



# MANUAL TÉCNICO

## PRÁCTICA 1

ALDO SAÚL VÁSQUEZ MOREIRA  
CARNET 202109754  
LAB. LENGUAJES FORMALES Y DE PROG.

## **ENCABEZADO**

Nombre: Aldo Saúl Vásquez Moreira

Carnet: 20109754

Laboratorio Lenguajes Formales y de Programación.

Nombre de Sistema: Gestor de Créditos Facultad de Ingeniería Universidad San Carlos de Guatemala

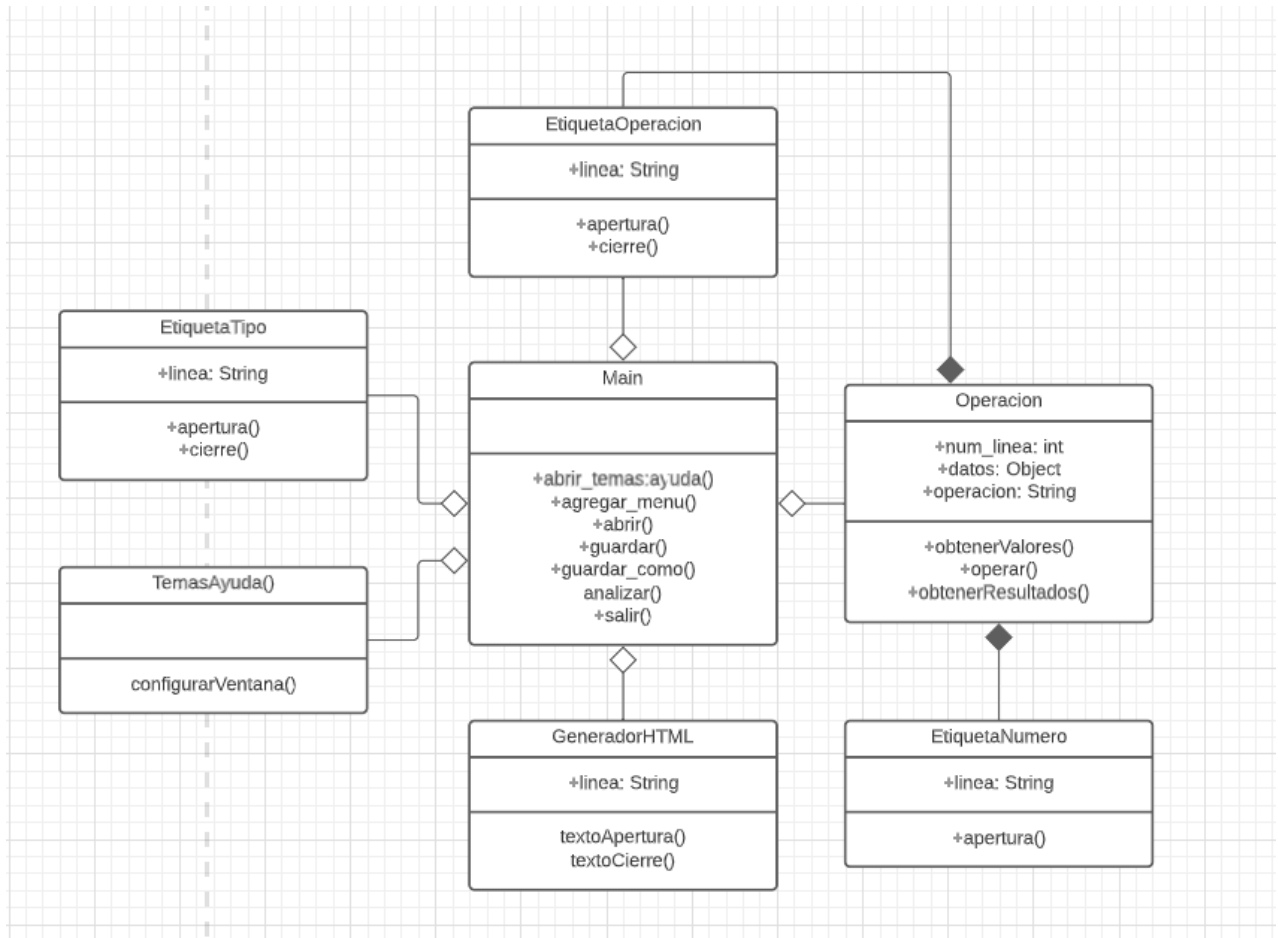
## **PRINCIPIO, TÉCNICA O PARADIGMA APLICADO DE PROGRAMACIÓN**

Se utilizó el paradigma de Programación Orientado a Objetos con Python.

