**Universidad de San Carlos de Guatemala**

**Facultad de Ingeniería**

**Escuela de Ciencias y Sistemas**

**Laboratorio de Lenguajes Formales y de Programación**

**Manual Técnico**

Proyecto 1: Analizador léxico

**Catedrático:** Mario Josué Solis Solórzano

**Sección:** A-

**Nombre:** Josué Daniel Rojché García

**Carné:** 201901103

**Requerimientos del Sistema**

En la realización del software el Editor de texto que se utilizó fue Visual Studio Code.

Texto

Descripción generada automáticamente

El programa fue realizado con la versión de Python 3.11.1 para sistemas operativos de 64 bits.



**Analizador Léxico**

Para que el programa funcione correctamente se realizaron diversas validaciones y manejo de errores para que no finalice indebidamente durante la ejecución de esta. Por lo cual explicaremos lo que realiza el código.

Primero se importan las librerías necesarias, las cuales se utilizarán más adelante, por ejemplo: la librería tkinter, la cual servirá para las ventanas graficas

Texto

Descripción generada automáticamente

**Menú Principal**

Para mostrar el menú principal se crea el main, el cual será el que se ejecute inicialmente, y comenzará por mostrar la pantalla principal.

La pantalla se crea con tk.Tk(), con title se le agrega un titulo a la pestaña, con geometry indicamos el tamaño que tendrá dicha ventana indicando el ancho por altura, con configure se indica el color de fondo que tendrá la ventana, resizable sirve para indicar que la ventana sea de tamaño fijo y que el usuario no pueda agrandarla o hacerla más pequeña.

Para crear una barra de menú se utiliza tk.Menu(), el cual sirve para agregarle los datos que sean necesarios, en este caso se crea el espacio para el manejo de los archivos por lo que se utiliza add\_cascade() con sus respectivos parámetros, el primer parámetro indica que se agregará al menú de barra, y el label le indica el nombre que tendrá para esas opciones, para agregar las subopciones se utiliza add\_command() el cual lleva como parámetros los siguientes, label el cual indica el nombre de la opción a utilizar, luego command al cual se le asigna el método al que accederá para realizar la acción que corresponda, luego con state se indica que la opción no estará disponible si no se carga un archivo con anterioridad.

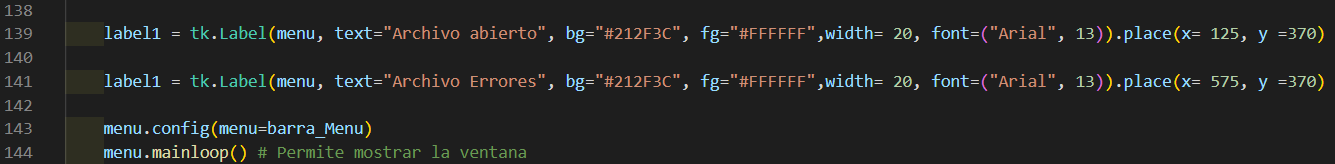
Con tk.Text() se agrega dos áreas de texto que servirán para editar el archivo y mostrar los errores del archivo al momento de realizar las opciones que correspondan a partir que se cargue el archivo en el sistema, con configure() se le indica el color de fondo que tendrá con bg, y state para indicar que al principio no se pueda editar en el área de texto.

Con tk.label() se crea un texto en una línea que estará plasmado para indicar cual cuadro de texto es para el archivo cargado y el que muestra los errores.

Con config() se agrega el parámetro que agrega a la ventana la barra de menú y con mainloop() se indica que que la ventana será visible para el usuario.

Texto

Descripción generada automáticamente



**Método abrir**

Primero se crea la variable almacenar la cual servirá para guardar todo el texto que tiene el archivo a abrir, la variable url guardará la dirección donde se encuentra dicho archivo, y para manejar el analizador realizamos la instancia hacia la clase AFD().

Ahora en el método abrir se utiliza filedialog.askopenfilename() para abrir una ventana que permite acceder a los archivos y seleccionar el que se requiera utilizar en el programa, el cual debe ser con la extensión .json, luego con if se valida si se ha seleccionado algún archivo, si el archivo es cargado entonces abre el archivo con el método open, y con read() lee el contenido del mismo y lo almacena en la variable creada globalmente, además se agregan estos datos al cuadro de texto que se creó inicialmente para poder editar el contenido del mismo, y se habilitan las demás opciones de la barra de menú con entryconfig, y con showinfo indicamos un mensaje para dar a entender al usuario que que el archivo se cargó en el sistema de forma exitosa.

Texto

Descripción generada automáticamente

**Método guardar**

Se crea una variable que tendrá el método open para acceder al archivo que se abrió y poder editar en él, guardando lo que obtenga del área de texto con el método get, y este se escribe con write en el archivo, luego se cierra esté para evitar el seguir usándolo sin ser necesario.

