Universidad Autónoma de Chiapas

Licenciatura en Ingeniería en Desarrollo y Tecnología de Software, Campus 1

Tuxtla Gutiérrez Chiapas, 19 de Febrero del 2024

Nombre del Docente:

GUTIERREZ ALFARO LUIS, DR

Nombre de la Materia:

COMPILADORES

Nombre del Alumno:

Clemente López Jasson Jared

Semestre y Grupo:

6-M

Nombre de la actividad:

Act. 1.5 Práctica . Unidad 1. Realiza un analizador Léxico en python for void burbuja arreglo

```python
import tkinter as tk
from tkinter import ttk
import ply.lex as lex

def analizar(event=None):
    datos = entry1.get("1.0", tk.END)
    lineas = datos.split('\n')  # Dividir los datos en líneas
    resultado_texto.config(state=tk.NORMAL)
    palabras_mostradas = set()  # Conjunto para almacenar las palabras ya mostradas
    for num_linea, linea in enumerate(lineas, start=1):  # Iterar sobre cada línea con su número correspondiente
        ci = ingreso(linea)  # Analizar la línea actual
        for estado in ci:  # Mostrar cada estado analizado de la línea
            palabra = estado.split()[3]  # Extraer la palabra (cuarta palabra en cada estado)
            if palabra == "lenght":
                continue
            if palabra in ["arreglo", ";", "int", "i"] and palabra in palabras_mostradas:
                continue  # Si ya se mostró esta palabra, omítela
            palabras_mostradas.add(palabra)  # Agrega la palabra al conjunto de palabras mostradas
            resultado_texto.insert(tk.END, f"Línea--> {num_linea}, {estado}\n")
    resultado_texto.config(state=tk.DISABLED)

def eliminar():
    resultado_texto.config(state=tk.NORMAL)
    resultado_texto.delete('1.0', tk.END)
    resultado_texto.config(state=tk.DISABLED)

tokens = ['ID', 'STATIC', 'VOID', 'BURBUJA', 'ARREGLO', 'ITERADOR', 'IZQPARENT', 'DERPARENT', 'IZQCORCHET', 'DERCORCHE

palabras_reservadas = {
    'static': 'STATIC',
    'void': 'VOID',
```

```
    'burbuja': 'BURBUJA',
    'arreglo':'ARREGLO',
    'i': 'ITERADOR',
    'for':'FOR',
    '(': 'IZQPARENT',
    ')': 'DERPARENT',
    '{': 'IZQCORCHET',
    '}': 'DERCORCHET',
    '[': 'LLAVEIZQ',
    ']':'LLAVEDER',
    '++':'INCREMENT',
    'int': 'INT',
    '=': 'IGUAL',
    ';': 'PUNTOCOMA',
    '.':'UNION'
}

def t_ID(t):
    r'[a-zA-Z_][a-zA-Z0-9_]*'
    t.type = palabras_reservadas.get(t.value, 'ID')
    return t

#regex tokens
t_INT = r'int'
t_IGUAL = r'='
t_MAS= r'\+'
t_MENOS = r'\-'
t_PUNTOCOMA = r';'
t_INCREMENT = r'\++'
t_MENORQUE = r'\<'
t_MAYORQUE = r'\>'
```

