inicializan los módulos del proyecto. Este tipo de elemento no solo ayuda a gestionar la percepción del tiempo de carga, sino que también refuerza la identidad visual del proyecto, haciendo que la primera impresión sea profesional y agradable.



## Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class ModernSplash extends JFrame {
8     private JProgressBar progressBar;
9     private JLabel titleLabel, loadingLabel, iconLabel;
10    private Timer progressTimer, fadeTimer, bounceTimer, colorTimer, floatTimer;
11    private float opacity = 0.0f;
12    private Color[] gradientColors = {new Color(76, 175, 80)};
13    private int gradientIndex = 0;
14
15    // Variables para animación del título
16    private int titleYPosition = 50; // Posición inicial
17    private int targetY = 170; // Posición final
18    private double velocity = 0; // Velocidad inicial
19    private double gravity = 1.2; // Simula la gravedad
20    private int bouncesRemaining = 3; // Número de rebotes
21    private double dampingFactor = 0.6; // Factor de amortiguación
22
23    // Variables para animación flotante del ícono
24    private int iconBaseY = 30; // Posición base del ícono
25    private int floatAmplitude = 10; // Amplitud de la flotación
26    private double floatAngle = 0; // Ángulo para la animación sinusoidal
27
28    public ModernSplash() {
29        initComponents();
30    }
31
32    private void initComponents() {
33        // Configuración principal del JFrame
34        setUndecorated(true);
35        setSize(400, 300);
36        setOpacity(0.0f);
37        setLocationRelativeTo(null);
38        setLayout(null);
39
40        // Fondo personalizado
41        JPanel backgroundPanel = new JPanel() {
42            @Override
43            protected void paintComponent(Graphics g) {
44                super.paintComponent(g);
45                Graphics2D g2d = (Graphics2D) g;
46                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
47
48                // Degradado del fondo
49                GradientPaint gradient = new GradientPaint(0, 0, new Color(30, 136, 229), getWidth(), getHeight(), new Color(63, 81, 181));
50                g2d.setPaint(gradient);
51                g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
52            }
53        };
54        backgroundPanel.setBounds(0, 0, 400, 300);
55        backgroundPanel.setLayout(null);
56        add(backgroundPanel);
57
58        // Ícono
59        iconLabel = new JLabel(new ImageIcon("C:\\Users\\jesus\\Downloads\\AnalizadorLexico\\src\\codigo\\icono.png"));
60        iconLabel.setBounds(125, iconBaseY, 150, 150);
61        iconLabel.setHorizontalTextPosition(SwingConstants.CENTER);
62        backgroundPanel.add(iconLabel);
63    }
```



```
64     // Título con animación
65     titleLabel = new JLabel("Token-Visor");
66     titleLabel.setBounds(50, titleYPosition, 300, 40);
67     titleLabel.setFont(new Font("Roboto", Font.BOLD, 24));
68     titleLabel.setForeground(Color.WHITE);
69     titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
70     backgroundPanel.add(titleLabel);
71
72     // Barra de progreso
73     progressBar = new JProgressBar() {
74         @Override
75         protected void paintComponent(Graphics g) {
76             super.paintComponent(g);
77             Graphics2D g2d = (Graphics2D) g;
78             g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
79
80             int progressWidth = (int) (getWidth() * (getValue() / 100.0));
81             g2d.setColor(gradientColors[gradientIndex]); // Color dinámico
82             g2d.fillRoundRect(0, 0, progressWidth, getHeight(), 10, 10);
83         }
84     };
85     progressBar.setBounds(50, 230, 300, 10);
86     progressBar.setValue(0);
87     backgroundPanel.add(progressBar);
88
89     // Texto de carga
90     loadingLabel = new JLabel("Iniciando...");
91     loadingLabel.setBounds(50, 250, 300, 20);
92     loadingLabel.setFont(new Font("Roboto", Font.PLAIN, 14));
93     loadingLabel.setForeground(Color.WHITE);
94     loadingLabel.setHorizontalAlignment(SwingConstants.CENTER);
95     backgroundPanel.add(loadingLabel);
96
97     // Iniciar animaciones
98     startAnimations();
99 }
100
101 private void startAnimations() {
102     // Fade-in de la ventana
103     fadeTimer = new Timer(30, (ActionEvent e) -> {
104         opacity += 0.02f;
105         if (opacity >= 1.0f) {
106             setOpacity(1.0f);
107             fadeTimer.stop();
108             startFloatAnimation(); // Inicia la animación flotante del ícono
109             startBounceAnimation(); // Inicia la animación del título
110         } else {
111             setOpacity(opacity);
112         }
113     });
114     fadeTimer.start();
115
116     // Cambio de color dinámico para la barra de progreso
117     colorTimer = new Timer(2000, (ActionEvent e) -> {
118         gradientIndex = (gradientIndex + 1) % gradientColors.length;
119         progressBar.repaint();
120     });
121     colorTimer.start();
122 }
123
124 private void startFloatAnimation() {
125     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez
126         // Animación sinusoidal para "flotar"
127         floatAngle += 0.05f; // Incremento más pequeño para ralentizar el movimiento
128         int offsety = (int) (Math.sin(floatAngle) * floatAmplitude);
129         iconLabel.setBounds(125, iconBaseY + offsety, 150, 150);
130     });
131 }
```



```
130     });
131     floatTimer.start();
132 }
133
134
135     private void startBounceAnimation() {
136         bounceTimer = new Timer(20, (ActionEvent e) -> {
137             velocity += gravity; // Incrementa velocidad por efecto de "gravedad"
138             titleYPosition += velocity; // Actualiza posición del título
139
140             // Detecta colisión con el suelo virtual
141             if (titleYPosition >= targetY) {
142                 titleYPosition = targetY; // Ajusta la posición al suelo
143                 velocity = -velocity * dampingFactor; // Rebote con amortiguación
144                 bouncesRemaining--;
145
146                 // Si no quedan más rebotes, detiene la animación
147                 if (bouncesRemaining == 0) {
148                     bounceTimer.stop();
149                     titleYPosition = targetY; // Asegura posición final
150                     startProgressBar(); // Inicia la barra de progreso
151                 }
152             }
153
154             // Actualiza la posición del título en el componente
155             titleLabel.setBounds(50, titleYPosition, 300, 40);
156         });
157
158         bounceTimer.start();
159     }
160
161     private void startProgressBar() {
162         progressTimer = new Timer(50, (ActionEvent e) -> {
163             int value = progressBar.getValue();
164             if (value < 100) {
165                 progressBar.setValue(value + 1);
166
167                 // Cambiar texto dinámicamente
168                 if (value < 30) {
169                     loadingLabel.setText("Preparando recursos...");
170                 } else if (value < 60) {
171                     loadingLabel.setText("Cargando módulos...");
172                 } else if (value < 90) {
173                     loadingLabel.setText("Finalizando configuración...");
174                 } else {
175                     loadingLabel.setText("¡Listo para comenzar!");
176                 }
177             } else {
178                 progressTimer.stop();
179                 startFadeout();
180             }
181         });
182         progressTimer.start();
183     }
184
185     private void startFadeout() {
186         fadeTimer = new Timer(30, (ActionEvent e) -> {
187             opacity -= 0.02f;
188             if (opacity <= 0.0f) {
189                 setOpacity(0.0f);
190                 fadeTimer.stop();
191                 dispose(); // Cierra la ventana
192             } else {
193                 setOpacity(opacity);
194             }
195         });
196         fadeTimer.start();
197     }
198
199     public static void main(String[] args) {
200         EventQueue.invokeLater(() -> {
201             ModernSplash splash = new ModernSplash();
202             splash.setVisible(true);
203         });
204     }
205 }
```



## Archivo LoginUI.java

El archivo LoginUI.java es una implementación de una interfaz gráfica de usuario (GUI) en Java diseñada para ofrecer una experiencia interactiva de inicio de sesión dentro del analizador léxico. Utilizando la biblioteca Swing, esta clase presenta una ventana dividida en dos paneles: un lado izquierdo visualmente atractivo con un ícono flotante animado y un mensaje de bienvenida, y un lado derecho funcional que contiene el formulario de inicio de sesión con campos para el usuario y la contraseña.

La importancia de este archivo radica en su combinación de funcionalidad y estética. Además de validar credenciales de acceso con una lógica sencilla, proporciona un diseño profesional que mejora la interacción del usuario mediante detalles como animaciones suaves y botones personalizados. Su estructura modular y enfoque en la usabilidad lo convierten en un componente clave del sistema, creando una primera impresión positiva y facilitando la transición hacia las funcionalidades principales del programa.

### Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class LoginUI extends JFrame {
8
9     private JTextField registerNameField;
10    private JPasswordField registerPasswordField;
11    private JPanel leftPanel;
12    private JLabel iconLabel;
13    private Timer floatTimer;
14    private int iconRageY = 100; // Posición base del ícono en el eje Y
15    private int floatAmplitude = 10; // Amplitud de la flotación
16    private double floatAngle = 0; // Ángulo para la animación sinusoidal
17
18    public LoginUI() {
19        initComponents();
20        startFloatAnimation(); // Inicia la animación flotante del ícono
21    }
22
23    private void initComponents() {
24        setTitle("Login UI");
25        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26        setSize(800, 500);
27        setLocationRelativeTo(null);
28       .setLayout(new GridLayout(1, 2));
29
30        // Panel izquierdo (Bienvenida)
31        leftPanel = new JPanel();
32        leftPanel.setBackground(new Color(0, 38, 95));
33        leftPanel.setLayout(null); // Posición absoluta para incluir el ícono
```



