UNIVERSIDAD AUTONOMA DE CHIAPAS

Facultad de contaduría y administración

Materia: Modelos y metodologías de desarrollo de software

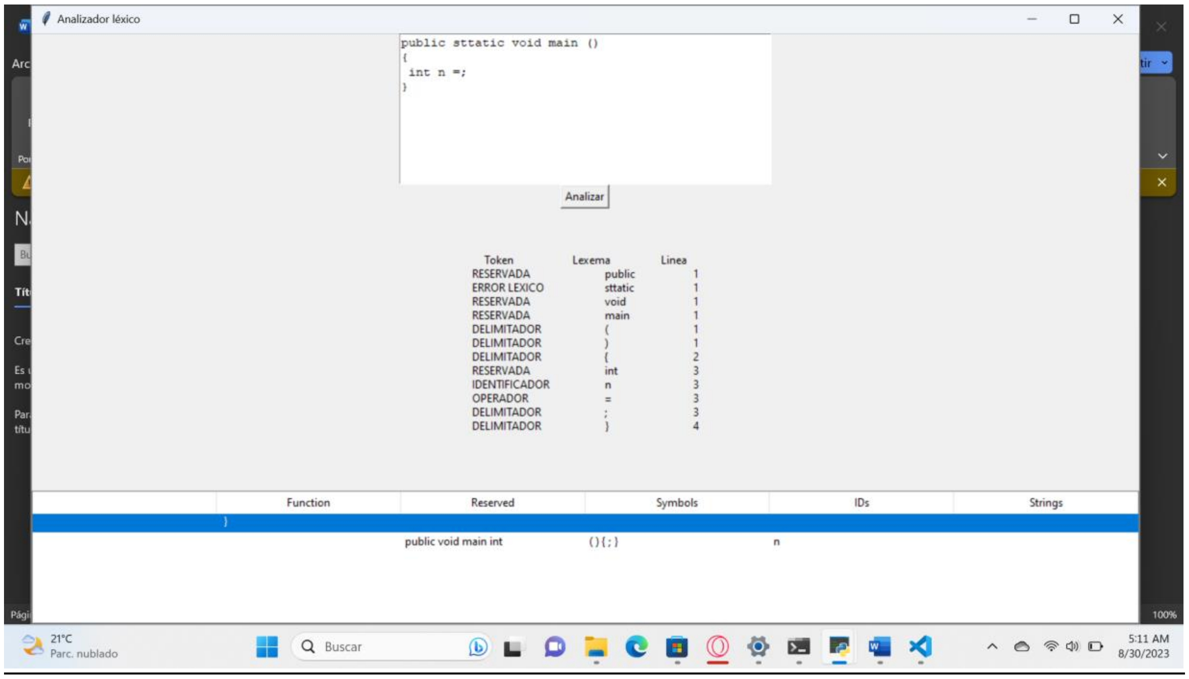
Docente: ***Luis Gutiérrez Alfaro***

Actividad: **Investigar los siguientes conceptos del analizador léxico.**

Alumno: Diego González Carpio

```
import tkinter as tk
import re
from tkinter import ttk

class Lexer:
    def __init__(self):
        self.RESERVADA = ['for', 'do', 'while', 'if', 'else', 'public',
'static', 'void', 'int','main']
        self.OPERADOR = ['=', '+', '-', '*', '/']
        self.DELIMITADOR = ['(', ')', '{', '}', ';']
        self.tokens_regex = {
            'RESERVADA': '|'.join(r'\b' + re.escape(keyword) + r'\b' for
keyword in self.RESERVADA),
            'OPERADOR': '|'.join(map(re.escape, self.OPERADOR)),
            'DELIMITADOR': '|'.join(map(re.escape, self.DELIMITADOR)),
            'NUMERO': r'\d+(\.\d+)?',
            'IDENTIFICADOR': r'[A-Za-z_]+'
        }
        self.token_patterns =
re.compile('|'.join(f'(?P<{t}>{self.tokens_regex[t]})' for t in
self.tokens_regex))

    def tokenize(self, text):
        tokens = []
        lines = text.split('\n')
        NumeroLinea = 1
        for line in lines:
            line_has_tokens = False
            for match in self.token_patterns.finditer(line):
                line_has_tokens = True
                for token_type, token_value in match.groupdict().items():
                    if token_type == 'IDENTIFICADOR' and token_value and
len(token_value) > 1:
                        tokens.append((NumeroLinea, 'ERROR LEXICO',
token_value))
                    elif token_type == 'IDENTIFICADOR' and token_value and
len(token_value) == 1:
                        tokens.append((NumeroLinea, 'IDENTIFICADOR',
token_value))
                    elif token_value:
                        tokens.append((NumeroLinea, token_type,
token_value))
            if line_has_tokens:
                NumeroLinea += 1
        return tokens
```

```python
    def analyze(self, text):
        tokens = self.tokenize(text)
        result = "Token\t\tLexema\t\tLinea\n"
        for NumeroLinea, token_type, token_value in tokens:
            result += f"{token_type}\t\t{token_value}\t\t{NumeroLinea}\n"
        return result

class LexerApp:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Analizador léxico")

        self.text_input = tk.Text(self.window, height=10, width=50)
        self.text_input.pack()

        self.analyze_button = tk.Button(self.window, text="Analizar",
command=self.analyze_text)
        self.analyze_button.pack()