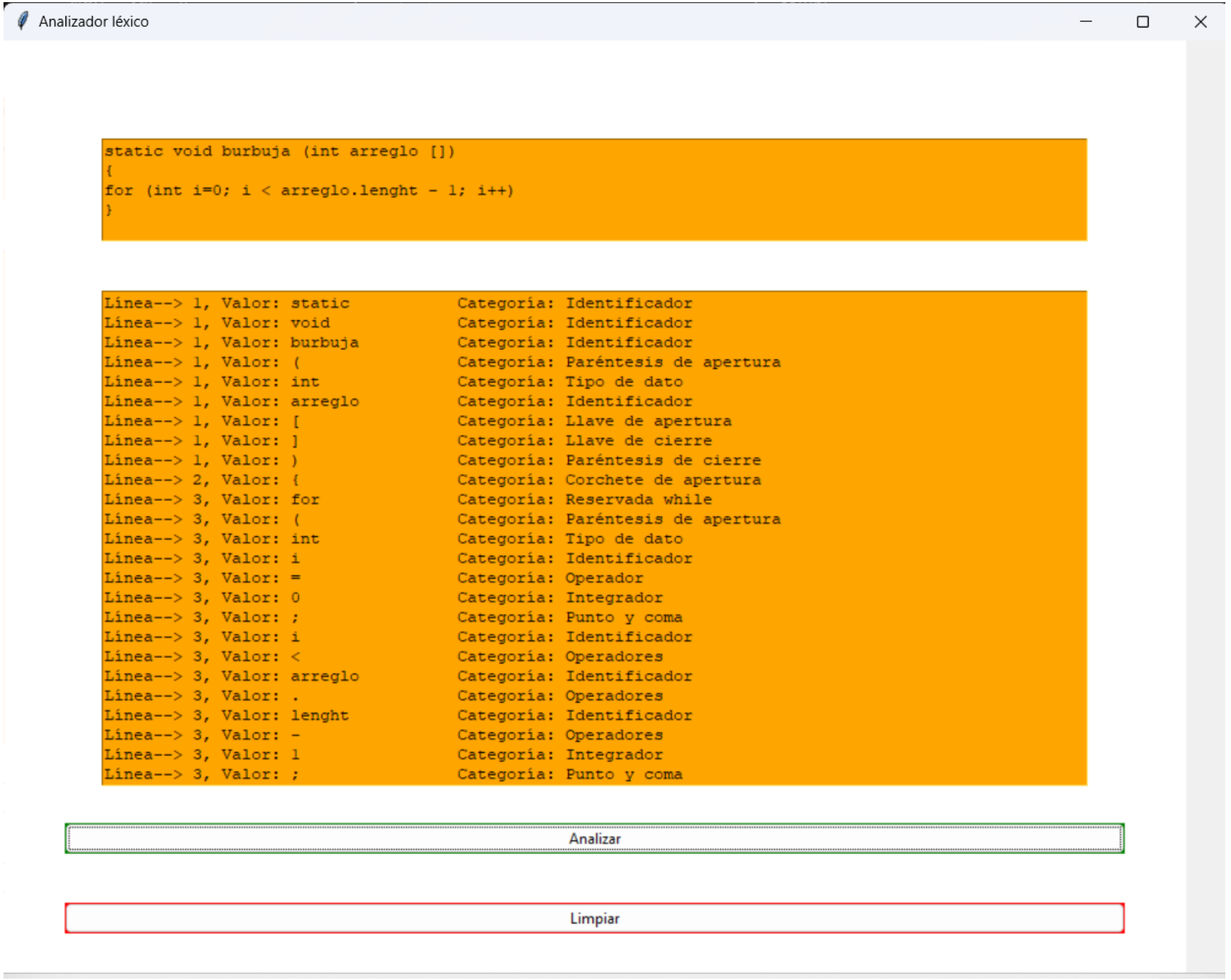
```
62  t_MAYORQUE = r'\>'
63  t_DIGIT = r'\d+'
64  t_UNION = r'\.'
65  #paréntesis
66  t_IZQPARENT = r'\('
67  t_DERPARENT = r'\)'
68  #corchete
69  t_IZQCORCHET = r'\{'
70  t_DERCORCHET = r'\}'
71  #llaves
72  t_LLAVEIZQ = r'\['
73  t_LLAVEDER = r'\]'
74  #ignore espacios en blanco
75  def t_WHITESPACE(t):
76      r'[\t]+'
77      pass
78  t_ignore = ' \t'
79
80  def t_error(t):
81      #print(f"Carácter ilegal: {t.value[0]} en la línea {t.lineno}")
82      t.lexer.skip(1)
83
84  def error(datos):
85      return [f'no definido: {datos}']
86
87  def ingreso(datos):
88
89      if len(datos) < 1:
90          return ['Cadena inválida: La cadena está vacía.']
91
92      token = lex.lex()  # Definir el objeto token aquí
```

```
    token.input(datos)
    lexer = []
    es_valido = True

    for toke in token:
        if toke.type == 'ID':
            es_valido = es_valido and (toke.value in palabras_reservadas or toke.value.isdigit())
            categoria = 'Identificador'
        elif toke.type in ['STATIC', 'VOID', 'BURBUJA', 'ARREGLO', 'ITERADOR']:
            categoria = 'Identificador'
        elif toke.type in ['MENORQUE', 'MAYORQUE', 'MAS', 'MENOS', 'UNION']:
            categoria = 'Operadores'
        elif toke.type == 'IZQPARENT':
            categoria = 'Paréntesis de apertura'
        elif toke.type == 'DERPARENT':
            categoria = 'Paréntesis de cierre'
        elif toke.type == 'IZQCORCHET':
            categoria = 'Corchete de apertura'
        elif toke.type == 'DERCORCHET':
            categoria = 'Corchete de cierre'
        elif toke.type == 'LLAVEIZQ':
            categoria = 'Llave de apertura'
        elif toke.type == 'LLAVEDER':
            categoria = 'Llave de cierre'
        elif toke.type == 'DIGIT':
            categoria = 'Integrador'
        else:
            es_valido = es_valido and toke.type in palabras_reservadas.values()
            if toke.type == 'FOR':
                categoria = 'Reservada while'
```

```
122                 categoria = 'Reservada while'
123             elif toke.type == 'INT':
124                 categoria = 'Tipo de dato'
125             elif toke.type == 'IGUAL':
126                 categoria = 'Operador'
127             elif toke.type == 'PUNTOCOMA':
128                 categoria = 'Punto y coma'
129             elif toke.type == 'INCREMENT':
130                 categoria = 'Operador Incremento'
131             else:
132                 categoria = 'No clasificado'
133
134         estado = "Valor: {:16} Categoría: {:16}".format(
135             str(toke.value), categoria)
136         lexer.append(estado)
137
138     return lexer
139
140 # Interfaz gráfica
141 ventana = tk.Tk()
142 ventana.title("Analizador léxico")
143 ventana.geometry("1000x800")
144
145 # Estilos para los widgets
146 ventana.style = ttk.Style()
147 ventana.style.configure('Green.TButton', background='green')
148 ventana.style.configure('Red.TButton', background='red')
149 ventana.style.configure('Frame.TFrame', background='white')  # Establecer color de fondo para el marco
150
151 frame = ttk.Frame(ventana, padding=(50, 50, 50, 50), style='Frame.TFrame')
```

```
151   frame = ttk.Frame(ventana, padding=(50, 50, 50, 50), style='Frame.TFrame')
152   frame.grid(column=0, row=0, sticky=(tk.W, tk.E, tk.N, tk.S))
153
154   entry1 = tk.Text(frame, height=5, width=100)
155   entry1.grid(column=0, row=0, padx=30, pady=30)
156   entry1.configure(bg='orange')
157   resultado_texto = tk.Text(frame, height=25, width=100, state=tk.DISABLED)
158   resultado_texto.grid(column=0, row=1, padx=10, pady=10)
159   resultado_texto.configure(bg='orange')
160
161   boton_analizar = ttk.Button(frame, text="Analizar", command=analizar, style='Green.TButton')
162   boton_analizar.grid(column=0, row=2, pady=20, sticky=tk.N+tk.S+tk.W+tk.E)
163
164   boton_limpiar = ttk.Button(frame, text="Limpiar", command=eliminar, style='Red.TButton')
165   boton_limpiar.grid(column=0, row=3, pady=20, sticky=tk.N+tk.S+tk.W+tk.E)
166
167   ventana.mainloop()
168
```