# MANUAL DE PRÁCTICAS



```
35     JLabel welcomeLabel = new JLabel("<html><center>Nos Alegra Verte De Nuevo!<br>Inicia sesión para continuar</center></html>");
36     welcomeLabel.setForeground(Color.WHITE);
37     welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));
38     welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
39     welcomeLabel.setBounds(50, 300, 300, 100);
40     leftPanel.add(welcomeLabel);
41
42     // ícono flotante
43     iconLabel = new JLabel(new ImageIcon("C:\\\\Users\\\\jesus\\\\Downloads\\\\AnalizadorLexico\\\\src\\\\codigo\\\\icono.png"));
44     iconLabel.setBounds(0, iconBaseY, 150, 150); // La posición X se ajustará dinámicamente
45     leftPanel.add(iconLabel);
46
47     add(leftPanel);
48
49     // Panel derecho (Formulario de registro)
50     JPanel rightPanel = new JPanel();
51     rightPanel.setBackground(Color.WHITE);
52     rightPanel.setLayout(new GridBagLayout());
53
54     GridBagConstraints gbc = new GridBagConstraints();
55     gbc.insets = new Insets(10, 10, 10, 10);
56     gbc.gridx = 0;
57     gbc.anchor = GridBagConstraints.CENTER;
58
59     JLabel createAccountLabel = new JLabel("Inicio De Sesión");
60     createAccountLabel.setFont(new Font("Arial", Font.BOLD, 24));
61     createAccountLabel.setForeground(Color.BLACK);
62     gbc.gridy = 0;
63     rightPanel.add(createAccountLabel, gbc);
64
65     registerNameField = new JTextField(20);
66     registerNameField.setFont(new Font("Arial", Font.PLAIN, 14));
67     registerNameField.setBorder(BorderFactory.createTitledBorder("Usuario"));
68
68     gbc.gridy = 1;
69     rightPanel.add(registerNameField, gbc);
70
71     registerPasswordField = new JPasswordField(20);
72     registerPasswordField.setFont(new Font("Arial", Font.PLAIN, 14));
73     registerPasswordField.setBorder(BorderFactory.createTitledBorder("Contraseña"));
74     gbc.gridy = 2;
75     rightPanel.add(registerPasswordField, gbc);
76
77     CustomButton signUpButton = new CustomButton("Ingresar");
78     signUpButton.setPreferredSize(new Dimension(200, 50));
79     signUpButton.addActionListener(e -> {
80         String name = registerNameField.getText();
81         String password = new String(registerPasswordField.getPassword());
82
83         // Validación simple
84         if (name.equals("jesus") && password.equals("jesus123")) {
85             JOptionPane.showMessageDialog(this, "Inicio de sesión exitoso");
86             this.dispose(); // Cierra la ventana actual
87             FrmPrincipal mainFrame = new FrmPrincipal();
88             mainFrame.setVisible(true); // Abre la ventana principal
89         } else {
90             JOptionPane.showMessageDialog(this, "Credenciales inválidas. Intenta de nuevo.", "Error", JOptionPane.ERROR_MESSAGE);
91         }
92     });
93     gbc.gridy = 3;
94     rightPanel.add(signUpButton, gbc);
95
96     add(rightPanel);
97 }
98
99 private void startFloatAnimation() {
100     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez
```



```
101     // Animación sinusoidal para "flotar"
102     floatAngle += 0.05; // Incremento más pequeño para ralentizar el movimiento
103     int offsetY = (int) (Math.sin(floatAngle) * floatAmplitude);
104
105     // Recalcula la posición horizontal centrada
106     int iconX = (leftPanel.getWidth() - iconLabel.getWidth()) / 2;
107
108     // Aplica la posición centrada y el desplazamiento vertical
109     iconLabel.setBounds(iconX, iconBaseY + offsetY, 150, 150);
110   });
111   floatTimer.start();
112 }
113
114 public static void main(String[] args) {
115   SwingUtilities.invokeLater(() -> {
116     new LoginUI().setVisible(true);
117   });
118 }
119
120
121 // Clase personalizada para el botón
122 class CustomButton extends JButton {
123   private boolean isHovered = false;
124
125   public CustomButton(String text) {
126     super(text);
127     setContentAreaFilled(false);
128     setFocusPainted(false);
129     setBorderPainted(false);
130    setFont(new Font("Arial", Font.BOLD, 16));
131     setForeground(Color.WHITE);
132     setCursor(new Cursor(Cursor.HAND_CURSOR));
133
134   addMouseListener(new java.awt.event.MouseAdapter() {
135     @Override
136     public void mouseEntered(java.awt.event.MouseEvent e) {
137       isHovered = true;
138       repaint();
139     }
140
141     @Override
142     public void mouseExited(java.awt.event.MouseEvent e) {
143       isHovered = false;
144       repaint();
145     }
146   });
147
148   @Override
149   protected void paintComponent(Graphics g) {
150     Graphics2D g2d = (Graphics2D) g.create();
151     g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
152
153     Color topColor = isHovered ? new Color(0, 120, 215) : new Color(33, 97, 140);
154     Color bottomColor = isHovered ? new Color(0, 85, 170) : new Color(0, 51, 102);
155
156     GradientPaint gradient = new GradientPaint(0, 0, topColor, 0, getHeight(), bottomColor);
157     g2d.setPaint(gradient);
158     g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
159
160     g2d.setColor(new Color(0, 0, 50));
161     g2d.fillRoundRect(2, 2, getWidth() - 4, getHeight() - 4, 30, 30);
162
163     super.paintComponent(g2d);
164     g2d.dispose();
165   }
166 }
167
168   @Override
169   protected void paintBorder(Graphics g) {
170 }
171 }
```

## Archivo FrmCreditos.java

El archivo FrmCreditos.java es una ventana gráfica diseñada para mostrar los créditos del software. Está construido utilizando la biblioteca Swing de Java, lo que permite una interfaz visual sencilla y efectiva. La ventana destaca los nombres de los desarrolladores involucrados en el proyecto, presentados en un diseño limpio y bien organizado dentro de paneles anidados. Además, cuenta con un botón "Cerrar" que permite al usuario cerrar la ventana de manera eficiente, manteniendo una experiencia de usuario intuitiva. Su diseño incluye elementos estilizados como colores de fondo personalizados, fuentes destacadas y etiquetas para una presentación profesional.