**Texto

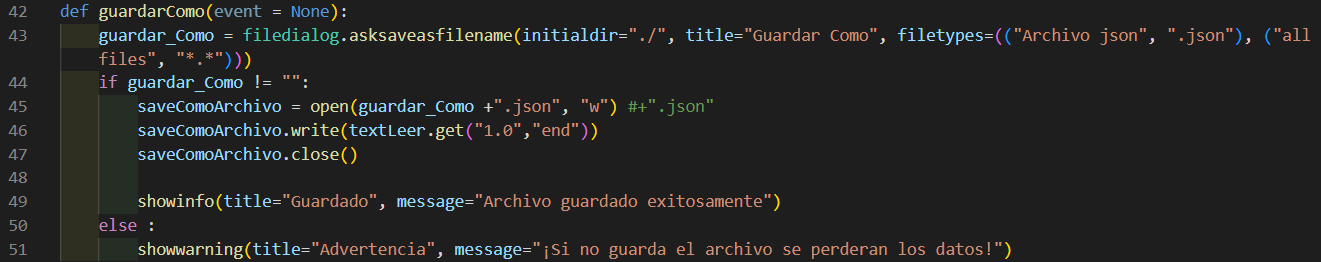
Descripción generada automáticamente**

**Método guardar como**

Se crea el método para permitir guardar lo que se editó en el cuadro de texto con otro nombre y en la ubicación que el usuario requiera, para ello se comienza creando la variable con filedialog.asksaveasfilename() el cual permite abrir la ventana para seleccionar la ruta donde se guardará el archivo y agregarle en nombre al mismo.

Se valida con if si el usuario ha ingresado algúna ruta y agregado un nombre al archivo, si el usuario ingresa correctamente los datos se utiliza open para crear el archivo y permitir escribir en el con el método write() y este obtiene con get los datos que se encuentran en el cuadro de texto y para finalizar se cierra el archivo con close().

Si el usuario cancela la opción de guardar como, entonces se muestra un mensaje de advertencia indicando que puede perder los datos si no los guarda.



**Método analizar**

Inicia validando si existen datos en la variable almacenar, si la variable contiene datos, entonces accede al método analizando() de la clase AFD(), y se envia como parámetro a la variable con los datos, con el método analizandoSintacticamente() se realizan las validaciones correspondientes para hacer los cálculos y escribirlos en un archivo .dot, el cual se escribe con el método write(), luego con os.system convertimos a pdf el contenido del archivo .dot para crear la grafica, y con el método webbrowser.open\_new() se abre automáticamente el archivo pdf para que el usuario pueda visualizarlo correctamente.

**Texto

Descripción generada automáticamente**

**Método errores**

Se valida con if si la variable almacenar tiene datos, si contiene datos, entonces se accede al método erroresValidados() de la clase AFD(), el cual retornará una cadena con el archivo escrito con los datos de los errores encontrados, después de haber analizado el archivo.

Estos datos se muestran en el área de texto con el método insert(), y se desactiva dicha área para que el usuario no pueda editar en él. Luego con open indicamos en donde se guardará el archivo .json con los datos, y se escriben los datos con el método write() y cerramos el archivo con close(). Además si la variable no contiene datos entonces se muestra un mensaje indicando que se cargue un archivo con anterioridad.

**Texto

Descripción generada automáticamente**

**Método manual usuario**

Este método tiene la finalidad de abrir el manual de usuario automáticamente, para que el usuario pueda visualizarlo, para ello se utiliza webbrowser.open\_new() y se envia como parámetro la ruta donde se encuentra el manual pdf.

Texto

Descripción generada automáticamente

**Método manual tecnico**

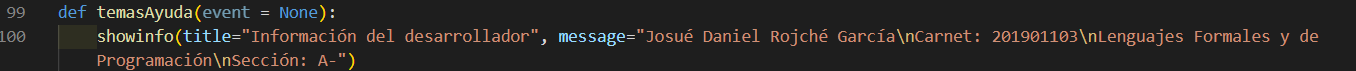
Este método tiene la finalidad de abrir el manual tecnico automáticamente, para que el usuario pueda visualizarlo, para ello se utiliza webbrowser.open\_new() y se envia como parámetro la ruta donde se encuentra el manual pdf.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Método temas ayuda**

Simplemente se muestra una ventana emergente informativa, indicando los datos del desarrollador de la aplicación.



