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 01 de abril de 2024

MANUAL TECNICO

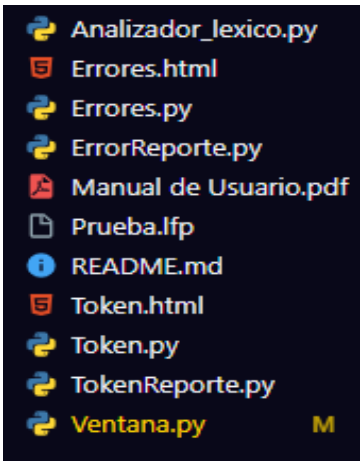
Con el uso de los paradigmas de programación orientado a objetos se desarrolló el programa solicitado por la empresa para el manejo de la información de instrucciones que son leídos al cargar un archivo, haciendo uso de las clases y los métodos para que el programa cumpla con su funcionamiento de manera eficiente.

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

La librería tkinter nos sirve para la creación de la interfaz gráfica del sistema.

```
from tkinter import ttk, filedialog, messagebox
import tkinter as tk
from tkinter.scrolledtext import ScrolledText
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.



Ventana.py

En esta clase están los métodos que ayudan a la creación de la interfaz gráfica de la aplicación y el funcionamiento, con métodos como el de abrir_archivo() o Analizando() que ayudan con la lectura del archivo para que funciones de manera correcta.

```
def abrir_archivo(self):
    archivo = filedialog.askopenfilename()
    if archivo:
        with open(archivo, 'r') as file:
            contenido = file.read()
            self.barra_scroll.delete(1.0, tk.END)
            global Data
            Data = contenido
            Data = self.barra_scroll.insert(tk.INSERT, Data)
            Data = self.barra_scroll.get("1.0","end-1c")
            print("================================================")
            print(Data)
        self.columna_numeros()
```

```
def Analizando(self):
    global Data
    Data = self.barra_scroll.get("1.0","end-1c")
    print(Data)
    entry = Analizador_lexico()
    entry.analizador(Data)
    f = open ("Analizador.lfp","w")
    f.write(Data)
    f.close()
```

Token.py

En esta clase se crea el método constructor del token y la función que ayuda para manejar la información del token.

```
#Clase que maneja los token
class Token:

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

    #Función que permite obtener la información del Token
    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
```

TokenReporte.py

En está clase creamos un método para crear el archivo HTML que sirve para mostrar el listado de tokens del archivo analizado.

```
import webbrowser

def reportoken(listaToken = []):
  info = " "
  htmlFile = open("Token.html", "w")

  htmlFile.write("""<!DOCTYPE HTML PUBLIC"

    <html>

    <head>
        <title>REPORTE </title>
     <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="
    </head>
    <body>
    <div class="container">
  <h2 style = "background-color:#6c757d"> REPORTE DE TOKENS </h2>
```

Errores.py

Contiene el método constructor y el método que sirve para obtener todos los errores léxicos que contiene el archivo analizado y poder manejar está información para crear el reporte.

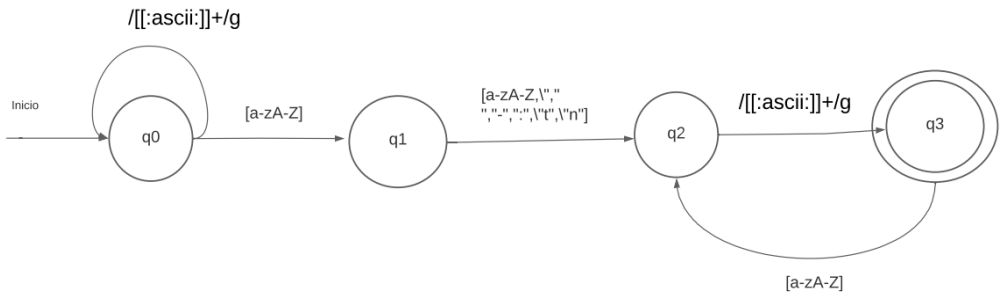
```
#clase para manejar los errores
class Error:

    def __init__(self,tipo,lexema,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)
```

Analizador_Lexico.py

En este método se crea el analizador léxico que sirve para encontrar los errores que se encuentren en el archivo mientras se analiza.

```python
from Token import Token
from Errores import Error
from ErrorReporte import reporterror
from TokenReporte import reportoken
import re



class Analizador_lexico:


    def __init__(self):
        self.listTokens = []
        self.listError = []
        self.listReservadas = []
        self.listValores = []

    def analizador(self,entry):
        #Reiniciar listas para que en c
        self.listTokens = []
        self.listError = []
        self.listReservadas = []

        buffer = ""
        centinela = "$"
        entry += centinela

        linea = 1
        columna = 1

        estado = 0

        index = 0
```

Autómata Finito Determinista



Fin del Programa