### Código Realizado:

```
1  /*  
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
4   */  
5  package codigo;  
6  
7  /**  
8   *  
9   * @author THINKPAD  
10  */  
11 public class FrmCreditos extends javax.swing.JFrame {  
12  
13  /**  
14   * Creates new form FrmCreditos  
15  */  
16  public FrmCreditos() {  
17      initComponents();  
18      this.setLocationRelativeTo(null);  
19  }  
20  
21  /**  
22   * This method is called from within the constructor to initialize the form.  
23   * WARNING: Do NOT modify this code. The content of this method is always  
24   * regenerated by the Form Editor.  
25   */  
26  @SuppressWarnings("unchecked")  
27  // <editor-fold defaultstate="collapsed" desc="Generated Code">  
28  private void initComponents() {  
29  
30      jLabel4 = new javax.swing.JLabel();  
31      jPanell = new javax.swing.JPanel();  
32      jPanel2 = new javax.swing.JPanel();  
33      btnLimpiaresIn1 = new javax.swing.JButton();
```



# **MANUAL DE PRACTICAS**





```
100     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
101     .addComponent(jLabel6)
102     .addContainerGap())
103   );
104   jPanel3Layout.setVerticalGroup(
105     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
106     .addGroup(jPanel3Layout.createSequentialGroup()
107       .addGap(12, 12, 12)
108       .addComponent(jLabel6)
109       .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
110       .addComponent(jLabel3)
111       .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
112       .addComponent(jLabel11)
113       .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
114       .addComponent(jLabel15)
115       .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
116       .addComponent(jLabel12)
117       .addContainerGap(23, Short.MAX_VALUE))
118   );
119
120   javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel12);
121   jPanel12.setLayout(jPanel2Layout);
122   jPanel2Layout.setHorizontalGroup(
123     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
124     .addGroup(jPanel2Layout.createSequentialGroup()
125       .addGap(20, 20, 20)
126       .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
127       .addContainerGap(22, Short.MAX_VALUE))
128     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
129       .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
130       .addComponent(btnLimpiarSin1)
131       .addGap(160, 160, 160))
132   );
133   jPanel2Layout.setVerticalGroup(
134     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
135     .addGroup(jPanel2Layout.createSequentialGroup()
136       .addGap(35, 35, 35)
137       .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
138       .addGap(18, 18, 18)
139       .addComponent(btnLimpiarSin1)
140       .addContainerGap(10, Short.MAX_VALUE))
141   );
142
143   javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
144   jPanel1.setLayout(jPanel1Layout);
145   jPanel1Layout.setHorizontalGroup(
146     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
147       .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
148   );
149   jPanel1Layout.setVerticalGroup(
150     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
151       .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE)
152   );
153
154   jPanel2.getAccessibleContext().setAccessibleName("Créditos");
155
156   javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
157   getContentPane().setLayout(layout);
158   layout.setHorizontalGroup(
159     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
160       .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
161   );
162   layout.setVerticalGroup(
163     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
164       .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
165   );
```



```
166     pack();
167 } // </editor-fold>
168
169 private void btnLimpiarSinActionPerformed(java.awt.event.ActionEvent evt) {
170     // TODO add your handling code here:
171     |    this.dispose(); // Cierra la ventana actual
172 }
173
174 /**
175 * @param args the command line arguments
176 */
177 public static void main(String args[]) {
178     /* Set the Nimbus look and feel */
179     //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
180     /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
181     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html.
182     */
183     try {
184         for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
185             if ("Nimbus".equals(info.getName())) {
186                 javax.swing.UIManager.setLookAndFeel(info.getClassName());
187                 break;
188             }
189         }
190     } catch (ClassNotFoundException ex) {
191         java.util.logging.Logger.getLogger(FrmCreditos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
192     } catch (InstantiationException ex) {
193         java.util.logging.Logger.getLogger(FrmCreditos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
194     } catch (IllegalAccessException ex) {
195         java.util.logging.Logger.getLogger(FrmCreditos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
196     } catch (javax.swing.UnsupportedLookAndFeelException ex) {
197         java.util.logging.Logger.getLogger(FrmCreditos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
198     }
199 }
200 //</editor-fold>
201
202 /* Create and display the form */
203 java.awt.EventQueue.invokeLater(new Runnable() {
204     public void run() {
205         new FrmCreditos().setVisible(true);
206     }
207 });
208 }
209
210 // Variables declaration - do not modify
211 private javax.swing.JButton btnLimpiarSin;
212 private javax.swing.JLabel jLabel1;
213 private javax.swing.JLabel jLabel2;
214 private javax.swing.JLabel jLabel3;
215 private javax.swing.JLabel jLabel4;
216 private javax.swing.JLabel jLabel5;
217 private javax.swing.JLabel jLabel6;
218 private javax.swing.JPanel jPanel1;
219 private javax.swing.JPanel jPanel2;
220 private javax.swing.JPanel jPanel3;
221 // End of variables declaration
222 }
```

### Archivo FrmPrincipal.java

El archivo `FrmPrincipal.java` es el componente principal de la interfaz gráfica de usuario (GUI) del proyecto. En este archivo, se gestionan las funcionalidades clave relacionadas con el análisis léxico y sintáctico, además de la interacción con el usuario. Utiliza el componente `JTextArea` para mostrar y manipular texto de entrada y salida. La clase interna `LineNumberingTextArea` se encarga de agregar numeración a las líneas del área de texto, lo que facilita la visualización y el análisis de códigos fuente. También contiene un método `analizarLexico` que recorre el texto ingresado, aplicando un lexer para identificar y mostrar los tokens correspondientes con su valor y tipo, generando una salida organizada que se muestra en un `JTextArea`.



Además de las funciones relacionadas con el análisis de código, FrmPrincipal.java implementa botones y acciones para permitir al usuario cargar, limpiar o analizar el texto ingresado. Entre estos botones se encuentran opciones para cargar archivos de texto, limpiar los resultados y abrir ventanas de ayuda e información. Por ejemplo, al presionar un botón de "Ayuda", se muestra una ventana con una imagen explicativa, y al presionar "Creditos", se abre una ventana con los créditos del proyecto. También se incluye la opción de analizar el código sintácticamente mediante un parser utilizando Sintax y el lexer generado por Java CUP.

