**Universidad de San Carlos de Guatemala**

Facultad de Ingeniería

Organización De Lenguajes Y Compiladores 1

Cat. Ing. Bautista

***Manual Técnico***

***Proyecto #1***

Jeffry Emanuel Méndez Diaz

Carnet: 20191557

***REGEXIVE***

# DESCRIPCIÓN

El programa Regexive se trata de un software de desarrollado con el fin de brindar herramientas para trabajar de forma efectiva con gramáticas y expresiones regulares en notación polaca o prefija. La aplicación tiene la capacidad de reconocer un archivo de entrada, donde se ingresarán comentarios, conjuntos validos de caracteres, expresiones regulares y evaluaciones de las mismas. En el caso de error se crearán reportes y se harán salidas en consola. Por cada expresión regular que el usuario defina se generan el árbol de la expresión, la tabla siguiente y de transiciones. Y para finalizar se crea un AFD para poder valuar las entradas. Se cuenta con un visualizador de los grafos generados para una mayor utilidad. El programa se desarrollo con ayuda de JLex y Cup para el manejo de la gramática.

**ESPECIFICACIONES**

|  |  |
| --- | --- |
| Netbeans IDE | Versión 8.2 RC |
| Java | Versión 1.8.0\_281 |
| Java(TM) SE Runtime Enviroment | Build 1.8.0\_281-b09 |
| Java HotSpot(TM) 64-Bit Server VM | Build 25.281-b09 |
| Sistema Operativo | Windows 10 |

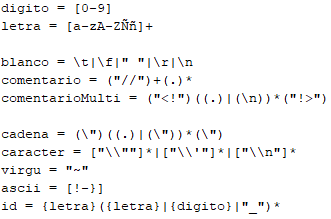
**MANEJO DE ARCHIVOS**

Para manejar los archivos entrantes se utilizan las clases File y se creo una clase llamada FileTypeFilter, esta sirve para filtrar y permitir únicamente la entrada de los archivos con la extensión requerida (OLC). Al momento de crear un archivo se hace con un nombre con numero aleatorio para evitar conflictos, al momento de guardar como, únicamente se crea otro archivo con el nuevo nombre que el usuario elija. En la opción de guarda solamente se asigna al archivo ya existente el nuevo texto que se ingrese. Al abrir un archivo se queda guardada la ubicación para realizar las operaciones ya mencionadas.

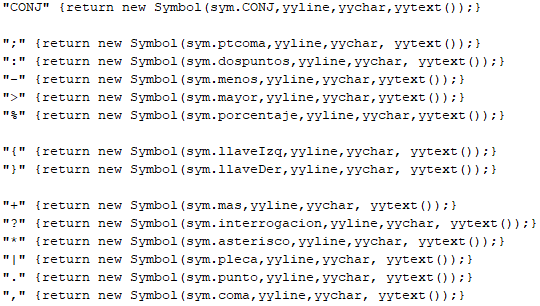


**ANALISIS LEXICO – JLEX**

* Para reconocer todos los caracteres de las entradas que se realizaran por los usuarios, se definieron las siguientes expresiones regulares:



* Con los demás caracteres se crearon símbolos para utilizarlos en el análisis sintáctico.



**ERRORES LEXICOS**

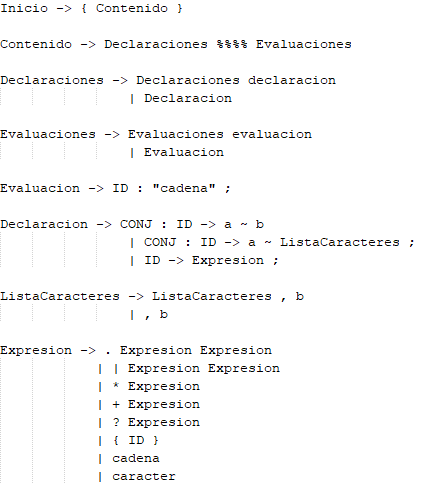
Al momento de que ocurre un error léxico, se crea un nuevo objeto de la clase Error File y se agrega a la lista general.

Posteriormente se procede a enseñarlo en consola.



**ANALISIS SINTACTICO – CUP**

* Para reconocer todas las cadenas de instrucciones que escriba el usuario, se utilizo la siguiente gramática. Se realizo implementando un análisis ascendente.



En el apartado de las producciones de Expresión, cada una se maneja como un nodo que puede tener conexiones tanto a la izquierda como a la derecha, según sea el símbolo de la expresión se ira armando el árbol binario. Al finalizar se crean los nodos de concatenación final y el de la hoja de aceptación (#).

**ERRORES SINTACTICOS**

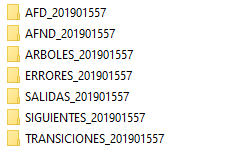
Al momento de que ocurre un error sintáctico, se crea un nuevo objeto de la clase Error File y se agrega a la lista general, si es recuperable se ignora esa línea y continua con la siguiente.

Posteriormente se procede a enseñarlo en consola.



**GRAFOS**

Al momento de generar los grafos de las expresiones, estos se almacenas en sus correspondientes carpetas identificadas con su nombre seguido de un numero de carnet. En cada una por expresión ingresada se crea un archivo .dot y la imagen .png que este genera al momento de ejecutar el comando de DOT.

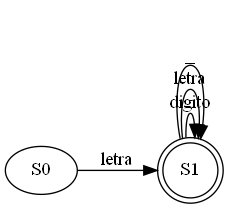
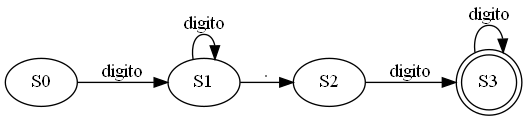
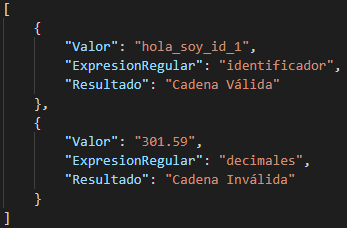




**ANALIZAR ENTRADAS**

Para poder analizar las entradas correctamente, primero se realizó el método del árbol para que desde la expresión que brinda el usuario, poder crear nuestra tabla de transiciones y posteriormente simular la validación a través de un AFD. Se recorren todos los caracteres de la cadena de evaluación y utilizando las transiciones se realiza un recorrido entre los estados hasta que o la cadena llego a su fin o que un carácter no sea validado. Independiente del resultado se crea un JSON donde están todas las evaluaciones que se realización con su respectiva conclusión.

Ejemplo:



**DIAGRAMA DE FLUJO DEL PROGRAMA**