## **CONVENCIONES DE NOMENCLATURA**

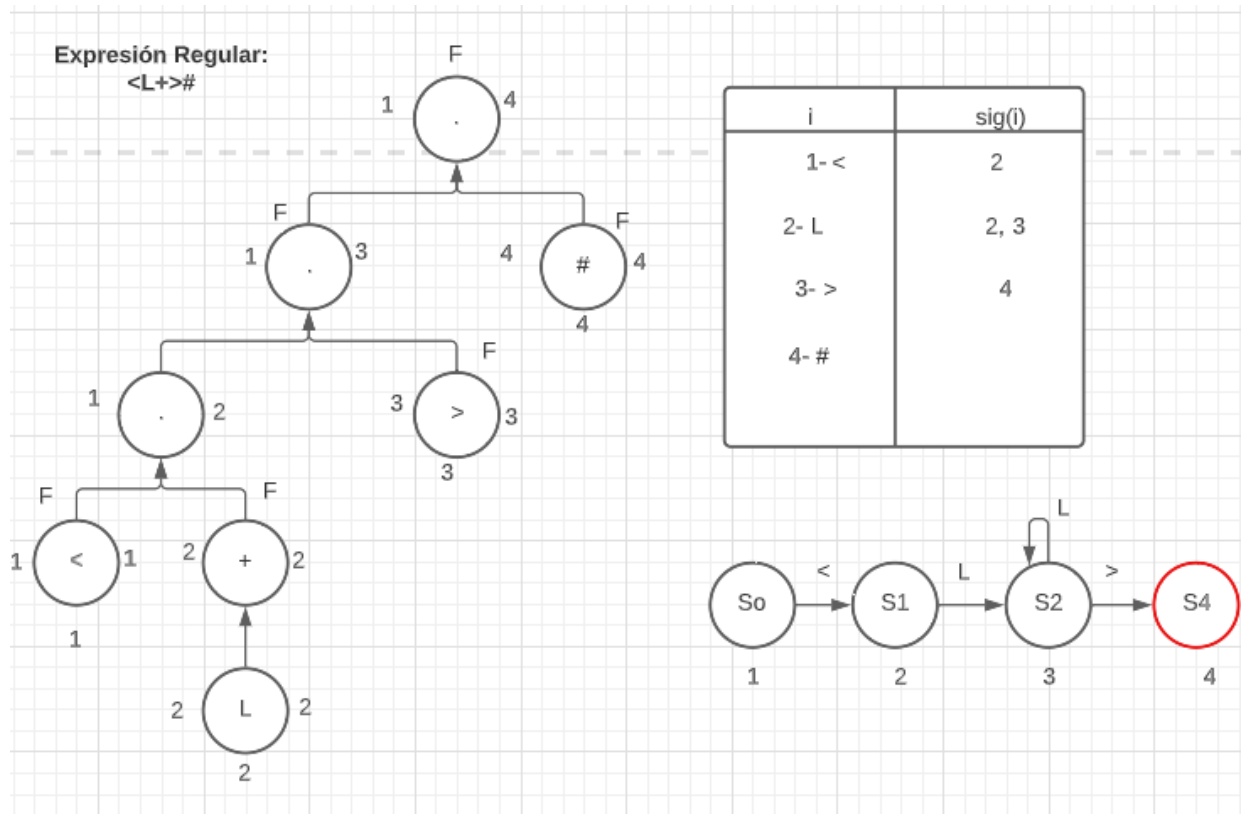
- Se declararon las clases con letra inicial mayúscula.
- Se declararon los métodos con letra inicial minúscula y aplicando la convención Snake Case y Camel Case.
- Se declararon las variables con letra inicial minúscula y con un guion bajo en caso fueran necesarias más de dos palabras.