## Código Realizado:

```
6 package codigo;
7
8 import java.io.File;
9 import java.io.FileNotFoundException;
10 import java.io.IOException;
11 import java.io.StringReader;
12 import java.nio.file.Files;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import java_cup.runtime.Symbol;
16
17 import javax.swing.*;
18 import javax.swing.text.Element;
19 import java.awt.*;
20
21
22 public class FrmPrincipal extends javax.swing.JFrame {
23
24     /**
25      * Creates new form FrmPrincipal
26      */
27     public FrmPrincipal() {
28         initComponents();
29         setLocationRelativeTo(null);
30     }
31
32     public class LineNumberingTextArea extends JTextArea {
33         private JTextArea textArea;
34
35         public LineNumberingTextArea(JTextArea textArea) {
36             this.textArea = textArea;
37             this.textArea.getDocument().addDocumentListener(new javax.swing.event.DocumentListener() {
@Override
public void insertUpdate(javax.swing.event.DocumentEvent e) {
```



```
39     updateLineNumbers();
40 }
41
42     @Override
43     public void removeUpdate(javax.swing.event.DocumentEvent e) {
44         updateLineNumbers();
45     }
46
47     @Override
48     public void changedUpdate(javax.swing.event.DocumentEvent e) {
49         updateLineNumbers();
50     }
51 });
52 updateLineNumbers();
53 }
54
55     private void updateLineNumbers() {
56         StringBuilder lineNumbers = new StringBuilder();
57         Element root = textArea.getDocument().getDefaultRootElement();
58         int lineCount = root.getElementCount();
59         for (int i = 1; i <= lineCount; i++) {
60             lineNumbers.append(i).append(System.lineSeparator());
61         }
62         setText(lineNumbers.toString());
63     }
64
65     @Override
66     public void paintComponent(Graphics g) {
67         super.paintComponent(g);
68         setBackground(new Color(27, 40, 49)); // Fondo gris claro
69         setForeground(Color.WHITE); // Números en negro
70         setEditable(false);
71     }
72 }
73
74     private void analizarLexico() throws IOException {
75         int cont = 1;
76
77         String expr = (String) txtResultado.getText();
78         Lexer lexer = new Lexer(new StringReader(expr));
79         String resultado = "LINEA " + cont + "\t\tsIMBOLO\n";
80         while (true) {
81             Tokens token = lexer.yylex();
82             if (token == null) {
83                 txtAnalizarLex.setText(resultado);
84                 return;
85             }
86             switch (token) {
87                 case Linea:
88                     cont++;
89                     resultado += "LINEA " + cont + "\n";
90                     break;
91                 case comillaDoble:
92                     resultado += "comillaDoble\t\t" + lexer.lexeme + "\n";
93                     break;
94                 case comillaSimple:
95                     resultado += "comillaSimple\t\t" + lexer.lexeme + "\n";
96                     break;
97                 case valorEntero:
98                     resultado += "valorEntero\t\t" + lexer.lexeme + "\n";
99                     break;
100                case valorFlotante:
101                    resultado += "valorFlotante\t\t" + lexer.lexeme + "\n";
102                    break;
103                case valorBooleano:
104                    resultado += "valorBooleano\t\t" + lexer.lexeme + "\n";
```



```
105     break;
106 case cadena:
107     resultado += "cadena\t" + lexer.lexeme + "\n";
108     break;
109 case carácter:
110     resultado += "carácter\t" + lexer.lexeme + "\n";
111     break;
112 case condicionalIf:
113     resultado += "condicionalIf\t" + lexer.lexeme + "\n";
114     break;
115 case condicionalElse:
116     resultado += "condicionalElse\t" + lexer.lexeme + "\n";
117     break;
118 case cicloFor:
119     resultado += "cicloFor\t\t" + lexer.lexeme + "\n";
120     break;
121 case cicloWhile:
122     resultado += "cicloWhile\t" + lexer.lexeme + "\n";
123     break;
124 case cicloDo:
125     resultado += "cicloDo\t" + lexer.lexeme + "\n";
126     break;
127 case funcionSwitch:
128     resultado += "funcionSwitch\t" + lexer.lexeme + "\n";
129     break;
130 case funcionCase:
131     resultado += "funcionCase\t" + lexer.lexeme + "\n";
132     break;
133 case funcionSalirCase:
134     resultado += "funcionSalirCase\t" + lexer.lexeme + "\n";
135     break;
136 case predeterminado:
137     resultado += "predeterminado\t" + lexer.lexeme + "\n";
138
139     break;
140 case detener:
141     resultado += "detener\t\t" + lexer.lexeme + "\n";
142     break;
143 case constante:
144     resultado += "constante\t\t" + lexer.lexeme + "\n";
145     break;
146 case variable:
147     resultado += "variable\t\t" + lexer.lexeme + "\n";
148     break;
149 case funcion:
150     resultado += "funcion\t\t" + lexer.lexeme + "\n";
151     break;
152 case clase:
153     resultado += "clase\t\t" + lexer.lexeme + "\n";
154     break;
155 case imprimir:
156     resultado += "imprimir\t\t" + lexer.lexeme + "\n";
157     break;
158 case funcionLeer:
159     resultado += "funcionLeer\t" + lexer.lexeme + "\n";
160     break;
161 case funcionIntroducirDatos:
162     resultado += "funcionIntroducirDatos\t" + lexer.lexeme + "\n";
163     break;
164 case operadorSuma:
165     resultado += "operadorSuma\t" + lexer.lexeme + "\n";
166     break;
167 case operadorResta:
168     resultado += "operadorResta\t" + lexer.lexeme + "\n";
169     break;
170 case operadorMultiplicacion:
171     resultado += "operadorMultiplicacion\t" + lexer.lexeme + "\n";
```



```
171     break;
172 case operadorDivision:
173     resultado += "operadorDivision\t" + lexer.lexeme + "\n";
174     break;
175 case operadorModulo:
176     resultado += "operadorModulo\t" + lexer.lexeme + "\n";
177     break;
178 case operadorPotencia:
179     resultado += "operadorPotencia\t" + lexer.lexeme + "\n";
180     break;
181 case operadorRaiz:
182     resultado += "operadorRaiz\t\t" + lexer.lexeme + "\n";
183     break;
184 case asignacion:
185     resultado += "asignacion\t\t" + lexer.lexeme + "\n";
186     break;
187 case comparacionIgualdad:
188     resultado += "comparacionIgualdad\t" + lexer.lexeme + "\n";
189     break;
190 case mayorQue:
191     resultado += "mayorQue\t\t" + lexer.lexeme + "\n";
192     break;
193 case menorQue:
194     resultado += "menorQue\t\t" + lexer.lexeme + "\n";
195     break;
196 case mayorIgual:
197     resultado += "mayorIgual\t\t" + lexer.lexeme + "\n";
198     break;
199 case menorIgual:
200     resultado += "menorIgual\t\t" + lexer.lexeme + "\n";
201     break;
202 case operadory:
203     resultado += "operadory\t\t" + lexer.lexeme + "\n";
204     break;
205 case operadorno:
206     resultado += "operadorno\t\t" + lexer.lexeme + "\n";
207     break;
208 case operadornoNo:
209     resultado += "operadornoNo\t\t" + lexer.lexeme + "\n";
210     break;
211 case incremento:
212     resultado += "incremento\t\t" + lexer.lexeme + "\n";
213     break;
214 case decremento:
215     resultado += "decremento\t\t" + lexer.lexeme + "\n";
216     break;
217 case parentIzq:
218     resultado += "parentIzq\t\t" + lexer.lexeme + "\n";
219     break;
220 case parentDer:
221     resultado += "parentDer\t\t" + lexer.lexeme + "\n";
222     break;
223 case llaveIzq:
224     resultado += "llaveIzq\t\t" + lexer.lexeme + "\n";
225     break;
226 case llaveDer:
227     resultado += "llaveDer\t\t" + lexer.lexeme + "\n";
228     break;
229 case corchetIzq:
230     resultado += "corchetIzq\t\t" + lexer.lexeme + "\n";
231     break;
232 case corchetDer:
233     resultado += "corchetDer\t\t" + lexer.lexeme + "\n";
234     break;
235 case finLinea:
236     resultado += "finLinea\t\t" + lexer.lexeme + "\n";
```



```
237         break;
238     case inicioTexto:
239         resultado += "inicioTexto\t\t" + lexer.lexeme + "\n";
240         break;
241     case finalTexto:
242         resultado += "finalTexto\t\t" + lexer.lexeme + "\n";
243         break;
244
245     case puntoDecimal:
246         resultado += "puntoDecimal\t\t" + lexer.lexeme + "\n";
247         break;
248     case dosPuntos:
249         resultado += "dosPuntos\t\t" + lexer.lexeme + "\n";
250         break;
251     case Identificador:
252         resultado += "Identificador\t\t" + lexer.lexeme + "\n";
253         break;
254     case Número:
255         resultado += "Número\t\t" + lexer.lexeme + "\n";
256         break;
257     case ERROR:
258         resultado += "ERROR\t" + lexer.lexeme + "\n";
259         break;
260     default:
261         resultado += "<" + lexer.lexeme + ">\n";
262         break;
263     }
264
265   }
266
267
268 /**
269 */

270 /**
271 * This method is called from within the constructor to initialize the form.
272 * WARNING: Do NOT modify this code. The content of this method is always
273 * regenerated by the Form Editor.
274 */
275 @SuppressWarnings("unchecked")
276 // <editor-fold defaultstate="collapsed" desc="Generated Code">
277 private void initComponents() {
278
279     jPanel1 = new javax.swing.JPanel();
280     btnArchivo = new javax.swing.JButton();
281     jScrollPane1 = new javax.swing.JScrollPane();
282     txtResultado = new javax.swing.JTextArea();
283     jScrollPane2 = new javax.swing.JScrollPane();
284     txtAnalizarLex = new javax.swing.JTextArea();
285     btnAnalizarLex = new javax.swing.JButton();
286     btnLimpiarLex = new javax.swing.JButton();
287     jPanel2 = new javax.swing.JPanel();
288     jScrollPane3 = new javax.swing.JScrollPane();
289     txtAnalizarsin = new javax.swing.JTextArea();
290     btnAnalizarSin = new javax.swing.JButton();
291     btnLimpiarSin = new javax.swing.JButton();
292     btnLimpiarLex1 = new javax.swing.JButton();
293     btnLimpiarLex2 = new javax.swing.JButton();
294     btnLimpiarLex3 = new javax.swing.JButton();
295     btnLimpiarSin1 = new javax.swing.JButton();
296
297     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
298     setBackground(new java.awt.Color(51, 51, 51));
299
300     jPanel1.setBackground(new java.awt.Color(13, 23, 32));
301     jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Analizador Lexico", javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.CENTER));
302     jPanel1.setForeground(new java.awt.Color(255, 255, 255));
```



