Universidad de San Carlos de Guatemala
Centro universitario de Occidente
División de ciencias de la Ingeniería en Ciencias y Sistemas

Curso: Lenguajes Formales y de Programación
Catedrático: Ing. Oliver Sierra
Sección: “A”

Proyecto 2

Daniel Eduardo Fernández Ovando …… 201731816

Quetzaltenango, 19 de Noviembre del 2019.

Manual Técnico del “Analizador Léxico”

Lenguajes Formales y de programación

Sección “A”

Nombre: Daniel Eduardo Fernández Ovando

Carné: 201731816

Manual Técnico del Analizador Léxico:

INSTRUCCIONES DE USO (inicio, objetivos, instrucciones sobre el analizador):

Este Manual Técnico se hace con el fin de darle al cliente o usuario una mejor idea o forma de poder enseñarle como utilizar o que pueda ver cómo funciona este Analizador. El objetivo de este analizador es tratar de ver diferentes tipos de patrones y así este lenguaje que reconozca que hemos ingresado a la hora de ingresarle al Analizador un archivo de texto. Cada cliente podrá ingresar cualquier archivo de texto que quiere que este analizador quiera que analice.

Dentro de este proyecto deberán después de haber abierto un archivo de texto, cada cliente elijará si solo quiere modificarlo, y guardarlo o si no le parece este lo puede borrar con el botón delete. Ahora sino quiere ninguna de estas opciones podrá elegir el botón analizar donde el archivo de texto será analizado, y este ya detectara por medio de una gramática que tipo es el token que ingresemos, analizara token por token e irá recorriendo todo el archivo de texto hasta que termine de analizarlo, ya en un recuento nos mostrara lo analizado. Luego ya podremos ver si tiene errores, o que tipo de token es, y podremos ver que en que fila y columna tenemos el dicho error, los demás tokens que vaya viendo el analizador léxico dentro del archivo de texto.

Partes que componen al juego:

- Inicio
- Analizador
- Información o ayuda.
- Archivo
- Abrir archivo de texto
- Guardar archivo de texto
- Borrar archivo de texto
- Ayuda
- Analizador

- Archivo de salida
- Archivo de Errores
- Diagramas
- Tokens
- Autómata de pila
- Gramática

Elementos del Manual Técnico:

Objetivos Generales:

- Familiarizar al estudiante con el lenguaje Java y Autómatas.
- Elaborar la lógica para la solución del problema planteado.
- Aplicar conceptos de programación dada en clase y laboratorio.
- El realizar buenas prácticas

Objetivos Específicos:

- Elaborar un Diagrama de clases para mejor comprensión del programa, un diagrama de Moore para entender el autómata.
- Poder ampliar el conocimiento en JAVA
- Poder ampliar conocimiento en programación con pilas
- Implementación de clases, pilas, y autómatas
- Implementación de ciclos, sentencias de control y arreglos o lo que se necesite para poder programar.

Elementos de requerimiento para la funcionalidad del programa (juego):

***** Requiere PC con los Sistemas operativos siguientes**

- Sistema Operativo (Windows o Linux)
- Java 1.8.0_201 o compatibles.
- NetBeans IDEA (creación: v. 8.2) o cualquier editor de lenguaje JAVA.
- Intellij IDEA o cualquier otro IDE.
- Conocimientos en el lenguaje JAVA
- .Jar ejecutable o proyecto completo.
- Entender lenguaje JAVA (para ver creación de programa).
- Experiencia previa en Interfaz Gráfica

- **Elementos para la construcción del programa**

    - Proyecto JAVA
    - Paquetes JAVA
    - Clases en JAVA
    - Atributos de las clases
    - Métodos de los atributos
    - Objetos
    - Interfaz Gráfica
    - Botones
    - Lista para reportes en HTML
    - Recursividad
    - Polimorfismo
    - Autómatas
    - Pilas

Descripción del Programa

Este proyecto es un analizador léxico el cual llevara varias formas, se podrá abrir txt, se podrá guardar, se podrá borrar, luego ya se podrá analizar, luego se podrá ver los tokens, luego los errores, y podremos ver los manuales y la gramática.

