10-3-2021

# Manual Técnico.

## REGEXIVE

Ariel Rubelce Macario Coronado
UNIVERSISDAD DE SAN CARLOS DE GUATEMALA.

# DESCRIPCIÓN GENERAL DE LA SOLUCIÓN.

REGEXIVE es una solución para que los estudiantes del curso de Organización de lenguajes y compiladores 1 puedan verificar la solución de sus respuesta de ejercicios, tareas incluso exámenes, la solución cuenta con la generación de método de árbol, método de Thompson, tabla de siguientes, de transiciones, AFD, se quiere que las expresiones a ingresar sean en notación polaca.

# REQUERIMIENTOS MÍNIMOS DEL ENTORNO DE DESARROLLO.

Para el desarrollo de la solución se debe de contar con tener un IDE para JAVA, contar con el JDK y JRE, contar con un editor de código y contar con un espacio para el desarrollo.

# DICCIONARIO DE CLASES.

**Analizadores:** Es aquella clase que nos ayuda a compilar los archivos léxico.flex y Sintactico.cup.

**AFD:** Clase que contiene como parámetros un nombre, una transición y un alfabeto.

**Conjunto:** Clase que define un conjunto, que tiene como parámetros, un nombre y un rango.

**Errores:** Clase que define un error este tiene como parámetro, tipoErrores, valorError, fila y columna.

**Interfaz:** Clase que tiene como función ejecutar nuestra Jframe para poder visualizar la ventaja.

**Léxico:** Clase que se genera al compilar léxico.flex, esta clase define nuestro alfabeto y caracteres especiales que usará nuestro alfabeto.

**Nodo:** Clase que define nuestros nodos de un árbol, clase que también se encarga de generar las gráficas, árbol, tablas y AFD.

**Parser:** Clase que se genera al compilar el archivo Sintactico.cup, esta se encarga de declarar nuestros terminales, no terminales y producciones.

**SiguientesN:** Clase que se define nuestros siguientes, tiene como parámetros, numHoja, nombreHoja, siguiente.

**Sym:** Clase que se encarga de definir nuestro alfabetos del léxico.

**TransicionN:** Clase que define una transición, tiene como parámetros nombreT, transicionT.

# DICCIONARIO DE METODOS.

**Cadena:** valida la cadena de archivo de entrada.

**Estadosarray**: estados que son posibles para la tabla de transiciones.

**getAFND:** Genera las distintas posiciones para graficar el AFND.

**getAFD**: Genera las posiciones para graficar el AFD.

**getCodigoInterno**: Genera las posiciones para realizar el grafico del Árbol binario.

**getEncabezado**: genera el encabezado de la tabla de transiciones.

**getTableInterna**: genera la tabla de siguientes de la expresión.

**getTableTransiciones**: genera la tabla de transiciones completa.

**graficarAFD**: genera el archivo .dot y genera la imagen de la grafica.

**graficarAFND**: genera el archivo .dot Y genera la imagen de la grafica.

**graficarTablaransciones**: genera el archivo .dot y genera la imagen de la tabla de transiciones.

**graficarTablaSiguientes**: genera el archivo .dot y genera la imagen de la tabla de siguientes.

**Mostrarimg**: muestra las graficas en la ventaja de jFrame.

**preOrden**: recorre el árbol con el recorrido preOrden.

**preordenArbolito**: recorre el árbol con el recorrido preOrden, pero este árbol solo lo recorre sin el numeral.

**ReporteErrores**: realizar el html de reporte de errores.

# DIAGRAMA DE CLASES.