# MANUAL DE PRÁCTICAS



```
303     btnArchivo.setBackground(new java.awt.Color(2, 35, 84));
304     btnArchivo.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
305     btnArchivo.setForeground(new java.awt.Color(255, 255, 255));
306     btnArchivo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/open-file_40455.png"))); // NOI18N
307     btnArchivo.setText("Abrir Archivo");
308     btnArchivo.setToolTipText("");
309     btnArchivo.addActionListener(new java.awt.event.ActionListener() {
310         public void actionPerformed(java.awt.event.ActionEvent evt) {
311             btnArchivoActionPerformed(evt);
312         }
313     });
314
315     txtResultado.setBackground(new java.awt.Color(27, 40, 49));
316     txtResultado.setColumns(20);
317     txtResultado.setForeground(new java.awt.Color(18, 251, 197));
318     txtResultado.setRows(5);
319     LineNumberingTextArea lineNumberingResultado = new LineNumberingTextArea(txtResultado);
320     JScrollPane scrollPaneResultado = new JScrollPane(lineNumberingResultado);
321     scrollPaneResultado.setRowHeaderView(lineNumberingResultado);
322     jScrollPane1 = scrollPaneResultado;
323
324
325     txtAnalizarLex.setEditable(false);
326     txtAnalizarLex.setBackground(new java.awt.Color(27, 40, 49));
327     txtAnalizarLex.setColumns(20);
328     txtAnalizarLex.setForeground(new java.awt.Color(18, 251, 197));
329     txtAnalizarLex.setRows(5);
330     jScrollPane2.setViewportView(txtAnalizarLex);
331
332     btnAnalizarLex.setBackground(new java.awt.Color(2, 35, 84));
333     btnAnalizarLex.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
334     btnAnalizarLex.setForeground(new java.awt.Color(255, 255, 255));
335     btnAnalizarLex.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/analysis256_24853.png"))); // NOI18N
336
337     btnAnalizarLex.setText("Analizar");
338     btnAnalizarLex.addActionListener(new java.awt.event.ActionListener() {
339         public void actionPerformed(java.awt.event.ActionEvent evt) {
340             btnAnalizarLexActionPerformed(evt);
341         }
342     });
343
344     btnLimpiarLex.setBackground(new java.awt.Color(2, 35, 84));
345     btnLimpiarLex.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
346     btnLimpiarLex.setForeground(new java.awt.Color(255, 255, 255));
347     btnLimpiarLex.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/23-Full_Trash_256x256_35388.png"))); // NOI18N
348     btnLimpiarLex.setText("Limpiar");
349     btnLimpiarLex.addActionListener(new java.awt.event.ActionListener() {
350         public void actionPerformed(java.awt.event.ActionEvent evt) {
351             btnLimpiarLexActionPerformed(evt);
352         }
353     });
354
355     jPanel12.setBackground(new java.awt.Color(13, 23, 32));
356     jPanel12.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Analizador Sintáctico", javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.DEFAULT));
357
358     txtAnalizarSin.setEditable(false);
359     txtAnalizarSin.setBackground(new java.awt.Color(27, 40, 49));
360     txtAnalizarSin.setColumns(20);
361     txtAnalizarSin.setRows(5);
362     jScrollPane3.setViewportView(txtAnalizarSin);
363
364     btnAnalizarSin.setBackground(new java.awt.Color(2, 35, 84));
365     btnAnalizarSin.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
366     btnAnalizarSin.setForeground(new java.awt.Color(255, 255, 255));
367     btnAnalizarSin.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/analysis256_24853.png"))); // NOI18N
368     btnAnalizarSin.setText("Analizar");
369     btnAnalizarSin.addActionListener(new java.awt.event.ActionListener() {
```



