Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales y de Programación
Catedrático: Inga. Zulma Karina Aguirre Ordoñez
Tutor académico: Douglas Omar Arreola Martínez

SECCION B-

Primer semestre 2022

MANUAL TÉCNICO
Proyecto 2

La Liga Bot

Nombre: Moises David Maldonado de León **Carné:** 202010833

Guatemala, 28 de abril de 2022

Introducción

En el presente documento, se describe la estructura y los demás aspectos técnicos del analizador léxico y sintáctico para la obtención de información y su respectiva visualización de manera óptima y amigable para cierto medio de comunicación de prestigio. Se conocerá no solo el detalle de estructura del programa sino también el proceso previo, es decir, las expresiones regulares utilizadas y el Autómata Finito Determinista (AFD) en cuanto a la parte del análisis léxico y también la gramática libre de contexto (Tipo 2) definida e implementada para el análisis sintáctico.

OBJETIVOS

Generales

Familiarizar al lector con la lógica planteada para el funcionamiento del sistema realizado mediante el uso del lenguaje de programación Python y aplicando un AFD para la parte del análisis léxico y gramática del tipo 2 para la creación del análisis sintáctico.

Específicos

- Mostrar la estructura del programa realizado.
- Dar a conocer con detalle los métodos y variables empleadas, así como su funcionalidad dentro del programa.
- Mostrar la tabla de tokens, expresiones regulares, el método del árbol y el autómata creado para el sistema.
- Mostrar la gramática creada e implementada para la correcta ejecución del programa.

Alcances

El programa se realizó con el fin de optimizar la obtención de información de un archivo CSV con demasiados datos, fue creado con el propósito de crear un entorno agradable donde el usuario puede ingresar ciertos comandos y posteriormente verificar si existen errores en las cadenas de entrada y a su vez, con una sola acción visualizar la información deseada.

Especificaciones

Requisitos del Hardware

- Mínimo 2GB de memoria RAM
- Procesador Intel Core 2 Duo 2 o superior
- Espacio en disco duro: 10 MB

Requisitos del Software

- El programa fue realizado en Python, específicamente con la versión 3.10.4
- El IDE utilizado fue Visual Studio Code, versión 1.65.2
- Funcional en sistemas operativos como Windows XP, 7,8,10 o MAC OS
- Tener instaladas o con acceso a las siguientes librerías y paquetes: tkinter, webbrowser y pandas.

Lógica del programa

Descripción de Clases	
Clase	**Descripción**
lista()	Clase que maneja la inserción de tokens y errores léxicos encontrados durante el análisis. También contiene los métodos para mostrar o eliminar datos y las funciones que generan los reportes y limpiar datos.
Token1	Clase que contiene un constructor para guardar los objetos de tipo error. Posee métodos para devolver la información en un arreglo y métodos *"get"* para obtener el valor individual de la clase.
error	Clase que contiene un constructor para guardar los objetos de tipo token. Posee métodos para devolver la información en un arreglo y métodos *"get"* para obtener el valor individual de la clase.
Sintactico()	Clase que recibe la lista de tokens y agrega errores sintácticos encontrados durante el segundo análisis. También contiene los métodos para las búsquedas y las funciones de la gramática de tipo 2.

Archivo main

Es el archivo llamado *“main.py”*. Es el archivo principal y el cual debe ejecutarse para que el sistema se inicie. En principio contiene, además de las librerías y paquetes a necesitar, las funciones principales como mostrar datos, ver reportes y el analizador léxico, que a su vez inicia el analizador sintáctico.