- El proyecto debe de reconocer una gramática que tiene la siguiente estructura:

Función principal {variable entera x = 5;}

La función principal es obligatoria y es donde el programa inicia puede tener muchas instrucciones dentro de ella, incluyendo llamada a funciones. Se pueden realizar llamadas a funciones pero estas deben de definirse antes de usarse, por ejemplo:

función f1 (entero x){ Imprimir (“el número es “ + x); }

función principal {variable entera x = 5;
f1(x);
}

Las instrucciones se resumen a continuación:

● **imprimir ();** Dentro de los paréntesis puede venir la concatenación de variables, strings o char. La concatenación se realiza con signos +. Ejemplo imprimir (a + b + “hola mundo”);

● **La asignación de una variable** se puede realizar como en el lenguaje de programación java. Con la diferencia que se utilizan las palabras reservadas del lenguaje. Se pueden definir de todos los tipos mencionados arriba. Ejemplo: variable entero x = 5;

● **Operaciones matemáticas.** Para esto pueden utilizarse los paréntesis para agrupar operadores. Ejemplo: x = (x * x / (5 +5))/(1+ 2);

● para los casos del if, for, while, la gramática es la misma de java solo que con el uso de palabras reservadas del lenguaje.

JAVADOC:

file:///C:/Users/danie/Desktop/Proyecto2/P2D/dist/javadoc/index.html

Gramática libre de contexto:

G = {N, T, S, P}
S = {E}
N = {E, C, V, D, T, L}
T = {Parametro, NUM, DIGITO, OPERADOR}
P =

E	-->	funcion principal { **C** }
		funcion id (**PARAMETRO**) { **C** }
C	-->	**V ;**
		V = **T L** ;
		imprimir ( );
		while( ){ **C** }
		if ( ) { C }
		for (**V** = **NUM ; id < NUM ;** ID++) { **C** }
		id (id) ;
		/*comentario/*
V	-->	variable **D** id
D	-->	entero
		decimal
		booleano
		cadena
		caracter
T	-->	**NUM**
		NUM . **NUM**
		true
		false
		"cadena"
L	-->	id **OPERADOR** id
		|e
PARAMETRO	-->	**D** id
		|e
NUM	-->	**DIGITO DIGITO**
		|e
DIGITO	-->	0|1|2|3|4|5|6|7|8|9
		|e
OPERADOR	-->	+|-|*|/|%|=|==|<|>|>=|<=
		|e

Gramática LL1:

Tabla de primeros:

No.Terminal	Primeros
E	funcion principal, ID
C	Variable, imprimir, while, if, for, id, comentario
V	Variable
D	entero, decimal, booleano, carácter, cadena
T	0, 1, 2, 3, 4, 5, 6 , 7, 8 , 9, true, false, cadena
L	Id, e
Parametro	entero, decimal, booleano, carácter, cadena, e
Num	0, 1, 2, 3, 4 , 5 ,6 ,7 ,8 ,9
Digito	0, 1, 2, 3, 4 , 5 ,6 ,7 ,8 ,9
Operador	+|-|*|/|%|=|==|<|>|>=|<=

Tabla de siguientes:

No.Terminal	Siguientes
E	$
C	$
V	; ,
D	id
T	id, e, $
L	;
Parametro	variable, imprimir, while
Num	; , id, e, $
DIGITO	; , id, e , $
OPERADOR	id

Tabla sintáctica:

	funcion principal	**id**	**variable**	**imprimir**	**while**	**if**
E	funcion principal { **C** }	funcion ID (PARAMETRO) { C }				
C		for (**V** = **NUM ;** ID < **NUM ;** ID++) { **C** }	V ;	**V** = **T L** ;	impri mir ( );	while( ){ **C** }
V			variable **D** ID			
D						
T						
L		ID **OPERADOR** ID				
Parametro			e	e	e	
Num						
Digito						
Operador						

for	**entero**	**decimal**	**booleano**	**carácter**	**cadena**	**true**
if ( ) { C }						
	entero	decimal	booleano	carácter	cadena	
					false	NUM.NUM
	D ID	e	e	e	e	

false	1, 2, 3,4,5,6,7,8,9	+|-|*|/|%|=|==|<|>|>=|<=	;	$
true	NUM			
			e	
	DIGITO DIGITO			
	0|1|2|3|4|5|6|7|8|9			
		+|-|*|/|%|=|==|<|>|>=|<=		