# MANUAL DE PRÁCTICAS



```
① 370     public void actionPerformed(java.awt.event.ActionEvent evt) {
371         btnAnalizarSinActionPerformed(evt);
372     });
373
374     btnLimpiarSin.setBackground(new java.awt.Color(2, 35, 84));
375     btnLimpiarSin.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
376     btnLimpiarSin.setForeground(new java.awt.Color(255, 255, 255));
377     btnLimpiarSin.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/23-Full_Trash_256x256_35388.png"))); // NOI18N
378     btnLimpiarSin.setText("Limpiar");
379     btnLimpiarSin.addActionListener(new java.awt.event.ActionListener() {
380         public void actionPerformed(java.awt.event.ActionEvent evt) {
381             btnLimpiarSinActionPerformed(evt);
382         }
383     });
384
385     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
386     jPanel2.setLayout(jPanel2Layout);
387     jPanel2Layout.setHorizontalGroup(
388         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
389             .addGroup(jPanel2Layout.createSequentialGroup()
390                 .addGap(21, 21, 21)
391                 .addComponent(jScrollPane3))
392             .addGroup(jPanel2Layout.createSequentialGroup()
393                 .addGapGap(21, 21, 21)
394                 .addComponent(btnAnalizarSin)
395                 .addGapPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
396                 .addComponent(btnLimpiarSin)
397                 .addGapGap(18, 18, 18))
398         );
399     jPanel2Layout.setVerticalGroup(
400         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
401             .addGroup(jPanel2Layout.createSequentialGroup()
402                 .addGapContainerGap()
403                 .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 117, javax.swing.GroupLayout.PREFERRED_SIZE)
404                 .addGapPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
405                 .addGroup(jPanel2Layout.createSequentialGroup()
406                     .addGroup(jPanel2Layout.createSequentialGroup()
407                         .addComponent(btnLimpiarSin)
408                         .addGapGap(0, 0, Short.MAX_VALUE)
409                         .addComponent(btnAnalizarSin, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
410                     .addGapContainerGap())
411             );
412
413     btnLimpiarLex1.setBackground(new java.awt.Color(2, 35, 84));
414     btnLimpiarLex1.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
415     btnLimpiarLex1.setForeground(new java.awt.Color(255, 255, 255));
416     btnLimpiarLex1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/23-Full_Trash_256x256_35388.png"))); // NOI18N
417     btnLimpiarLex1.setText("Limpiar");
418     btnLimpiarLex1.addActionListener(new java.awt.event.ActionListener() {
419         public void actionPerformed(java.awt.event.ActionEvent evt) {
420             btnLimpiarLex1ActionPerformed(evt);
421         }
422     });
423
424     btnLimpiarLex2.setBackground(new java.awt.Color(2, 35, 84));
425     btnLimpiarLex2.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
426     btnLimpiarLex2.setForeground(new java.awt.Color(255, 255, 255));
427     btnLimpiarLex2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/codigo/business_man_user_support_supportfortheuser_aquestion_the")));
428     btnLimpiarLex2.addActionListener(new java.awt.event.ActionListener() {
429         public void actionPerformed(java.awt.event.ActionEvent evt) {
430             btnLimpiarLex2ActionPerformed(evt);
431         }
432     });
433
434     btnLimpiarLex3.setBackground(new java.awt.Color(2, 35, 84));
```



## **MANUAL DE PRÁCTICAS**





## **MANUAL DE PRÁCTICAS**



```
501         .addGap(1, 1, 1))
502         .addGroup(jPanellLayout.createSequentialGroup()
503             .addContainerGap()
504             .addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
505                 .addComponent(btnLimpiarLex1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
506                 .addComponent(btnAnalizarLex, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
507                 .addComponent(btnLimpiarLex, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
508             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
509             .addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
510                 .addGroup(jPanellLayout.createSequentialGroup()
511                     .addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
512                         .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 379, javax.swing.GroupLayout.PREFERRED_SIZE)
513                         .addGroup(javax.swing.GroupLayout.Alignment.LEADING, jPanellLayout.createSequentialGroup()
514                             .addGap(20, 20, 20)
515                             .addComponent(btnLimpiarLex2)
516                             .addGap(83, 83, 83)
517                             .addComponent(btnLimpiarLex3)
518                             .addGap(77, 77, 77)
519                             .addComponent(btnLimpiarSin1)))
520                         .addGap(18, 18, 18)
521                         .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
522                         .addComponent(jScrollPane1))
523                     .addGap(11, 11, 11))
524     );
525
526     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
527     getContentPane().setLayout(layout);
528     layout.setHorizontalGroup(
529         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
530             .addGroup(layout.createSequentialGroup()
531                 .addGap(18, 18, 18)
532                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
533                 .addGap(18, 18, 18)
534             );
535             layout.setVerticalGroup(
536                 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
537                     .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
538             );
539
540     pack();
541 } // </editor-fold>
542
543 private void btnArchivoActionPerformed(java.awt.event.ActionEvent evt) {
544     // TODO add your handling code here:
545     JFileChooser chooser = new JFileChooser();
546     chooser.showOpenDialog(null);
547     File archivo = new File(chooser.getSelectedFile().getAbsolutePath());
548
549     try {
550         String ST = new String(Files.readAllBytes(archivo.toPath()));
551         txtResultado.setText(ST);
552     } catch (FileNotFoundException ex) {
553         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
554     } catch (IOException ex) {
555         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
556     }
557 }
558
559 private void btnLimpiarLexActionPerformed(java.awt.event.ActionEvent evt) {
560     // TODO add your handling code here:
561     txtAnalizarLex.setText(null);
562 }
563
564 private void btnLimpiarsinActionPerformed(java.awt.event.ActionEvent evt) {
565     // TODO add your handling code here:
566     txtAnalizarsin.setText(null);
567 }
```



# MANUAL DE PRÁCTICAS



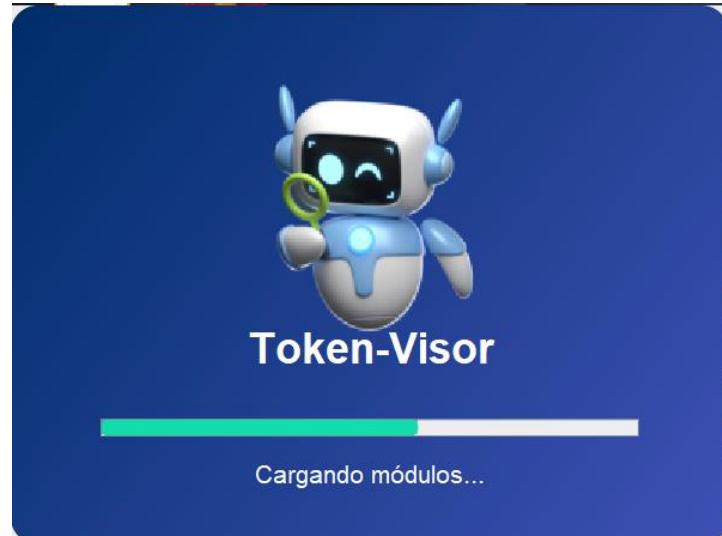
```
567     }
568
569     private void btnAnalizarLexActionPerformed(java.awt.event.ActionEvent evt) {
570         try {
571             analizarLexico();
572         } catch (IOException ex) {
573             Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
574         }
575     }
576
577     private void btnLimpiarLex1ActionPerformed(java.awt.event.ActionEvent evt) {
578         // TODO add your handling code here:
579         txtResultado.setText(null);
580     }
581
582     private void btnLimpiarLex2ActionPerformed(java.awt.event.ActionEvent evt) {
583         // Crear una nueva ventana (JFrame) para mostrar la imagen
584         JFrame imageFrame = new JFrame("Imagen de Ayuda");
585         imageFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
586
587         // Cargar la imagen desde el proyecto
588         ImageIcon imageIcon = new ImageIcon(getClass().getResource("/codigo/3.1.png")); // Cambia la ruta a la de tu imagen
589
590         // Obtener la imagen original
591         Image originalImage = imageIcon.getImage();
592
593         // Calcular el 90% de las dimensiones originales
594         int originalWidth = imageIcon.getIconWidth();
595         int originalHeight = imageIcon.getIconHeight();
596         int scaledWidth = (int) (originalWidth * 0.67);
597         int scaledHeight = (int) (originalHeight * 0.67);
598
599         // Redimensionar la imagen al 90% de su tamaño original
600
601         Image resizedImage = originalImage.getScaledInstance(scaledWidth, scaledHeight, Image.SCALE_SMOOTH);
602
603         // Crear un nuevo ImageIcon con la imagen redimensionada
604         ImageIcon resizedIcon = new ImageIcon(resizedImage);
605
606         // Crear un JLabel y establecer la imagen redimensionada
607         JLabel imageLabel = new JLabel(resizedIcon);
608
609         // Agregar el JLabel al JFrame
610         imageFrame.add(imageLabel);
611
612         // Ajustar el tamaño de la ventana al tamaño de la imagen redimensionada
613         imageFrame.pack();
614
615         // Hacer visible el JFrame
616         imageFrame.setVisible(true);
617     }
618
619     private void btnLimpiarLex3ActionPerformed(java.awt.event.ActionEvent evt) {
620         // TODO add your handling code here:
621         FrmCreditos cred = new FrmCreditos();
622         cred.setVisible(true);
623     }
624
625     private void btnLimpiarSin1ActionPerformed(java.awt.event.ActionEvent evt) {
626         // TODO add your handling code here:
627         this.dispose(); // Cierra la ventana actual
628     }
629
630     private void btnLimpiarSin1MouseReleased(java.awt.event.MouseEvent evt) {
631         // TODO add your handling code here:
632     }
```



