**Universidad de San Carlos de Guatemala**

**Facultad de Ingeniería**

**Escuela de Ciencias y Sistemas**

**Laboratorio de Lenguajes Formales y de Programación**

**Manual Técnico**

Proyecto 2: Analizador léxico y Sintáctico

**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**

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

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 .txt, 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 showarning indicamos un mensaje para dar a entender al usuario que no cargó ningún archivo en el sistema.

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 alguna 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.

Texto

Descripción generada automáticamente

**Método nuevo**

Se crea el método para permitir crear un nuevo archivo, por lo que se accede a la variable que almacena la url, en el caso de que se haya abierto un archivo anteriormente, con un if se valida si esta variable se encuentra vacía, ya que si está vacía simplemente se borrarán los datos con el método inicializar. Si de lo contrario, la variable no está vacía, entonces se abre un mensaje en pantalla con el método askquestion() indicando que si no guarda los datos serán borrados, si el usuario presiona “No” entonces se abrirá una ventana para guardar el archivo de lo contrario, simplemente se llama al método inicializar().

Texto

Descripción generada automáticamente

**Método generarSentenciasMDB**

Si la variable contiene datos, entonces accede al método analizando() de la clase AFD(), y se envía como parámetro a la variable con los datos, con el método analizadorSintactico() se realizan las validaciones correspondientes para verificar que la estructura de las funciones sea correctas. Luego se crean dos variables booleanas que almacenaran el resultado de evaluar si las listas de errores léxicos o sintácticos está vacía y así poder escribir el archivo con el método escribiendoArchivo(), el cual obtendrá la cadena escrita con las funciones en mongoDB y así poder guardar esos datos con el método open y write, y de lo contrario mostrar un mensaje indicando que existen errores que deben ser corregidos.

**Texto

Descripción generada automáticamente**

**Método verTokens**

Este método tiene la finalidad de abrir una ventana en la cual se mostraran los datos de la tabla de tokens, por lo cual se crea la ventana con Tk.Toplevel para indicar que es secundaria, luego se proporcionan titulo, tamaño, color y se agrega una tabla con Treeview y se proporciona un estilo con style, y una barra para poder visualizar mas datos si la tabla crece aún más, además se agregan las columnas con colum, y los encabezados de los mismos con heading, para luego agregar con tag\_configure un color predeterminado a las filas.

**Texto

Descripción generada automáticamente**

Con for se recorre la tabla de tokens y se agregan los datos a la tabla con el método insert, además la tabla se agrega a la pantalla con pack y con mainloop es visible.

**Pantalla de computadora con letras

Descripción generada automáticamente con confianza media**

**Método verErrores**

Este método tiene la finalidad de abrir una ventana en la cual se mostraran los datos de la tabla de errores léxicos y sintacticos, por lo cual se crea la ventana con Tk.Toplevel para indicar que es secundaria, luego se proporcionan titulo, tamaño, color y se agrega una tabla con Treeview y se proporciona un estilo con style, y una barra para poder visualizar más datos si la tabla crece aún más, además se agregan las columnas con colum, y los encabezados de los mismos con heading, para luego agregar con tag\_configure un color predeterminado a las filas.

**Texto

Descripción generada automáticamente**

Con for se recorre la tabla de errores, tanto léxicos como sintacticos y se agregan los datos a la tabla con el método insert, además la tabla se agrega a la pantalla con pack y con mainloop es visible la ventana.

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Método inicializar**

Este método tiene la finalidad de limpiar los datos del programa y poder comenzar a utilizarlo otra vez sin la necesidad de cerrarlo y volverlo a abrir, por lo que se accede a todas las variables del sistema y se limpian, también se accede al método limpiarDatos para borrar los datos del analizador léxico, y con delete se borran los datos del área de edición de texto.

**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 inicializan las listas que continene las letras para comentarios, identificación, nueva, json, función, y tipo de funcion que podrá leer el autómata, se inicializa las variables necesarias a utilizar, como fila, columna, los estados actual, anterior, estados finales, así como las listas para almacenar los tokens léxicos y sintácticos, y las listas para almacenar los errores de ambos, etc.

Patrón de fondo

Descripción generada automáticamente con confianza media

**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, 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().

Texto

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.

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

**Método almacenarToken**

Recibe como parámetro el lexema y la descripción del token 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

**Método almacenarError**

Recibe como parámetro el lexema y la descripción del token 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.

Texto

Descripción generada automáticamente

**Método analizadorSintactico**

Este método sirve para obtener los datos de la tabla de tokens y validar que se respete la estructura de las funciones y de los comentarios, por lo que se utiliza un bucle while para recorrer toda la tabla. En las validaciones if podrá observar que se va validando las opciones para las asignaciones de los valores que corresponden a sus determinados campos. 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

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

**Método escribiendoArchivo**

Este método básicamente realiza lo mismo que el método anterior, solamente que las validaciones no almacenan nada ya que solamente se requiere que se reconozca los datos para poder convertirlos a sentencias mongoDB, al final retorna la cadena donde se almacenaron las sentencias.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**Método almacenarSintactico**

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

Texto

Descripción generada automáticamente

**Método almacenarErrorSintactico**

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

Texto

Descripción generada automáticamente

**Métodos para obtener las listas**

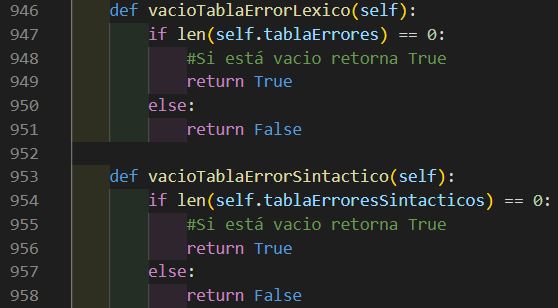
Estos métodos sirven para retornar cada una de las tablas y así poder obtenerlas en las ventanas para recorrerlas y mostrarlas en las tablas.

Texto

Descripción generada automáticamente

**Métodos para validar listas vacías**

Estos métodos sirven para retornar True o False de acuerdo a si la lista se encuentra vacía o no, esto servirá para validar si es posible escribir el archivo con las sentencias en mongoDB, ya que si existen datos en las listas de errores, entonces no se ejecutará el método de escribirArchivo.



**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

**Gramática Libre del Contexto (GLC)**

La GLC fue construida para verificar el orden de las funciones y los comentarios que se obtienen y que se encuentran en una tabla de tokens, la cual fue analizada por el analizador léxico.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene botella, foto, medidor

Descripción generada automáticamente