## DIAGRAMA DE CLASES

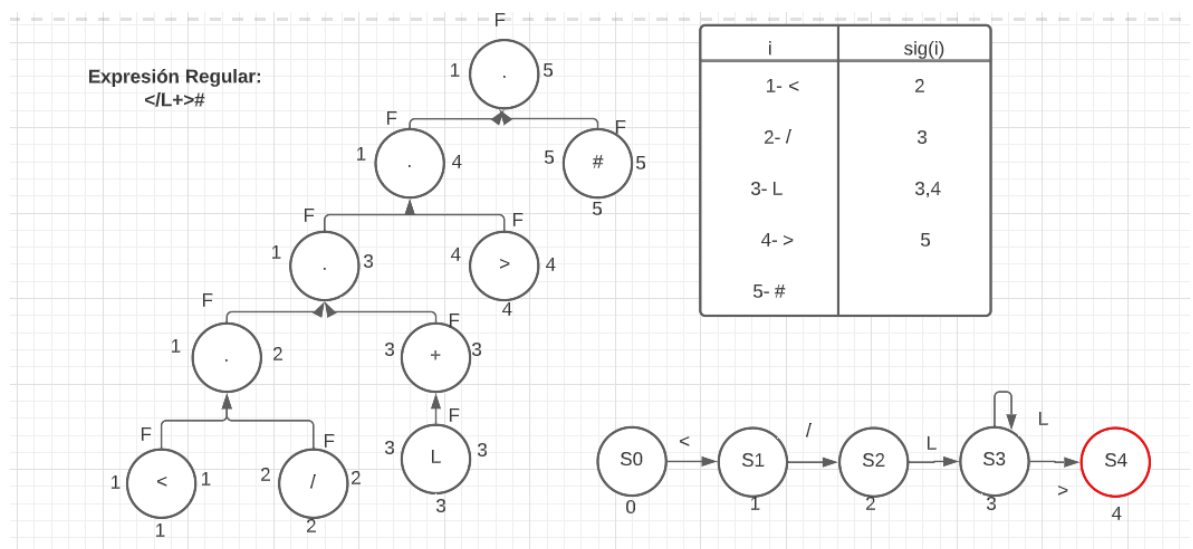


## AUTÓMATAS FINITOS DETERMINISTAS

- Etiquetas de Apertura <Tipo> y <Texto>

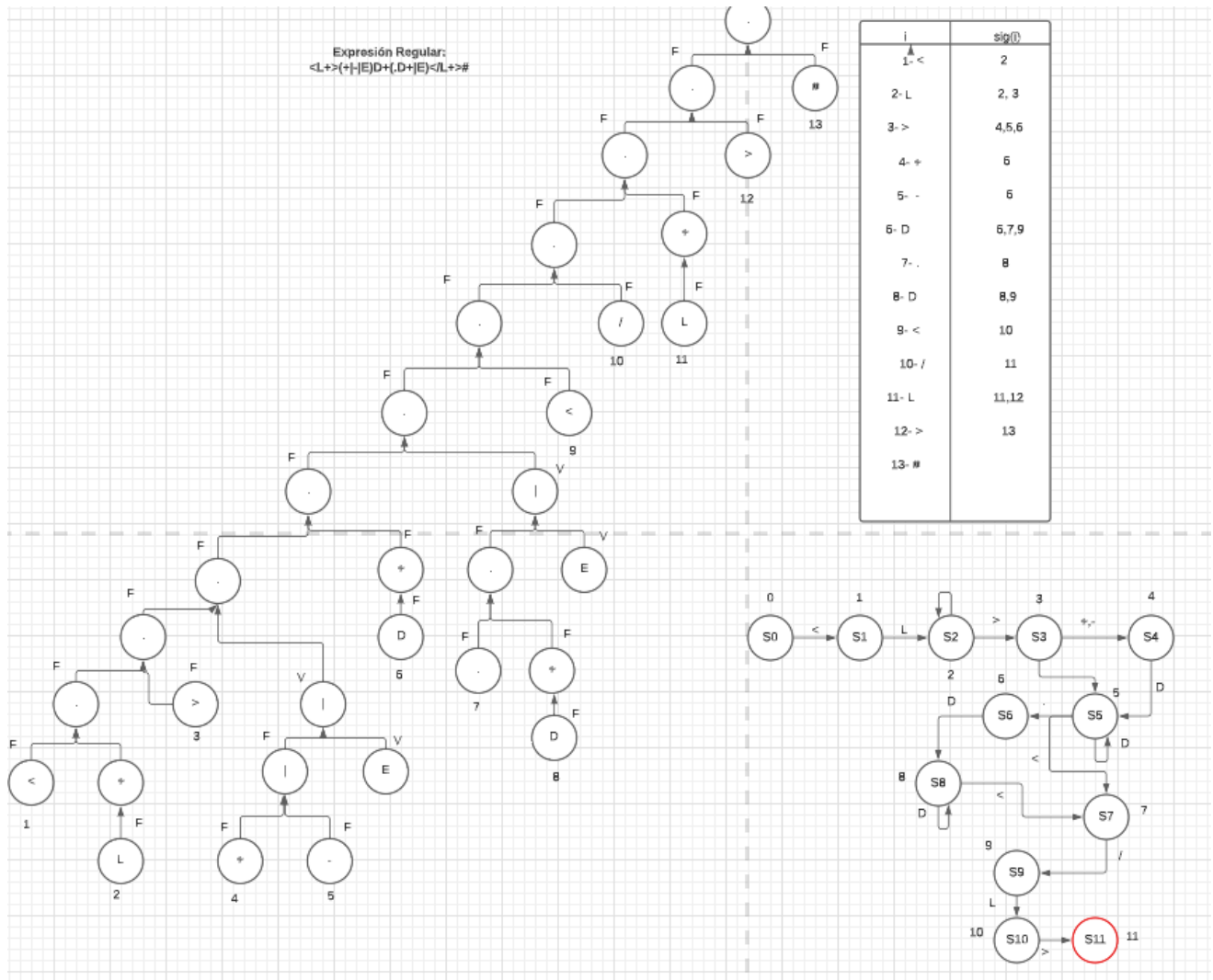


- Etiqueta de cierre </Tipo>, </Texto> y </Operacion>



- Etiqueta de Apertura <Operacion = XXXX>





## MÉTODOS PRINCIPALES

- **apertura:** Cada clase de las diferentes etiquetas contiene este método el cual corresponde al encargado del análisis léxico de la etiqueta de apertura.

```

6
7     def apertura(self):
8         cadena = ""
9         self.estado = 1
10        for i in range(0, len(self.linea)):
11            self.transicion = self.linea[i]
12            if self.estado == 1:
13                if self.transicion == "<":
14                    cadena += self.transicion
15                    self.estado = 2
16            elif self.estado == 2:
17                if self.transicion.isalpha() and self.linea[i+1] != ">":
18                    cadena += self.transicion
19                    self.estado = 2
20                elif self.transicion.isalpha() and self.linea[i+1] == ">":
21                    cadena += self.transicion
22                    self.estado = 3
23                elif self.transicion.isalpha() == False:
24                    return False
25            elif self.estado == 3:
26                if self.transicion == ">":
27                    cadena += self.transicion
28                    self.estado = 3
29        if cadena == "<Tipo>":
30            return True
31        else:
32            return False

```

- **cierre:** Cada clase de las diferentes etiquetas contiene este método el cual corresponde al encargado del análisis léxico de la etiqueta de

```

34     def cierre(self):
35         cadena = ""
36         self.estado = 1
37         for i in range(0, len(self.linea)):
38             self.transicion = self.linea[i]
39             if self.estado == 1:
40                 if self.transicion == "<":
41                     cadena += self.transicion
42                     self.estado = 2
43             elif self.estado == 2:
44                 if self.transicion == "/":
45                     cadena += self.transicion
46                     self.estado = 3
47             elif self.estado == 3:
48                 if self.transicion.isalpha() and self.linea[i+1] != ">":
49                     cadena += self.transicion
50                     self.estado = 3
51                 elif self.transicion.isalpha() and self.linea[i+1] == ">":
52                     cadena += self.transicion
53                     self.estado = 4
54                 elif self.transicion.isalpha() == False:
55                     return False
56             elif self.estado == 4:
57                 if self.transicion == ">":
58                     cadena += self.transicion
59                     self.estado = 4
60        if cadena == "</Tipo>":
61            return True
62        else:
63            return False

```

- **obtenerValores:** Este método se encarga de identificar la etiqueta de apertura de *operación*. Una vez se verifica que la etiqueta es correcta, identifica las etiquetas *número*, y almacena los datos de cada una de las etiquetas.

```

#metodo para retonar los valores de cada operacion
def obtenerValores(self, num_linea):
    contador = 0
    operandos = []
    while EtiquetaOperacion(self.datos[num_linea]).cierre() != True:
        etiqueta_numero = EtiquetaNumero(self.datos[num_linea])
        retorno = etiqueta_numero.apertura()
        if retorno != False:
            operandos.append(retorno)
            num_linea += 1
            contador += 1
        else:
            break
    num_linea += 1
    contador += 1
    if retorno != False:
        return [operandos, contador]
    else:
        return [False, contador]