Seguido de eso se encuentra toda la creación de la interfaz gráfica mediante el uso de la librería tkinter.

```python
import os as documento
from listas import lista
import tkinter as tk
from tkinter import *
from tkinter import ttk
from tkinter import scrolledtext as stxt
from tkinter import filedialog
from tkinter import messagebox as MessageBox
import webbrowser
import pandas as pd

data = lista()

def obtenerCSV(): ...

def mostrarDatos(): ...

def respuestaDatos(respuesta): ...

def analizarTexto():  #Analizador léxico ...

def verificaPalabrasReservadas(lexema,fila,columna): #Método que verifica las palabras reservadas ...

def verManualUsuario(): ...

def verManualTecnico(): ...

def ayuda(): ...

#-----------------------------Interfaz gráfica-----------------------------------

#Raiz
root = tk.Tk()
#Configuración Raiz
root.title('La Liga Bot')
root.geometry('1050x675')
root.resizable(0,0)
root.config(bg="#87F7EC")
root.eval('tk::PlaceWindow . center')

#Botón 1
# botonCargar = tk.Button(text="Cargar archivo", command=abrirDocumentoForm)
# botonCargar.place(x=25, y=20)
# botonCargar.config(font=("Courier", 12), bg="#0A1246",fg="white",width=15)

#--------------------------- Botones menú ----------------------------------------
#Botón 1
botonReporteErrores = tk.Button(text="Reporte de errores", command=data.reporteErrores)
botonReporteErrores.place(x=780, y=25)
botonReporteErrores.config(font=("Courier", 12), bg="#0A1246",fg="white",width=23)

#Botón 1
botonLimpiarErrores = tk.Button(text="Limpiar log de errores",command=data.limpiarLogErrores)
botonLimpiarErrores.place(x=780, y=65)
botonLimpiarErrores.config(font=("Courier", 12), bg="#0A1246",fg="white",width=23)
```

Descripción de las funciones		
Clase	**Método**	**Descripción**
Lista()	**__init__**	Se inicia al ser llamada la función y contiene los arreglos donde se irán guardando los tokens y errores.
	insertarToken(self,token,valor,fila,columna)	Método para la inserción de un token en el arreglo. Se deben de enviarle los parámetros solicitados.
	insertError(self, caracter,tipo,fila,columna)	Método para la inserción de un error en el arreglo. Se deben de enviarle los parámetros solicitados.
	def mostrarTokens(self) **def mostrarErrores(self)**	Métodos para mostrar los tokens y errores en la consola. Estos métodos son secundarios y es para pruebas de verificación de datos.
	def limpiarLogErrores(self)	Este método elimina la información en la lista de errores.
	def reporteTokens(self)	Función que recorre la lista de tokens y genera una tabla en un archivo HTML
	def reporteErrores(self)	Función que recorre la lista de errores y genera una tabla en un archivo HTML
	def limpiarLogTokens(self)	El segundo método, elimina la información de la lista de tokens.

Descripción de las funciones		
Clase	**Método**	**Descripción**
Token1	**__init__**	Se inicia al ser llamada la función y contiene los atributos de los tokens: tipo de token, valor, fila y columna
	def enviarTokens(self)	Función que retorna los atributos del constructor en forma de arreglo.
	def getToken(self) **def getValor(self)** **def getFila(self)** **def getColumna(self)**	Funciones que retornan el valor de cada atributo respectivamente como una variable String.

Descripción de las funciones		
Clase	**Método**	**Descripción**
error	**__init__**	Se inicia al ser llamada la función y contiene los atributos de los errores: caracter, tipo de error, fila y columna
	def enviarErrores(self)	Función que retorna los atributos del constructor en forma de arreglo.
	def getCaracter(self) **def getTipo(self)** **def getFila(self)** **def getColumna(self)**	Funciones que retornan el valor de cada atributo respectivamente como una variable String.

Descripción de las funciones		
Archivo	**Método**	**Descripción**
Main.py	**def mostrarDatos()** **def respuestaDatos()**	La primera función inserta el texto en el ScrolledText con una etiqueta de alineación hacia la derecha. La segunda función cumple el mismo objetivo, pero con alineación a la derecha y es la respuesta del bot.
	def verManualUsuario() **def verManualTecnico()**	Estas funciones despliegan los respectivos manuales al ser llamadas en los botones respectivos.
	def analizarTexto()	Función que maneja el autómata. Al inicio se manda a llamar el método de mostrarErrores() y la función mostrarTokens(). Contiene toda la estructura de análisis que se mostrará más adelante con el método del árbol. También inicia seguido de eso el análisis sintáctico.
	def verificaPalabrasReservadas()	Función que verifica si un lexema genera una palabra reservada, en caso contrario lo designa como un ID.
	def ayuda()	Genera una ventana con el texto de ayuda de comandos para que el usuario tenga la información a la mano.

