Universidad de San Carlos de Guatemala Facultad de ingeniería Ingeniería en Ciencias y Sistemas Lenguajes Formales y de Programación



MANUAL TECNICO

Leonel Antonio González García 201709088

Guatemala 20 de marzo de 2022

MANUAL TECNICO

Siempre al inicio del programa encontraremos las librerías que nos ayudaran con la creación de las funciones que se necesitan.

La librería tkinter es la que nos permite trabajar la interfaz gráfica.

```
from tkinter import Tk

from tkinter import *

from tkinter import ttk

from tkinter import filedialog

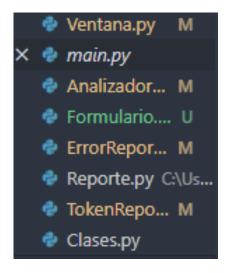
from Analizador_lexico import Analizador_lexico

from ErrorReporte import reporterror

from TokenReporte import reportoken

import webbrowser
```

Para el desarrollo óptimo de este programa se recurrió al uso de clases.



Clase Ventana

En esta clase están los métodos que sirven para crear la interfaz gráfica del programa y el funcionamiento de está.

```
def ventana carga():
   archivo = filedialog.askopenfilename(initialdir = "/")
   #Abre el achivo
   archivo = open(archivo ,'r')
   read = archivo.read()
   archivo.close()
   global Data
   Data = read
   Data = textfield.insert(INSERT, Data)
   Data = textfield.get("1.0", "end-1c")
   print("-----")
   print(Data)
def Analizando():
   global Data
   Data = textfield.get("1.0", "end-1c")
   print(Data)
   entry = Analizador lexico()
   entry.analizador(Data)
   f = open ("Analizador.txt", "w")
   f.write(Data)
   f.close()
   entry.Formulario()
```

```
ventana = Tk()
ventana.title("PROYECTO")
ventana.geometry("%dx%d+%d+%d" % (900,500,350,100))
ventana.resizable(0,0)
panel Frame = Frame(ventana)
panel_Frame.pack(side = "top")
panel_Frame.place(width = "900", height = "500")
panel_Frame.config(background = "gray")
textfield = Text(panel_Frame)
textfield.pack()
textfield.place(x= "165", y = "20",width = "720", height = "470")
scrollbar = Scrollbar(ventana, command = textfield.yview)
scrollbar.pack()
scrollbar.place(x = "880", y = "20", width = "20", height = "470")
textfield.config(yscrollcommand = scrollbar)
button = Button(panel_Frame, text="Cargar", command = ventana_carga, foreground = "white")
button.pack()
button.config(bg = "black")
button.place(x = 10, y = 20, width = 150, height = 40)
```

.

Clase Analizador Léxico

Esta clase sirve para crear el analizador léxico, que nos sirve para crear el formulario y los reportes al analizar el archivo.

```
class Analizador_lexico:
   def __init__(self):
    self.listTokens = []
       self.listError = []
       self.listReservadas = []
       self.listValores = []
   def analizador(self,entry):
       self.listTokens = []
       self.listError = []
       self.listReservadas = []
       buffer = ""
       centinela = "$"
       entry += centinela
       linea = 1
       columna = 1
       estado = 0
       index = 0
       while index < len(entry):
           caracter = entry[index]
```

```
if estado == 0:
    #Reconoce Signos
    if caracter == "~":
        #se le suma uno a la columna
        columna += 1
        #se agrega el caracter al buffer
        buffer += caracter
        #se crea y agrega el token a la lista de tokens
        token = Token("VIRGULILLA",buffer,linea,columna)
        self.listTokens.append(token)
        #se limpia el buffer
        buffer = ""
        #se cambia de estado
        estado = 0

elif caracter == "<":
        columna += 1
        buffer += caracter
        token = Token("MENOR QUE", buffer,linea,columna)
        self.listTokens.append(token)
        buffer = ""
        estado = 0</pre>
```

Clase Formulario

Aquí se crea el HTML que contendrá el formulario y sus componentes.

Clase ErrorReporte y TokenReporte

En estás clases se crean los reportes de token y errores que se abren al analizar el archivo.

Clase Token

En está clase se define los atributos del constructor del token y sus métodos set y get.

```
class Token:
    def __init__(self,tipo,lexema,linea,columna):
       self.tipo = tipo
       self.lexema = lexema
        self.linea = linea
        self.columna = columna
    def getInfo(self):
       print("\n ======"")
       print("Tipo: ", self.tipo)
        print("Lexema: ", self.lexema)
        print("Linea: ", self.linea)
       print("Columna: ", self.columna)
    def getTipo(self):
        return self.tipo
    def getLexema(self):
        return self.lexema
    def getLinea (self):
       return self.linea
    def getColumna (self):
        return self.columna
```

Clase Error

En está clase se encuentra el constructor de los errores.

```
class Error:

def __init__(self,tipo,lexemd,linea,columna):
    self.tipo = tipo
    self.lexema = lexema
    self.linea = linea
    self.columna = columna

#función que permite obtener la información de los errores

def getError(self):
    print("\n ============")
    print("Tipo: ", self.tipo)
    print("Lexema: ", self.lexema)
    print("Linea: ", self.linea)
    print("Columna: ", self.columna)
```

Fin del Programa.

Expresiones Regulares

Signos

Tipo	Carácter	Token
Virgulilla	Carácter ~	~
Corchete que Abre	Carácter []
Corchete que	Carácter]]
Cierra		
Dos puntos	Carácter :	:
Menor que	Carácter <	<
Mayor que	Carácter >	>
Comilla Doble	Carácter ""	6633
Coma	Carácter ,	,

Palabras Reservadas

Tipo	Palabra Reservada	Lexema
Formulario	formulario	formulario
Тіро	tipo	tipo
Valor	valor	valor
Fondo	fondo	fondo
Nombre	nombre	nombre
Valores	valores	valores
Evento	evento	evento
Identificador	Inicia con comillas dobles y finaliza al cerrar comillas dobles	"etiqueta"