**Clase operar**

**Texto

Descripción generada automáticamente**Se utiliza la librería math para realizar algunos cálculos, el método operando recibe como parámetros valor 1, valor 2 y el tipo de operación a realizar, dentro del método se validan con if que los datos no sean nulos, y si no lo son se verifica que el tipo de operación coincida con la que corresponde y si existe coincidencia se realiza la operación y se retorna el resultado.

**Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

**Clase Token**

En esta clase se ingresan fila, columna y lexema en el constructor, los cuales servirán para agregar en conjunto los datos a la tabla de tokens.

Texto

Descripción generada automáticamente

**Clase AFD**

En el método constructor se inicializa la lista de letras que podrá leer el automata, también la lista de números que podrá utilizarse, se inicializa las variables necesarias a utilizar, como fila, columna, los estados actual, anterior, estados finales, así como la lista para almacenar los tokens, y la lista para almacenar los errores, etc.

Imagen que contiene Aplicación

Descripción generada automáticamente

**Método analizando**

Este método recibe como parámetro el texto que se obtiene del archivo a analizar, el texto se almacena en una variable para manejar los datos localmente, por lo que se utiliza un bucle while para recorrer todos los datos del texto, luego se accede a cada carácter del texto y con un if se valida que se omitan los espacios, y saltos de línea que encuentre en el texto, para luego comenzar con las validaciones de if que indicaran en que estado se encuentra y que carácter puede almacenar en base a el diseño del Autómata Finito Determinista, el cual encuentra el diseño al final de este documento.

Básicamente lo que se realiza es repetitivo, solamente se valida que se encuentre en un estado en especifico y luego cuando ingrese al carácter que coincida, envía el carácter como parámetro al método almacenarToken(), el cual sirve para enviar los datos a la lista de tokens. Luego se indica el estado actual como estado anterior, y el estado actual es el siguiente estado del diseño del AFD, y con else validamos que si el carácter no es admitido por el autómata, entonces se trata de un error, por lo cual es necesario almacenar el carácter en el método almacenarError() y se utiliza una bandera que sirve para indicar que el error se ha encontrado y se quita hasta que encuentre nuevamente la bandera como false, en otro estado.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El código continúa hasta aumentar la columna, eliminar el carácter analizado de la cadena de texto y retornar el estado actual si corresponde al estado de aceptación.

Texto

Descripción generada automáticamente

**Método almacenarToken**

Recibe como parámetro el lexema y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en el automata, luego este se almacena en la tabla de tokens.

Texto

Descripción generada automáticamente con confianza baja

**Método almacenarError**

Recibe como parámetro el lexema y se crea una instancia para almacenar la fila y columna que corresponde al lexema encontrado en el autómata, luego este se almacena en la tabla de errores.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

**Método erroresValidados**

Este método sirve para acceder a la tabla de errores y escribir en una variable la estructura para el archivo de errores .json que requiere generar el usuario en la opción de errores, por lo que simplemente se accede a cada dato y se agrega al espacio que corresponde, luego se elimina la ultima coma y se retorna la cadena.

Texto

Descripción generada automáticamente

**Método analizandoSintacticamente**

Este método sirve para obtener los datos de la tabla de tokens y validar que se realicen las operaciones conforme la jerarquía con la que fue ingresado el archivo, respetando la estructura del mismo, por lo que se utiliza un bucle while para recorrer toda la tabla, sin embargo también se requiere de escribir el archivo .dot con los datos obtenidos de los cálculos, para así mostrarlos en forma gráfica. En las validaciones if podrá observar que se va validando las opciones para las asignaciones de los valores que corresponden a sus determinados campos, sin embargo cuando se llega hasta el if donde se obtiene un valor anidado se comienza a ingresar a un método que puede hacerse recursivo si es necesario validar otras operaciones que sean anidadas, de tal manera que se realicen todas las operaciones de manera ordenada y correctamente. Esto es algo que se repite en todo el método, por lo cual solo se agregan las otras imágenes.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Método operacionAnidada**

Este método básicamente realiza lo mismo que el método anterior, solamente que las validaciones se comienzan desde que encuentra la asignación de la operación para así llevar el orden de la jerarquía de las operaciones a realizar conforme el archivo leído. Y al encontrar el que corresponde automáticamente retorna el resultado al método anterior.

Texto

Descripción generada automáticamente

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

**Autómata Finito Determinista**

En la imagen siguiente se ilustra el autómata finito determinista que fue diseñado para darle solución a la lectura correcta del archivo a analizar en el software.

El AFD fue construido a partir del método del árbol, tomando em cuenta la expresión regular.



Gráfico de dispersión

Descripción generada automáticamente

Diagrama, Esquemático

Descripción generada automáticamenteDiagrama, Esquemático

Descripción generada automáticamente

Diagrama, Esquemático

Descripción generada automáticamente

Diagrama, Esquemático

Descripción generada automáticamente