Descripción de las funciones		
Clase	**Método**	**Descripción**
Sintactico()	**def __init__**	Función de inicialización, genera listas temporales de tokens y errores (hereda copia de lista de tokens y errores del análisis léxico).
	def insert(lista)	Función que se llama para asignar la copia de lista de tokens a esta clase.
	def insertErrorS()	Función que inserta los errores sintácticos a la lista de errores misma del análisis léxico.
	def quitarToken()	Método que emplea pop para sacar el primer elemento de la “pila”.
	def getToken()	Método que obtiene el primer elemento de la “pila” para observarlo.
	def inicio()	Función de producción inicial. Llama el método de getToken para decidir que función llamar.
	def resultadoPartido()	Función de producción resultado_partido, si se ejecuta correctamente llama la función resultadoCSV()
	def jornada()	Función de producción Jornada, si se ejecuta correctamente llama la función jornadaCSV()
	def asignarNombre()	Función de producción asignar_nombre, si se ejecuta correctamente retorna el nombre.
	def goles()	Función de producción goles, si se ejecuta correctamente llama la función golesCSV()
	def condicion()	Función de producción condicion, si se ejecuta correctamente retorna el 1 de 3 valores de condición.
	def t_Temporada()	Función de producción T_Temporada, si se ejecuta correctamente llama la función tablaTempCSV()
	def partidos()	Función de producción partidos, si se ejecuta correctamente llama la función tempEquipoCSV()
	def rango1()	Función de producción rango1, si se ejecuta correctamente llama la función resultadoCSV()
	def rango2()	Función de producción rango2, si se ejecuta correctamente retorna un token o cadena vacía
	def top()	Función de producción top, si se ejecuta correctamente llama la función topCSV()
	def condicion2()	Función de producción condicion2, si se ejecuta correctamente retorna 1 de 2 valores de condición.
	def bandera()	Función de producción bandera, si se ejecuta correctamente retorna un token o cadena vacía.
	def adios()	Función de producción adios, si se ejecuta correctamente retorna un mensaje y termina la ejecución.
	def resultadoCSV()	Retorna el resultado de un partido en específico
	def jornadaCSV()	Genera un HTML con los resultados de una jornada y temporada específica.
	def golesCSV()	Retorna los goles anotados por un equipo.
	def tablaTmp()	Genera un HTML con los equipos y puntos obtenidos en una temporada específica.
	def tempEquipo()	Genera un HTML con los partidos disputados por un equipo en una temporada específica.
	def topCSV()	Retorna el top de mejores o peores equipos clasificados.

Flujo del programa

Análisis Léxico

Terminales

Para el lenguaje utilizado se identificaron los siguientes terminales:

Terminales
1. Identificadores
2.Palabras reservadas
3. Símbolos
4. Cadenas entre comillas dobles
5. Dígitos

Tabla de tokens

Se describen los conjuntos utilizados: letras minúsculas del alfabeto (L), números enteros entre 0 y 9 (D) y finalmente los símbolos reconocidos en el lenguaje (S).