```

- **operar:** Una vez se han almacenado los datos de la operación y el tipo de operación este método se encarga de realizar la operación matemática correspondiente.

```

32 #metodo para operar
33 def operar(self, datos):
34     if self.operacion == "suma":
35         total = 0
36         cadena = ""
37         cont = 0
38         for i in datos:
39             cont += 1
40             cadena += str(i)
41             if cont == (len(datos)):
42                 pass
43             else:
44                 cadena += "+"
45             total += i
46             cadena += "="
47             cadena += str(round(total,2))
48     elif self.operacion == "resta":
49         total = 0
50         cadena = ""
51         cont = 0
52         for i in datos:
53             cont += 1
54             cadena += str(i)
55             if cont == (len(datos)):
56                 pass
57             else:
58                 cadena += "-"
59             if cont == 1:
60                 total = i
61             else:
62                 total -= i
63             cadena += "="

```



```

62         total -= i
63         cadena += "-"
64         cadena += str(round(total,2))
65     elif self.operacion == "multiplicacion":
66         total = 0
67         cadena = ""
68         cont = 0
69         for i in datos:
70             cont += 1
71             cadena += str(i)
72             if cont == (len(datos)):
73                 pass
74             else:
75                 cadena += "*"
76                 if cont == 1:
77                     total = i
78                 else:
79                     total *= i
80         cadena += "="
81         cadena += str(round(total,2))
82     elif self.operacion == "division":
83         total = 0
84         cadena = ""
85         cont = 0
86         for i in datos:
87             cont += 1
88             cadena += str(i)
89             if cont == (len(datos)):
90                 pass
91             else:
92                 cadena += "/"
93                 if cont == 1:
94                     total = i
95                 else:
96                     total /= i
97         cadena += "="
98         cadena += str(round(total,2))
99     elif self.operacion == "potencia":
100         total = datos[0]**datos[1]
101         cadena = str(datos[0])+"^"+str(datos[1])+"="+str(round(total,2))
102     elif self.operacion == "raiz":
103         total = pow(datos[0], 1/datos[1])
104         cadena = "sqrt"+"("+str(datos[0])+" )"+"="+str(round(total,2))
105     elif self.operacion == "inverso":
106         total = (1/datos[0])
107         cadena = str(datos[0])+"^-1"+"="+str(round(total,2))
108     elif self.operacion == "seno":
109         total = math.sin(datos[0])
110         cadena = "sin"+"("+str(datos[0])+" )"+"="+str(round(total,2))
111     elif self.operacion == "coseno":
112         total = math.cos(datos[0])
113         cadena = "cos"+"("+str(datos[0])+" )"+"="+str(round(total,2))
114     elif self.operacion == "tangente":
115         total = math.tan(datos[0])
116         cadena = "tan"+"("+str(datos[0])+" )"+"="+str(round(total,2))
117     elif self.operacion == "mod":
118         total = 0
119         cadena = ""
120         cont = 0
121         for i in datos:
122             cont += 1
123             cadena += str(i)
124             if cont == (len(datos)):
125                 pass
126             else:
127                 cadena += "%"
128                 if cont == 1:
129                     total = i
130                 else:
131                     total %= i
132         cadena += "="
133         cadena += str(round(total,2))
134     return cadena

