



**Universidad Autónoma de Chiapas**  
**Facultad de contaduría y administración C-I**



**Carrera:**

Lic. En ing en desarrollo y tecnologías de software.

**Materia:**

Compiladores.

**Catedrático:**

Mtro. Luis Gutiérrez Alfaro.

**Nombre del alumno:**

González Aguilar Eduardo - A211154

**Semestre:** 6. **Grupo:** M.

**Nombre de la actividad:**

Proyecto Final.

**Link de GitHub:**

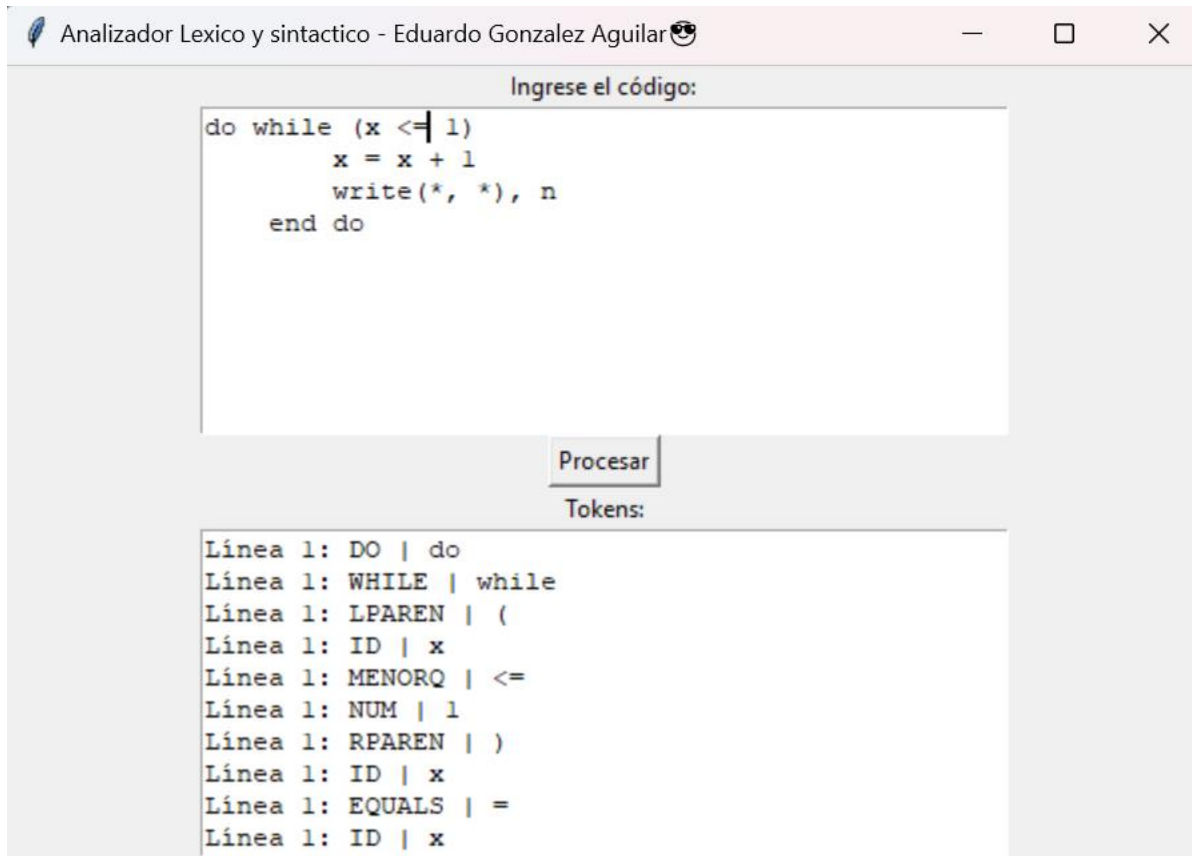
[AguilarEduardo/Compilador \(github.com\)](https://github.com/AguilarEduardo/Compilador)

**Fecha de entrega:**

16/11/2023.

En mi caso realice mí que mi analizador léxico y sintáctico analizara el “do - while” de fortran, el programa lo ejecuta perfectamente, detecta los errores sintácticos cuando el do while no esta bien escrito. Cabe aclarar que mi programa no define correctamente la ubicación de las líneas.

Anexo pruebas de escritorio y capturas de código. Además en github anexo el programa en .py.