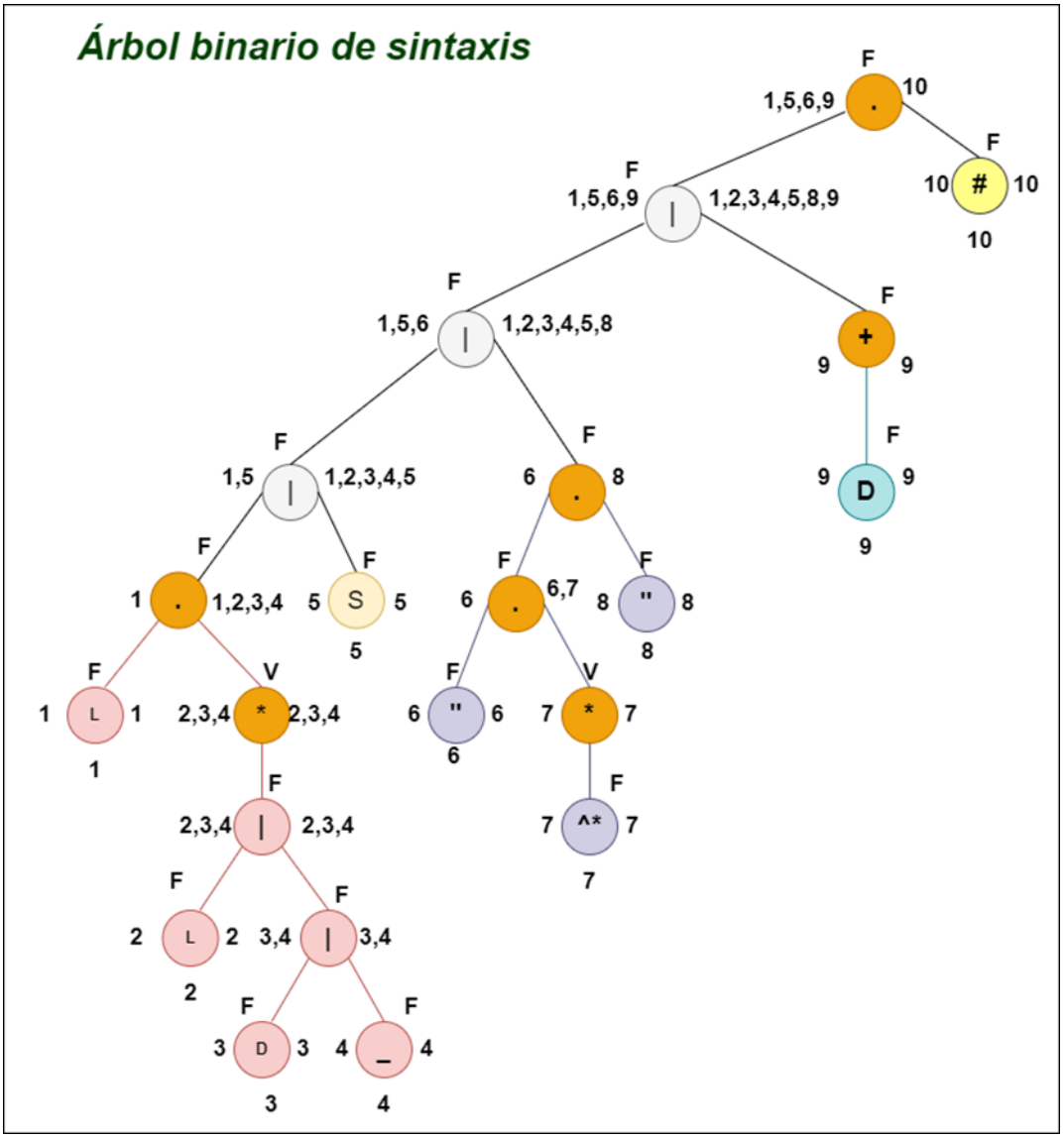
L = [a-z,A-Z,Ñ,ñ] D = [0-9] S=[' < ', ' > ', ' - ']

Token	Patrón	ER	Lexema
Identificador	Una letra, que puede ser seguida de una combinación de más letras o números o un guión bajo sin orden o cantidad específica.	L(L|D|'_')*	**reporteEspanol** **reporteGlobal1** **archivo_1**
Símbolo	Un símbolo de "<" o ">" o " - "	S	**<** **-** **>**
Cadena doble	Inicia con una comilla doble seguida de una secuencia de cualquier cosa y termina con una comilla doble	(“) (^”)* (”)	**“Valencia”** **“Zaragoza”**
Dígito	Número compuesto por uno o más dígitos	D+	**22** **2018**