```

- **obtenerResultados:** Este método se encarga de mostrar en una cadena de texto cada uno de los operandos y el resultado final.

```

135
136 def obtenerResultados(self):
137     error = False
138     linea = 0
139     resultado = ""
140     if self.operacion == "suma":
141         if self.obtenerValores(self.num_linea)[0] != False:
142             operandos, contador = self.obtenerValores(self.num_linea)
143             resultado = self.operar(operandos)
144             self.num_linea += contador
145         else:
146             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
147     elif self.operacion == "resta":
148         if self.obtenerValores(self.num_linea)[0] != False:
149             operandos, contador = self.obtenerValores(self.num_linea)
150             resultado = self.operar(operandos)
151             self.num_linea += contador
152         else:
153             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
154     elif self.operacion == "multiplicacion":
155         if self.obtenerValores(self.num_linea)[0] != False:
156             operandos, contador = self.obtenerValores(self.num_linea)
157             resultado = self.operar(operandos)
158             self.num_linea += contador
159         else:
160             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
161     elif self.operacion == "division":
162         if self.obtenerValores(self.num_linea)[0] != False:
163             operandos, contador = self.obtenerValores(self.num_linea)
164             resultado = self.operar(operandos)
165             self.num_linea += contador
166         else:
167             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
168
169     elif self.operacion == "potencia":
170         if self.obtenerValores(self.num_linea)[0] != False:
171             operandos, contador = self.obtenerValores(self.num_linea)
172             resultado = self.operar(operandos)
173             self.num_linea += contador
174         else:
175             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
176     elif self.operacion == "raiz":
177         if self.obtenerValores(self.num_linea)[0] != False:
178             operandos, contador = self.obtenerValores(self.num_linea)
179             resultado = self.operar(operandos)
180             self.num_linea += contador
181         else:
182             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
183     elif self.operacion == "inverso":
184         if self.obtenerValores(self.num_linea)[0] != False:
185             operandos, contador = self.obtenerValores(self.num_linea)
186             resultado = self.operar(operandos)
187             self.num_linea += contador
188         else:
189             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
190     elif self.operacion == "seno":
191         if self.obtenerValores(self.num_linea)[0] != False:
192             operandos, contador = self.obtenerValores(self.num_linea)
193             resultado = self.operar(operandos)
194             self.num_linea += contador
195         else:
196             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
197     elif self.operacion == "coseno":
198         if self.obtenerValores(self.num_linea)[0] != False:
199             operandos, contador = self.obtenerValores(self.num_linea)
200             resultado = self.operar(operandos)
201             self.num_linea += contador
202         else:
203             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
204     elif self.operacion == "tangente":
205         if self.obtenerValores(self.num_linea)[0] != False:
206             operandos, contador = self.obtenerValores(self.num_linea)
207             resultado = self.operar(operandos)
208             self.num_linea += contador
209         else:
210             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
211     elif self.operacion == "mod":
212         if self.obtenerValores(self.num_linea)[0] != False:
213             operandos, contador = self.obtenerValores(self.num_linea)
214             resultado = self.operar(operandos)
215             self.num_linea += contador
216         else:
217             print(f"Error en la linea {self.num_linea+self.obtenerValores(self.num_linea)[1]}")
218
219     return resultado, self.num_linea

```

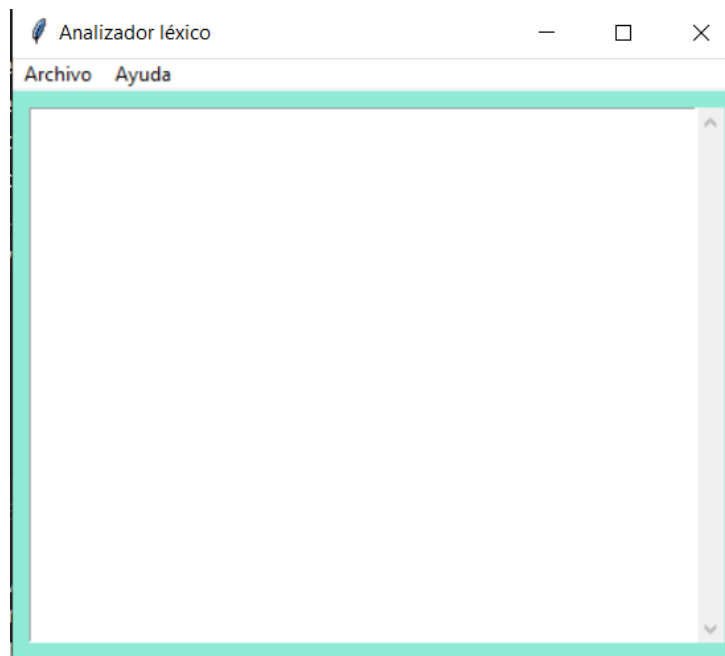
## DESCRIPCIÓN

Se utilizó el Paradigma de Programación Orientado a Objetos debido a la capacidad de abstracción que se tiene para modelar objetos de la vida real en el código. Asimismo, se cuenta con diversas clases para modular de una mejor forma el código para que así fuera más visible y entendible al momento de tener la necesidad de realizar un cambio.

Adicionalmente, se utilizó la biblioteca Tkinter debido a que viene por defecto en las bibliotecas de Python y ofrece muchas opciones para crear una buena interfaz para los usuarios. También se utilizó la biblioteca *cmath* para realizar cada una de las operaciones matemáticas que pueda ingresar el usuario.