Analizador:

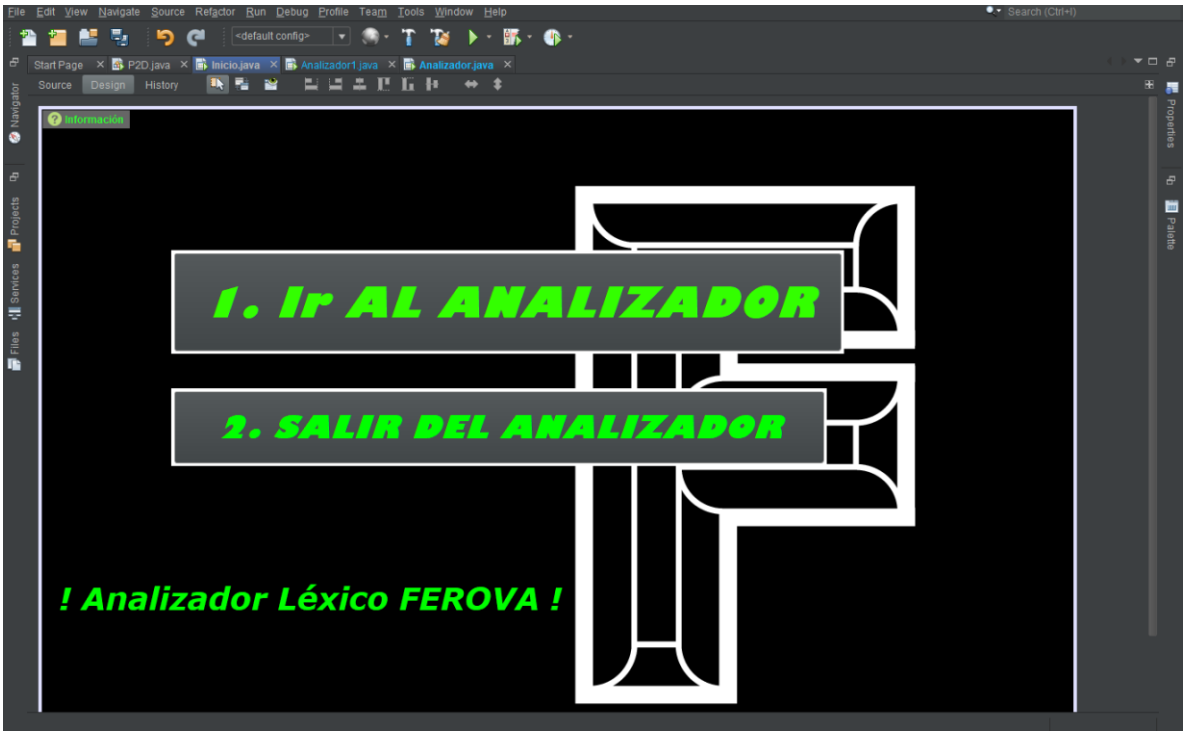
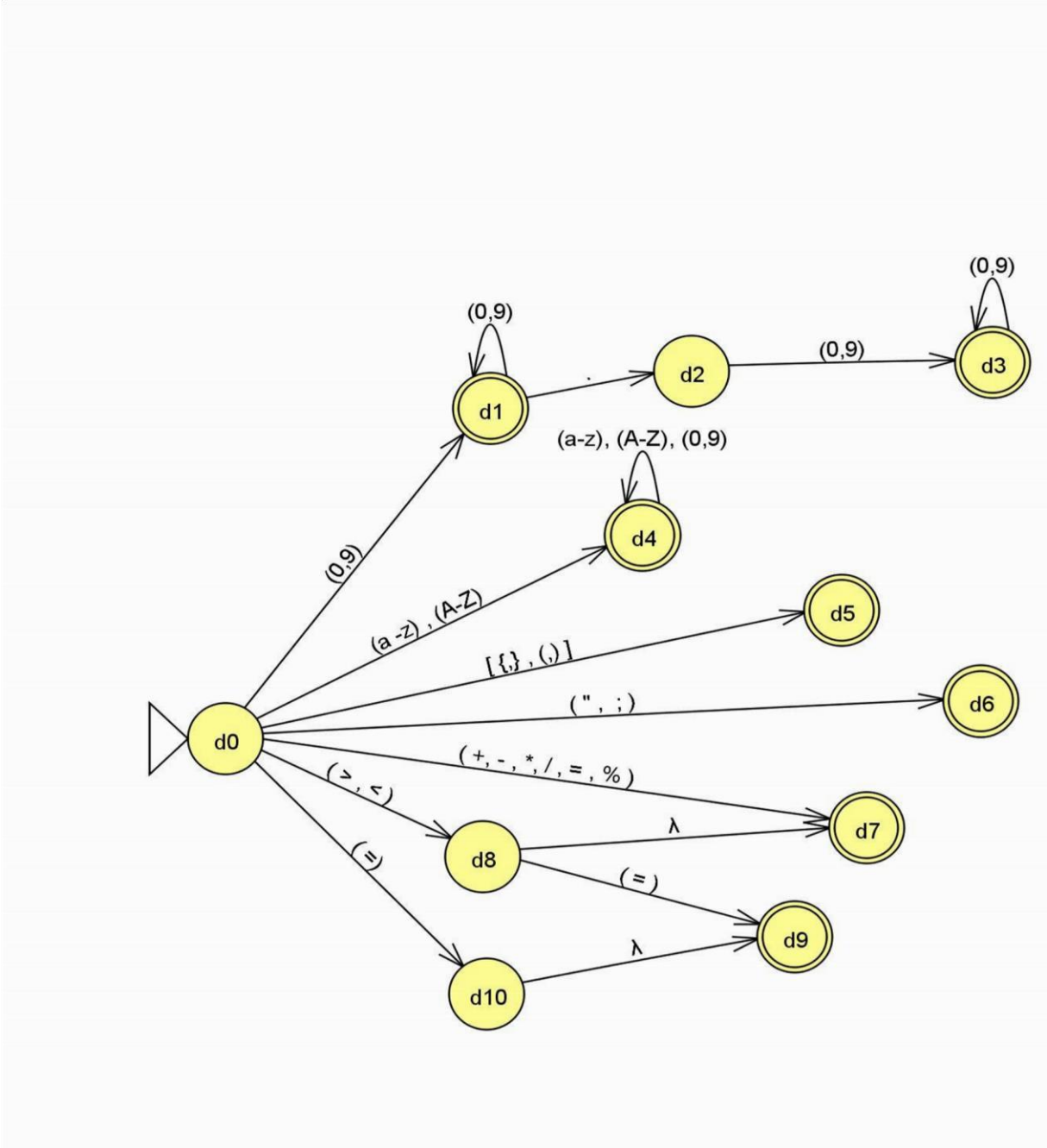


Diagrama de Moore:



Alfabeto:

Analizador léxico: Un analizador léxico o analizador lexicográfico (en inglés scanner) es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes léxicos) o símbolos. Estos tokens sirven para una posterior etapa del proceso de traducción, siendo la entrada para el analizador sintáctico.

Analizador Sintáctico: Un analizador sintáctico (o parser) es un programa informático que analiza una cadena de símbolos de acuerdo a las reglas de una gramática formal. El término proviene del latín pars, que significa parte (del discurso). Usualmente hace parte de un compilador, en cuyo caso, transforma una entrada en un árbol sintáctico de derivación.

JAVA: es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable}

NetBeans: es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Autómata: Un autómata es un modelo matemático para una máquina de estado finito (FSM sus siglas en inglés). Una FSM es una máquina que, dada una entrada de símbolos, "salta" a través de una serie de estados de acuerdo a una función de transición (que puede ser expresada como una tabla).

Pila y cola: Las pilas y colas son estructuras de datos que se utilizan generalmente para simplificar ciertas operaciones de programación. Estas estructuras pueden implementarse mediante arrays o mediante listas enlazadas.

Gramática: Gramática (autómata) Una gramática ("G") desde el punto de vista de la teoría de autómatas es un conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un lenguaje específico L.