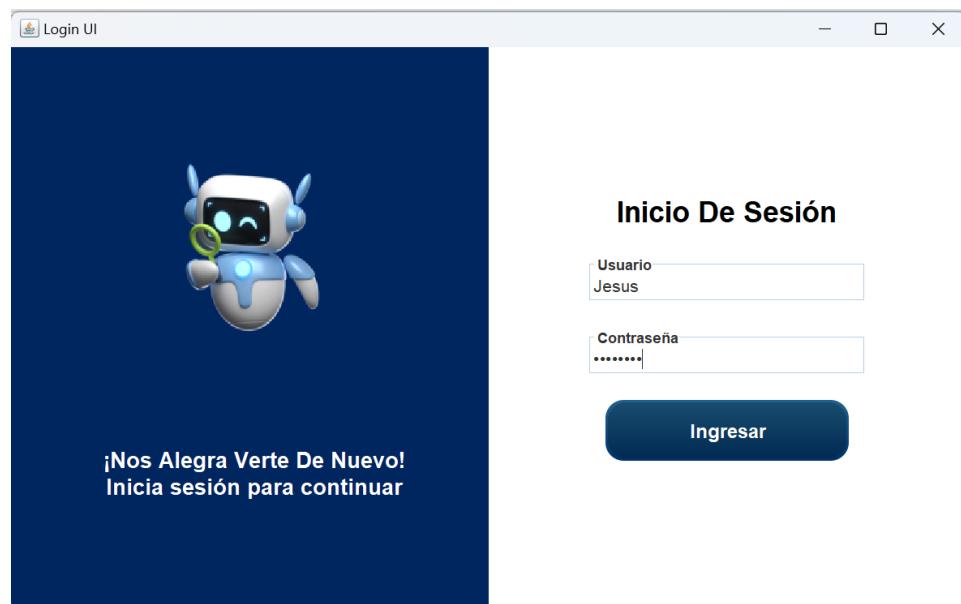
```
633 // Botón
634 private void btnAnalizarSinActionPerformed(java.awt.event.ActionEvent evt) {
635     String ST = txtResultado.getText();
636     Sintax s = new Sintax(new codigo.LexerCup(new StringReader(ST)));
637     s.setTextoCompleto(ST); // Pasa el texto completo al parser
638
639     try {
640         s.parse();
641         if (s.getErrores().isEmpty()) {
642             txtAnalizarSin.setText("Análisis realizado correctamente");
643             txtAnalizarSin.setForeground(new Color(18, 251, 197));
644         } else {
645             StringBuilder errores = new StringBuilder();
646             for (String error : s.getErrores()) {
647                 errores.append(error).append("\n");
648             }
649             txtAnalizarSin.setText(errores.toString());
650             txtAnalizarSin.setForeground(Color.red);
651         }
652     } catch (Exception ex) {
653         txtAnalizarSin.setText("Error inesperado durante el análisis.");
654         txtAnalizarSin.setForeground(Color.red);
655     }
656 }
657
658
659 /**
660 * @param args the command line arguments
661 */
662 public static void main(String args[]) {
663     /* Set the Nimbus look and feel */
664     /* Look and feel setting code (optional)
665
666     /* Create and display the form */
667     java.awt.EventQueue.invokeLater(new Runnable() {
668         public void run() {
669             new FrmPrincipal().setVisible(true);
670         }
671     });
672 }
673
674 // Variables declaration - do not modify
675 private javax.swing.JButton btnAnalizarLex;
676 private javax.swing.JButton btnAnalizarSin;
677 private javax.swing.JButton btnArchivo;
678 private javax.swing.JButton btnLimpiarLex;
679 private javax.swing.JButton btnLimpiarLex1;
680 private javax.swing.JButton btnLimpiarLex2;
681 private javax.swing.JButton btnLimpiarLex3;
682 private javax.swing.JButton btnLimpiarSin;
683 private javax.swing.JButton btnLimpiarSin1;
684 private javax.swing.JPanel jPanel1;
685 private javax.swing.JPanel jPanel2;
686 private javax.swing.JScrollPane jScrollPane1;
687 private javax.swing.JScrollPane jScrollPane2;
688 private javax.swing.JScrollPane jScrollPane3;
689 private javax.swing.JTextArea txtAnalizarLex;
690 private javax.swing.JTextArea txtAnalizarSin;
691 private javax.swing.JTextArea txtResultado;
692 // End of variables declaration
693 }
```

PANTALLAS RESULTANTES CON PRUEBAS DEL ANALIZADOR SINTACTICO

Pantalla Splash

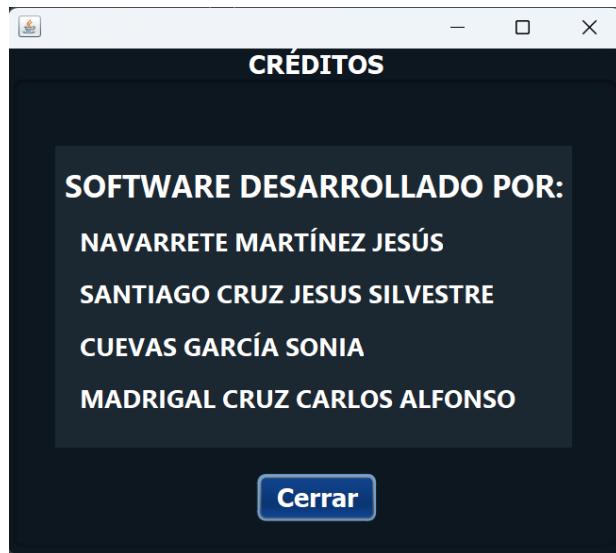


Pantalla Login





## Pantalla de creditos



## Pantalla de ayuda para el uso del software

Imagen de Ayuda

### SINTAXIS

Tipos de Datos		Operadores	
Ent = 1	Numericos	== Igualdad	& Y lógico
Flo = 1.1	*	> Mayor que	O lógico
Bool = v / f	/	< Menor que	! negación lógica
Cad = "Hola"	%	>= Mayor Igual	
Carrs = 'a'	^	<= Menor Igual	

**Estructura Básica**

```
Clase main () {
    Ent prueba = 1 ;;
}
```

**Condicionales**

```
Si (id == 1) {
    Cad salud = "Hola";;
}
Si (id == 1) {
    Cad salud = >"Hola";;
} siNo {
    Cad desp => "Adios";;
}
```

**Función segun**

```
segun (id) {
    caso Ent :
        Ent a >> 1;;
        Detener;;
    predeterminado :
        Cad b = "Hola";;
        predeterminado;;
}
```

**Declaraciones**

```
Constante d => 10 ;;
Variable id => 10 ;;
```

**Variables**

```
Imprimir (id);;
Leer (id);;
```

**Constantes**

**Comparación**

**Lógicos**

**Comentarios**

// Comentario entre líneas

**Ciclos**

**Do While**

```
hacer {
    prueba++;
} mientras (id == 1) {
    prueba++;
}
```

**While**

```
mientras (id == 1) {
    prueba++;
}
```

**For**

```
ciclo (holo >= 5 ; ser > 2 ; hola++);
    Ent hola++;
}
```

**Constantes**

**Variables**

**Constantes**

**Variables**

Pi E

**¿CÓMO UTILIZAR EL SOFTWARE?**

Esta herramienta permite procesar un código escrito en un lenguaje específico identificando sus tokens a través del analizador léxico y verificando su estructura gramatical con el analizador sintáctico.