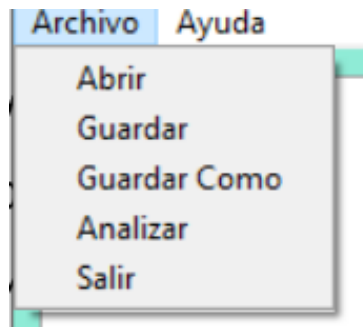
## INTERFACES PRINCIPALES

- **Pantalla de inicio**



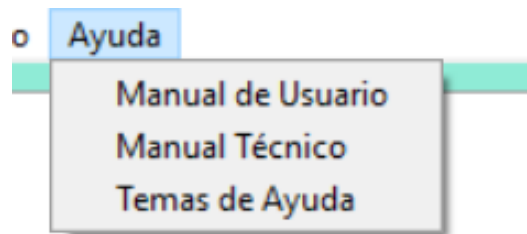
Esta ventana muestra el área de texto en la cual se adjuntará el contenido de un archivo al momento de abrirlo. Así, como el menú de barra con los menús *Archivo* y *Ayuda*.

- **Menú archivo**



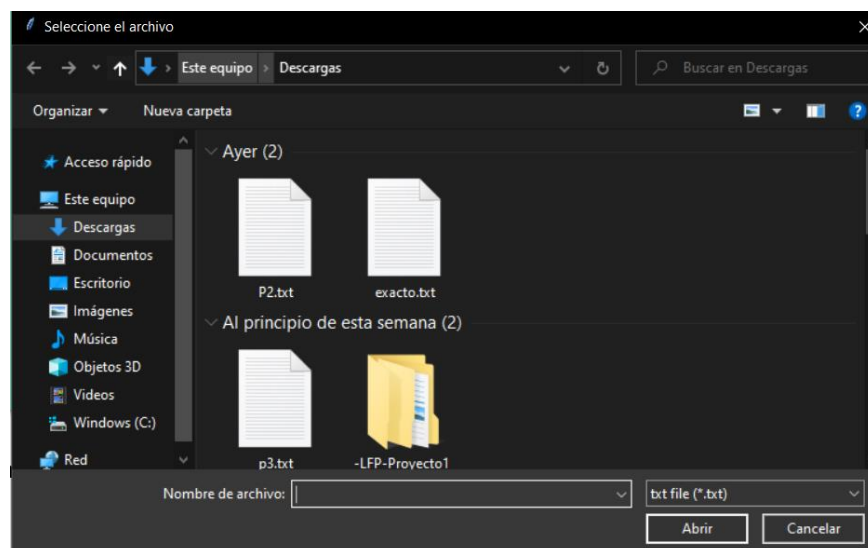
Este menú contiene las opciones de menú mencionadas anteriormente.

- **Menú ayuda**



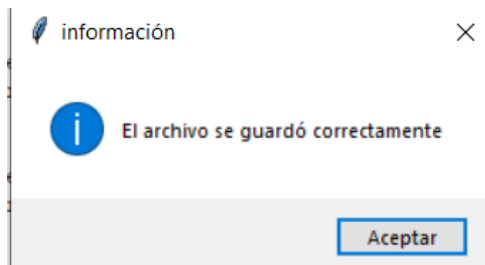
Este menú contiene las opciones de menú mencionadas anteriormente.

- **Opción abrir archivo**



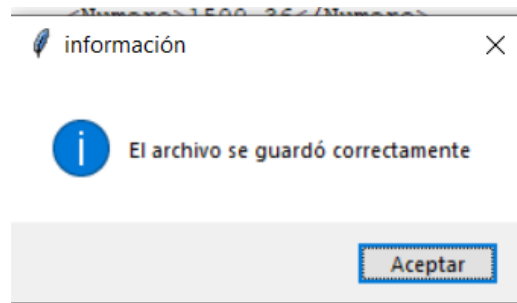
Esta opción despliega el administrador de archivos y únicamente los archivos con la extensión permitida.

