UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

MANUAL TÉCNICO

JUAN ANTONIO SOLARES SAMAYOA
CARNET 201800496
MIERCOLES 16 DE SEPTIEMBRE DE 2020

RESUMEN DE LA APLICACIÓN

La aplicación consiste en un analizador léxico, que analiza los lenguajes HTML, CSS y JavaScript junto con un analizador sintáctico de expresiones aritméticas que están contenidos en archivos de formato .rmt, desarrollada en el lenguaje de programación *Python.*

El análisis léxico (también denominado escáner) es la primera fase del proceso de compilación, que consiste en el reconocimiento de tokens por medio de patrones definidos por una gramática a partir de un código fuente.

DEFINICIÓN DE CLASES UTILIZADAS

CLASES UTILIZADAS PARA ANALISIS LEXICO: Estos analizadores consisten en reconocer el archivo con código fuente de la entrada para reconocer patrones e irlos agregado a una lista de tokens. Durante este mismo proceso, se van reconociendo errores léxicos y agregándolos a una lista para posteriormente se mostrados en el reporte de errores al finalizar el análisis.

En los primeros dos líneas de la aplicación, habrán dos rutas (representados) por un comentario las cuales indican la ruta de reubicación del archivo sin errores, de acuerdo al sistema operativo en donde se esté ejecutando la aplicación. Para esta aplicación la función de reubicación de archivos es funcional solamente para el sistema operativo Windows.

- AnalizadorLexicocss.py
- AnalizadorLexicoJS.py
- AnalizadorLexicohtml.py
- Lexrmt.py

Cada una de las clases anteriores contienen los siguientes métodos, a continuación se muestran los más importantes:

- Escanear: recibe como parámetro la cadena de texto que contiene el código fuente a analizar, para posteriormente analizarlo en cada uno de los estados definidos en el AFD
- AgregarError: Agrega los símbolos o caracteres que no formen parte del lenguaje que se está analizando a una lista, para posteriormente ser mostrados en un reporte.
- AgregarToken: Reconoce un token que cumple con un patrón establecido
- GenerarReporte: Genera el reporte de errores léxicos encontrados durante el análisis, en formato HTML.
- GenerarSalida: Genera el archivo de salida libre de errores léxicos, se almacenará en una ruta que está en las primeras líneas del código fuente.

CLASE UTILIZADA PARA ANALISIS SINTACTICO

- Sintactico.py

En esta clase está compuesta por dos fases, detalladas a continuación:

ANALISIS SINTACTICO

Para la implementación del análisis sintáctico, se utilizó un método propio, que está conformado por dos partes:

- La primera parte consiste en el reconocimiento de los caracteres de una expresión aritmética, carácter por carácter, hasta encontrar un salto de línea, al momento de ser detectado un salto de línea, se agrega la expresión completa en una lista. Se realiza el proceso anterior para cada una de las líneas que estén contenidas en la caja de texto de la entrada.
- La segunda parte consiste en el análisis detallado de cada expresión, analizando cada uno de los caracteres. Se utilizó una pila para verificar que la cantidad de paréntesis tanto de apertura como cierre sean las adecuadas, además de validaciones como verificación de signos y puntos.

Ejemplos de expresiones aritméticas correctas:

1. ((5+6*var1)+(6+8*33-12))
2. 50.25+50.25*(35)+(((5+2)))
3. Var1+var2-(((4+((3+4*x)))))

Ejemplos de expresiones aritméticas incorrectas

1. 1+2.33+3*879+/56 -> dos operadores juntos
2. 5.f+22-33+((3+444)) -> una letra después del punto decimal
3. 5...3+7+9 -> tres puntos decimales
4. ((3+5))-(((3+2-(8) -> error de paréntesis

AUTOMATAS UTILIZADOS PARA PROGRAMAR EL ESCANER

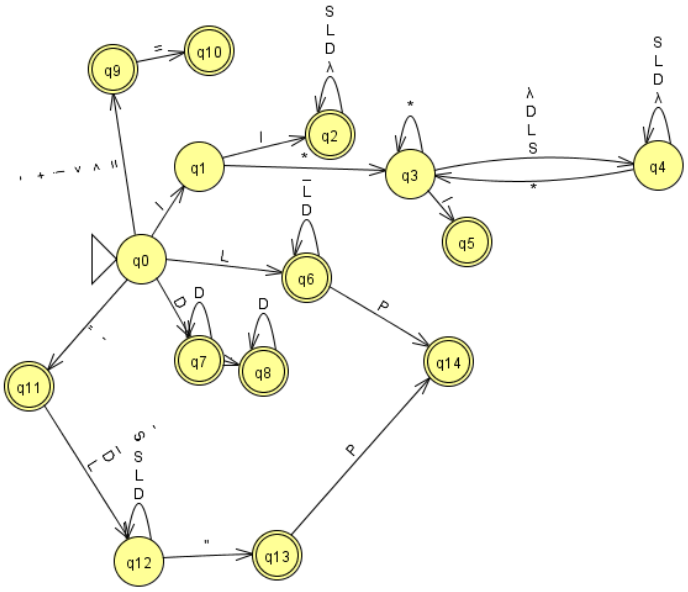
Significado de las letras: (aplica para todos los autómatas)