        self.result_label = tk.Label(self.window, text="Vacio", height=20,
width=50)
        self.result_label.pack()

        self.table_frame = tk.Frame(self.window)
        self.table_frame.pack()

        self.table = ttk.Treeview(self.table_frame, columns=("Function",
"Reserved", "Symbols", "IDs", "Strings"))
        self.table.heading("#1", text="Function")
        self.table.heading("#2", text="Reserved")
        self.table.heading("#3", text="Symbols")
        self.table.heading("#4", text="IDs")
        self.table.heading("#5", text="Strings")
        self.table.pack()

    def analyze_text(self):
        lexer = Lexer()
        text = self.text_input.get("1.0", "end-1c")
        result = lexer.analyze(text)
        self.result_label.config(text=result)

        self.table.delete(*self.table.get_children())

        function, data = self.analyze(text)
        self.table.insert("", "end", values=(function, '', '', '', ''))

        for row in data:
            self.table.insert("", "end", values=row)

    def analyze(self, text):
        lexer = Lexer()
        tokens = lexer.tokenize(text)
        function = ''
        reserved = ''
        symbols = ''
        ids = ''
        strings = ''
        data = []

        for NumeroLinea, token_type, token_value in tokens:
            if token_type in lexer.RESERVADA:
                if token_type == 'void':
                    function += f'{token_value}\n'
            elif token_type == 'RESERVADA':
                reserved += f'{token_value} '
            elif token_type == 'DELIMITADOR':
                symbols += f'{token_value} '
            elif token_type == 'IDENTIFICADOR':
                ids += f'{token_value} '
            elif token_type == 'NUMERO':
                strings += f'{token_value} '

        data.append((function, reserved, symbols, ids, strings))
        return f" {token_value}\n", data

    def run(self):
        self.window.mainloop()

app = LexerApp()
app.run()
```



```
public sttatic void main ()
{
 int n =;
}
```

```
   Token          Lexema      Linea
RESERVADA           public        1
ERROR LEXICO        sttatic       1
RESERVADA           void          1
RESERVADA           main          1
DELIMITADOR         (             1
DELIMITADOR         )             1
DELIMITADOR         {             2
RESERVADA           int           3
IDENTIFICADOR       n             3
OPERADOR            =             3
DELIMITADOR         ;             3
DELIMITADOR         }             4
```

	Function	Reserved	Symbols	IDs	Strings
	}				
		public void main int	( ) { ; }	n	