**¿Cómo utilizar el analizador?**

- Ingresar el código :** El código puede ser ingresado de dos maneras:
  - Archivo de texto : Puedes cargar un archivo.txt que contiene el código a analizar, asegúrese de que el archivo esté en el formato correcto.
  - Entrada manual: Si prefieres ingresar el código directamente, puedes hacerlo a través de la interfaz proporcionada.
- Realizar el análisis léxico:** Da clic en el botón Analizar, esto convierte el código en una lista de tokens que representan unidades significativas (como palabras clave, identificadores, operadores, etc).
- Procesar los tokens generados:** Una vez realizado el análisis léxico, haz clic en el botón ("Analizar") dentro de la sección del Analizador Sintáctico. Esto verificará si los tokens siguen las reglas de la gramática definida para el lenguaje y mostrará los resultados en el cuadro inferior.

**Nota:** Si deseas reiniciar el análisis o limpiar los datos, haz clic en los botones ("Limpiar"). Esto eliminará tanto el contenido ingresado como los resultados mostrados.

FO-ACA-11

Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditabile.

Versión 1

Fecha: 25/10/2018



GOBIERNO DEL  
ESTADO DE MÉXICO

# MANUAL DE PRÁCTICAS



## Pantalla Principal

Analizador Lexico

Abrir Archivo Limpiar Analizar Limpiar

1

Analizador Sintactico

Analizar Limpiar

Cerrar

Iconos laterales: Ayuda, Comunidad, Cerrar

Diseño: La interfaz tiene un tema oscuro con cuadros blancos para los paneles de contenido. Los botones están en cuadros azules con iconos y texto. Los paneles principales están vacíos.



## Pantalla principal con demostración de verificación de tokens y un análisis correcto

Análizador Lexico

Abrir Archivo      Limpiar      Analizar      Limpiar

```
1 Clase hola(){
2 Ent edad=> 5;-
3 Ent edad2=>8;-
4 Ent edad3=> 51;-
5 Ent cantidad =>25;-
6 Flot precio => 7.7 ;-
7 Imprimir (precio) ;-
8 suma + suma2 ;-
9
10 ciclo ( sueldo => 5 ;; Boo ;; sueldo++ ){
11 Ent dia ++ ;;
12 }
13
14 segun (dia){
15 caso Ent: Flot precio => 7.7 ;; Detener ;;
16 predeterminado: Ent cantidad =>25; ;; Detener ;;
17 }
18
19 }
```

LINEA	SIMBOLO
1	Clase
2	edad
3	Identificador
4	parentizq
5	parentDer
6	llavelzq
7	LINEA
8	valorEntero
9	Identificador
10	asignacion
11	Número
12	finLinea
13	LINEA
14	valorEntero
15	Identificador
16	asignacion
17	Número
18	finLinea
19	LINEA
	valorEntero
	Identificador
	asignacion
	Número

Analizador Sintactico

Análisis realizado correctamente

Analizar      Limpiar



## Pantalla principal con demostración de verificación de tokens y detección de errores por sintaxis mal colocadas

Análizador Lexico

LINEA	SIMBOLO
1	clase
2	Ent
3	edad=
4	5;-;
5	Identificador
6	edad2=>8;-;
7	parentizq
8	parentDer
9	llaveizq
10	LINEA 2
11	valorEntero
12	Identificador
13	asignacion
14	Numero
15	finLinea
16	LINEA 3
17	valorEntero
18	Identificador
19	asignacion
	Numero

Abrir Archivo      Limpiar      Analizar      Limpiar      Cerrar

Análizador Sintactico

Error de sintaxis. Línea: 2, Columna: 10, Texto: "=="  
Error de sintaxis. Línea: 6, Columna: 21, Texto: "-;"  
Error de sintaxis. Línea: 8, Columna: 7, Texto: "suma2"  
Error de sintaxis. Línea: 10, Columna: 18, Texto: ">"  
Error de sintaxis. Línea: 10, Columna: 42, Texto: ")"  
Error de sintaxis. Línea: 15, Columna: 11, Texto: "Flot"

Analizar      Limpiar

## V. Conclusiones:

El objetivo principal del desarrollo del proyecto fue implementar un simulador de análisis léxico y sintáctico de un compilador, utilizando herramientas y tecnologías que permitieran procesar y analizar el código fuente ingresado por el usuario. El propósito era entender cómo se estructuran los lenguajes de programación a través de los conceptos de autómatas y gramáticas, y cómo un compilador procesa estos elementos. Para ello, se utilizó el lenguaje de programación **Java** debido a su robustez y facilidad para trabajar con bibliotecas de análisis, como **Java CUP** para el análisis sintáctico y **JFlex** para el análisis léxico. Estas bibliotecas tienen la capacidad de generar automáticamente los analizadores, lo que facilita su implementación y uso.

Durante el desarrollo del proyecto, se presentaron varias problemáticas, especialmente en el análisis léxico. La creación de un lexer que pudiera reconocer y clasificar correctamente los tokens fue un desafío importante, ya que se necesitaba una estructura clara para manejar palabras clave, identificadores, operadores y otros elementos del lenguaje. A veces, la integración entre el analizador léxico y el sintáctico presentaba errores al no identificar correctamente ciertos patrones o al no vincular adecuadamente los tokens con las reglas sintácticas definidas en el parser. Sin embargo, a través de la depuración y pruebas continuas, se logró superar estas dificultades, mejorando la precisión del análisis.

El resultado obtenido fue un software funcional que permitía a los usuarios ingresar código, cargar archivos y visualizar los resultados del análisis léxico y sintáctico en tiempo real. Además, se implementó una interfaz gráfica para facilitar la interacción con el usuario, permitiendo realizar el análisis de manera intuitiva. Este proyecto no solo demostró la importancia de la teoría aprendida en la materia de Lenguajes y Autómatas, sino que también vinculó aprendizajes previos de las unidades de autómatas finitos, gramáticas formales y expresiones regulares. Estos conocimientos son fundamentales para entender cómo los compiladores interpretan y procesan los lenguajes de programación.

La materia de Lenguajes y Autómatas 1 es esencial para mi carrera, ya que proporciona una comprensión profunda de los fundamentos de los lenguajes de programación, su estructura y cómo construir herramientas que los interpreten. El análisis léxico y sintáctico es la base de la creación de compiladores, y el dominio de estas técnicas abre el camino hacia el diseño de lenguajes de programación y herramientas de software eficientes. Además, aprender a manejar autómatas y gramáticas es clave para resolver problemas complejos en la ingeniería de software, especialmente cuando se trabaja en proyectos que requieren procesamiento de texto, compilación o diseño de lenguajes especializados.