Analizador léxico
static void burbuja (int arreglo [])
{
for (int i=0; i < arreglo.lenght - 1; i++)
}
Línea--> 1, Valor: static Categoría: Identificador
Línea--> 1, Valor: void Categoría: Identificador
Línea--> 1, Valor: burbuja Categoría: Identificador
Línea--> 1, Valor: (Categoría: Paréntesis de apertura
Línea--> 1, Valor: int Categoría: Tipo de dato
Línea--> 1, Valor: arreglo Categoría: Identificador
Línea--> 1, Valor: [Categoría: Llave de apertura
Línea--> 1, Valor:] Categoría: Llave de cierre
Línea--> 1, Valor:) Categoría: Paréntesis de cierre
Línea--> 2, Valor: { Categoría: Corchete de apertura
Línea--> 3, Valor: for Categoría: Reservada while
Línea--> 3, Valor: (Categoría: Paréntesis de apertura
Línea--> 3, Valor: int Categoría: Tipo de dato
Línea--> 3, Valor: i Categoría: Identificador
Línea--> 3, Valor: = Categoría: Operador
Línea--> 3, Valor: 0 Categoría: Integrador
Línea--> 3, Valor: ; Categoría: Punto y coma
Línea--> 3, Valor: i Categoría: Identificador
Línea--> 3, Valor: < Categoría: Operadores
Línea--> 3, Valor: arreglo Categoría: Identificador
Línea--> 3, Valor: . Categoría: Operadores
Línea--> 3, Valor: lenght Categoría: Identificador
Línea--> 3, Valor: - Categoría: Operadores
Línea--> 3, Valor: 1 Categoría: Integrador
Línea--> 3, Valor: ; Categoría: Punto y coma
Analizar
Limpiar

https://github.com/Jasson2003/Act.-1.5-Pr-ctica-.git