S = símbolo
L = Letra
D = Digito

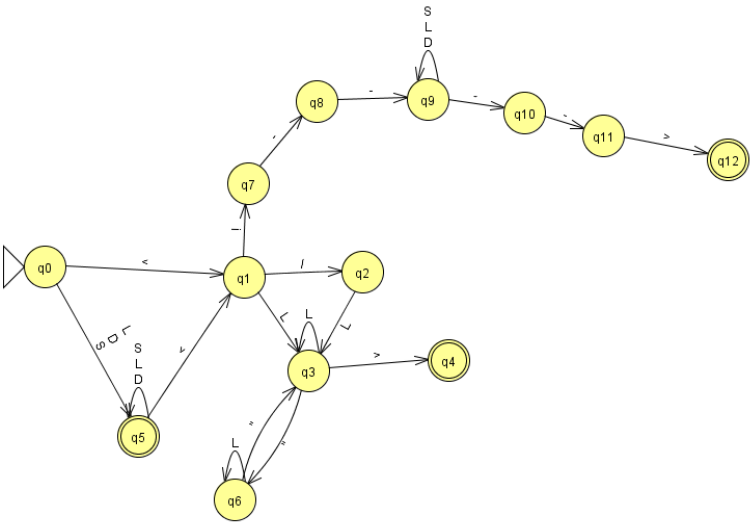
P = Paréntesis
>, <, _, -, /, * = Otros símbolos

LOS SIGUIENTES AUTOMATAS SON IMPLEMENTACIÓN PROPIA DEL ESTUDIANTE.

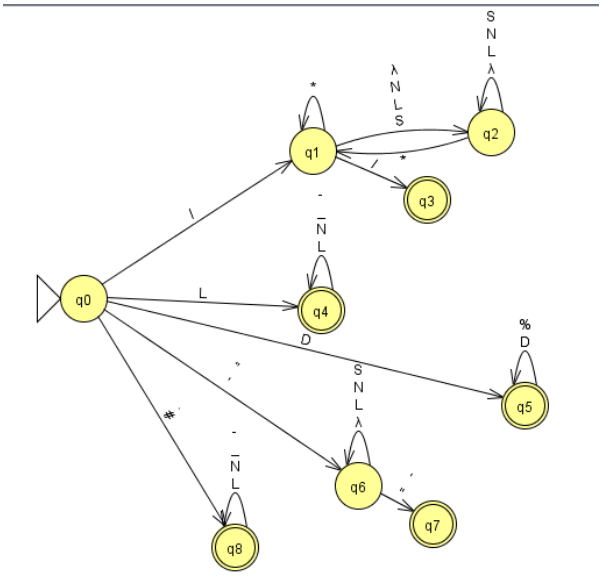
- JAVASCRIPT



- HTML



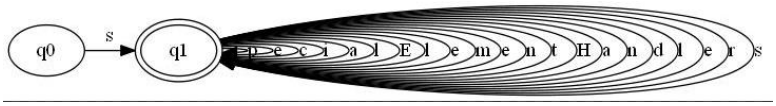
- CSS



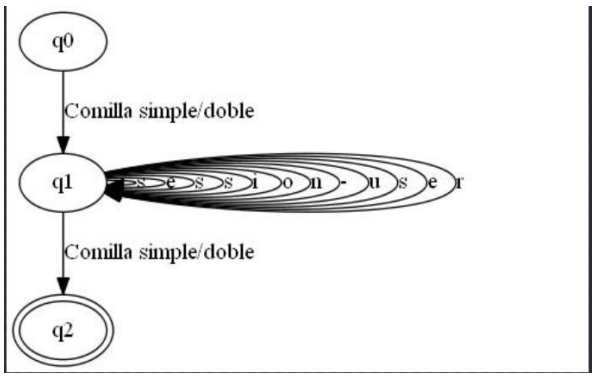
AUTOMATAS RECONOCIDOS POR MEDIO DE GRAPHVIZ

Durante el reconocimiento de tokens del lenguaje JavaScript, se mostrarán tres autómatas en formato jpg que serán formados a medida que se reconozcan los tokens en el análisis léxico.

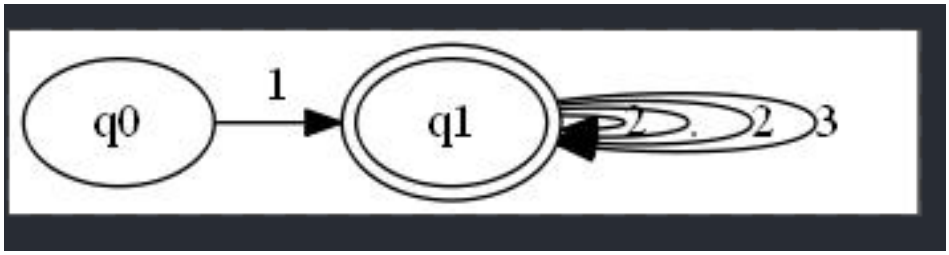
- Autómata para ID



- Autómata para Cadena de texto



- Autómata para digito



OTRAS HERRAMIENTAS UTILIZADAS

- Para realizar la interfaz gráfica de la aplicación, se utilizó la librería Tkinter de Python, la cual provee las herramientas necesarias para generar interfaces graficas atractivas al usuario.

- Para el desarrollo de la aplicación se utilizó la librería graphviz para la generación de los autómatas (grafos) que representa el proceso de reconocimiento de tokens (ID, Digito, cadena de texto).

GLOSARIO DE TERMINOS

- **Error Léxico:** carácter no perteneciente a determinado lenguaje. Por ejemplo, un símbolo que no puede formar parte de una sentencia de un lenguaje de programación, como ($, % &, ¡, @). Los símbolos reconocidos como error léxico pueden variar de acuerdo al lenguaje o gramática analizada.
- **AFD:** Autómata Finito Determinista
- **ID:** Identificador.