The screenshot shows a window titled "Analizador Lexico y sintactico - Eduardo Gonzalez Aguilar". Inside the window, there is a text area labeled "Ingrese el código:" containing the following Fortran code:

```
do while (x <= 1)
  x = x + 1
  write(*, *) , n
end do
```

Below the text area is a button labeled "Procesar". Underneath the button is a section labeled "Tokens:" which displays the following output:

```
Línea 1: DO | do
Línea 1: WHILE | while
Línea 1: LPAREN | (
Línea 1: ID | x
Línea 1: MENORQ | <=
Línea 1: NUM | 1
Línea 1: RPAREN | )
Línea 1: ID | x
Línea 1: EQUALS | =
Línea 1: ID | x
```

Ingrese el código:

```
do while (x <= 1
    x = x + 1
    write(*, *) , n
end do
```

Procesar

Tokens:

```
Línea 1: DO | do
Línea 1: WHILE | while
Línea 1: LPAREN | (
Línea 1: ID | x
Línea 1: MENORQ | <=
Línea 1: NUM | 1
Línea 1: ID | x
Línea 1: EQUALS | =
Línea 1: ID | x
Línea 1: PLUS | +
```

Error de sintaxis



Error de sintaxis en 'x'  
En la línea 1

Aceptar

C: > Users > sklao > OneDrive > Escritorio > Uni OwO

```
1  #Importando Librerias
2  import re
3  import tkinter as tk
4  from tkinter import messagebox
5  from ply import lex, yacc
6
7  #Definir los tokens
8  tokens = (
9      'DO',
10     'WHILE',
11     'PRINT',
12     'ID',
13     'NUM',
14     'STRING',
15     'SEMICOLON',
16     'LPAREN',
17     'RPAREN',
18     'LBRACE',
19     'RBRACE',
20     'MAYOR',
21     'MENORQ',
22     'EQUALS',
23     'PLUS',
24     'COMA',
```

```
25 )
26 t_SEMICOLON = r';'
27 t_LPAREN = r'\('
28 t_RPAREN = r'\)'
29 t_LBRACE = r'\{'
30 t_RBRACE = r'\}'
31 t_MAYOR = r'>'
32 t_MENORQ = r'<='
33 t_EQUALS = r'='
34 t_PLUS = r'\+'
35 t_COMA = r'\,'
36
37 def t_newline(t):
38     r'\n+'
39     t.lexer.lineno += len(t.value)
40
41 def t_ID(t):
42     r'[a-zA-Z*][a-zA-Z0-9]*'
43     if t.value == 'do':
44         t.type = 'DO'
45     elif t.value == 'while':
46         t.type = 'WHILE'
47     elif t.value == 'print':
48         t.type = 'PRINT'
49     return t
50
```

```
51 # Regla para identificar números
52 def t_NUM(t):
53     r'\d+'
54     t.value = int(t.value)
55     return t
56
57 # Ignorar espacios en blanco y saltos de línea
58 t_ignore = ' \t\n'
59
60 def t_error(t):
61     error_message(f"Token desconocido '{t.value[0]}'", t.lineno)
62     t.lexer.skip(1)
63
64 # Construcción del Lexer
65 lexer = lex.lex()
66
67 # Definición de la gramática para el análisis sintáctico
68 def p_for_loop(p):
69     '''do_while_loop : DO WHILE LPAREN ID MENORQ NUM RPAREN ID EQUALS ID PLUS NUM ID LPAREN ID COMA ID RPAREN COMA ID ID DO'''
70     pass
71
72 # Manejo de errores de sintaxis
73 def p_error(p):
74     if p:
75         error_message(f"Error de sintaxis en '{p.value}'", p.lineno)
76     else:
77         error_message("Error de sintaxis: final inesperado del código", len(code_text.get("1.0", "end-1c").split('\n')))
78
```

```

79 # Construcción del parser
80 parser = yacc.yacc()
81
82 # Función para el análisis Léxico
83 def lex_analyzer(code):
84     lexer.input(code)
85     tokens_list = []
86     while True:
87         tok = lexer.token()
88         if not tok:
89             break
90         tokens_list.append((tok.lineno, tok.type, tok.value))
91     return tokens_list
92
93 # Función para el análisis sintáctico
94 def parse_code(code):
95     parser.parse(code, lexer=lexer)
96
97 def error_message(message, line_number):
98     messagebox.showerror("Error de sintaxis", f"{message}\nEn la línea {line_number}")
99

```

```

100 # Función para procesar el código ingresado
101 def process_code():
102     code = code_text.get("1.0", "end-1c")
103     tokens_list = lex_analyzer(code)
104     result_text.delete("1.0", "end")
105     for token in tokens_list:
106         line_number, token_type, token_value = token
107         result_text.insert("end", f"Línea {line_number}: {token_type} | {token_value}\n")
108
109     try:
110         parse_code(code)
111     except Exception as e:
112         messagebox.showerror("Error al analizar", str(e))
113
114 # Creación de la ventana de la interfaz gráfica
115 window = tk.Tk()
116 window.title("Analizador Lexico y sintactico - Eduardo Gonzalez Aguilar 😊")
117 window.geometry("600x400")
118
119 # Etiqueta y campo de texto para ingresar el código
120 code_label = tk.Label(window, text="Ingrese el código:")
121 code_label.pack()
122
123 code_text = tk.Text(window, height=10, width=50)
124 code_text.pack()
125
126 # Botón para procesar el código
127 process_button = tk.Button(window, text="Procesar", command=process_code)
128 process_button.pack()
129

```

```
130     # Etiqueta y campo de texto para mostrar los tokens
131     result_label = tk.Label(window, text="Tokens:")
132     result_label.pack()
133
134     result_text = tk.Text(window, height=10, width=50)
135     result_text.pack()
136
137     # Ejecución de la interfaz gráfica
138     window.mainloop()
139
```