Expresión Regular: (L(L|D|'_')*) | S | ((“) (^”)* (”)) |D+

Método del árbol

ER	((L(L|D|'_')*) | S | ((“) (^”)* (”)) |D+) #



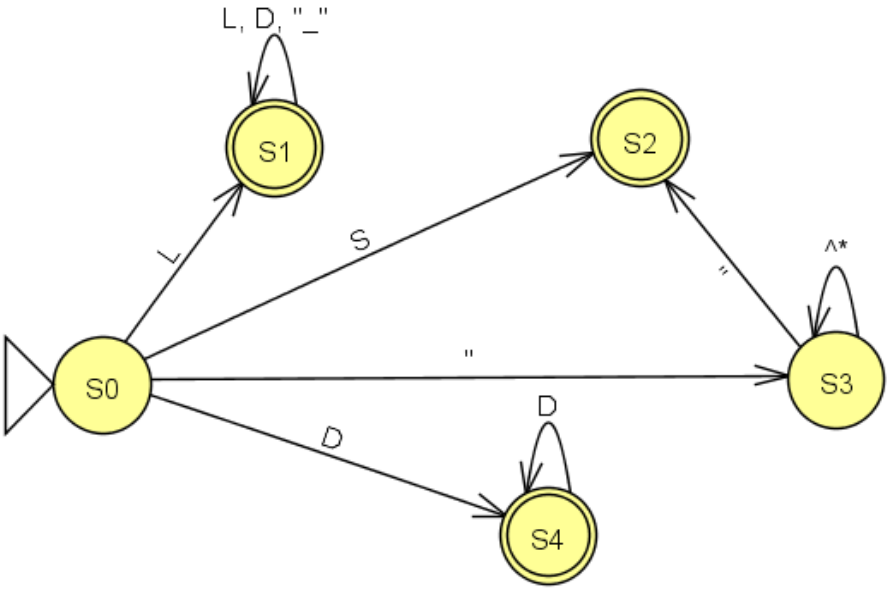
Valor	Hoja	Siguientes
L	1	2,3,4,10
L	2	2,3,4,10
D	3	2,3,4,10
_	4	2,3,4,10
S	5	10
"	6	7,8
^*	7	7,8
"	8	10
D	9	9,10
#	10	----------

Tabla de transiciones

	Estado	Valores	Siguientes
Inicio	**S0**	L, S, ", D 1,5,6,9	L:{2,3,4,10} = **S1** S:{10} = **S2** ":{7,8} = **S3** D:{9,10} = **S4**
Aceptación	**S1**	L, D, _ , # 2,3,4,10	L:{2,3,4,10} = **S1** D:{2,3,4,10} = **S1** _:{2,3,4,10} = **S1**
Aceptación	**S2**	# 10	----------
	S3	^*, " 7,8	^*:{7,8} = **S3** ":{10} = **S2**
Aceptación	**S4**	D, # 9,10	D:{9,10} = **S4**

Método del árbol – Construcción del AFD

	Estados	Σ					
		L	S	"	D	^*	_
Inicio	S0	S1	S2	S3	S4	-	-
#	S1	S1	-	-	S1	-	S1
#	S2	-	-	-	-	-	-
	S3	-	-	S2	-	S3	-
#	S4	-	-	-	S4	-	-



Análisis Sintáctico (Gramática libre del contexto utilizada)

Terminales = {tk_resultado, tk_cadena, tk_vs, tk_temporada, tk_Smenor, tk_año, tk_guion, tk_Smayor, tk_jornada, tk_num tk_f, tk_goles, tk_local, tk_visitante, tk_total, tk_tabla, tk_partidos, tk_ji, tk_jf, tk_top, tk_superior, tk_inferior, tk_n, tk_adios, tk_id}

No terminales = {<INICIO>, <RESULTADO_PARTIDO>, <JORNADA>, <GOLES>, <T_TEMPORADA>, <PARTIDOS>, <TOP>, <ADIOS>, <ASIGNAR_NOMBRE>, <CONDICION>, <RANGO1>, <RANGO2>,<CONDICION2>, <BANDERA>}

Inicio = <INICIO>

Producciones:

<INICIO> ::= <RESULTADO_PARTIDO> | <JORNADA> | <GOLES> | <T_TEMPORADA> | <PARTIDOS> | <TOP> | <ADIOS>

<RESULTADO_PARTIDO> ::= tk_resultado tk_cadena tk_vs tk_cadena tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor

<JORNADA>::= tk_jornada tk_num tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor <ASIGNAR_NOMBRE>

<ASIGNAR_NOMBRE>::= tk_f tk_id | épsilon

<GOLES> ::= tk_goles <CONDICION> tk_cadena tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor

<CONDICION>::= tk_local | tk_visitante | tk_total

<T_TEMPORADA> ::= tk_tabla tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor <ASIGNAR_NOMBRE>

<PARTIDOS> ::= tk_partidos tk_cadena tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor <ASIGNAR_NOMBRE> <RANGO1> <RANGO2>

<RANGO1> ::= tk_ji tk_num | épsilon

<RANGO2> ::= tk_jf tk_num | épsilon

<TOP> ::= tk_top <CONDICION2> tk_temporada tk_Smenor tk_año tk_guion tk_año tk_Smayor

<CONDICION2>::= tk_superior | tk_inferior <BANDERA>

<BANDERA> ::= tk_n tk_num | épsilon

<ADIOS> ::= tk_adios