- **Opción guardar**



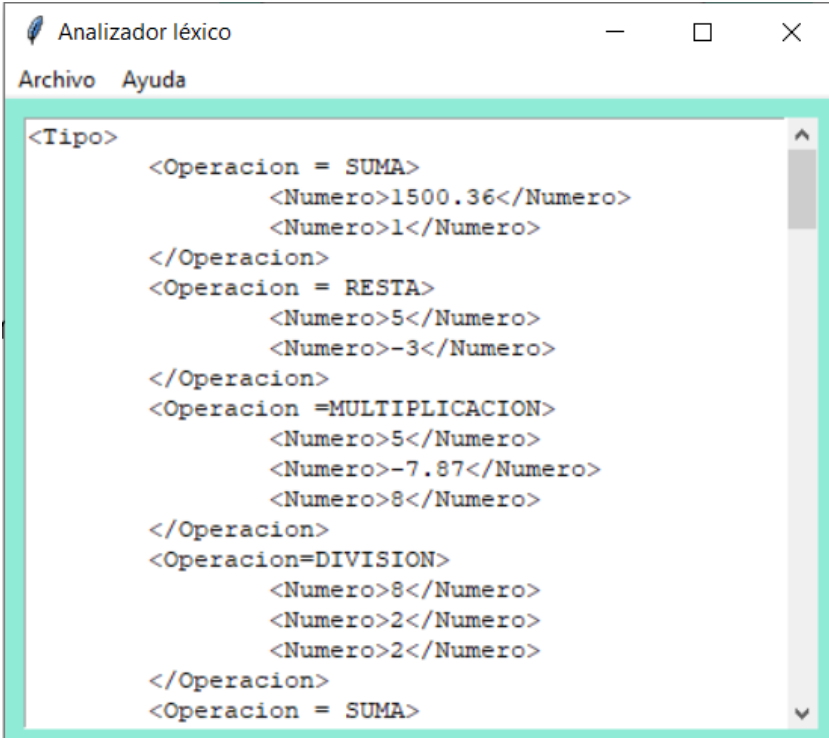
Si el archivo ya se encontraba guardado con anterioridad únicamente indicará que se han guardado los nuevos cambios.

- **Opción guardar como**



Esta ventana nos permitirá asignar un nombre al archivo e indicará por medio de una ventana emergente que se ha guardado correctamente.

- **Ventana Principal con contenido cargado**

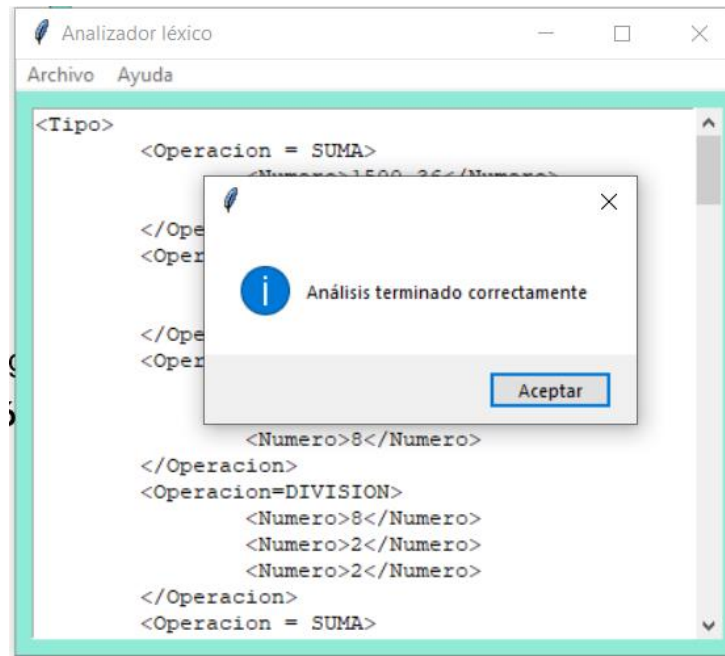


The screenshot shows a window titled "Analizador léxico" with a menu bar containing "Archivo" and "Ayuda". The main content area displays a list of operations and numbers in a structured format. The list is enclosed in a light blue border and has a vertical scrollbar on the right side. The text is as follows:

```
<Tipo>  
  <Operacion = SUMA>  
    <Numero>1500.36</Numero>  
    <Numero>1</Numero>  
  </Operacion>  
  <Operacion = RESTA>  
    <Numero>5</Numero>  
    <Numero>-3</Numero>  
  </Operacion>  
  <Operacion = MULTIPLICACION>  
    <Numero>5</Numero>  
    <Numero>-7.87</Numero>  
    <Numero>8</Numero>  
  </Operacion>  
  <Operacion = DIVISION>  
    <Numero>8</Numero>  
    <Numero>2</Numero>  
    <Numero>2</Numero>  
  </Operacion>  
  <Operacion = SUMA>
```

Al cargar un archivo al sistema este se mostrará de la siguiente forma.

- **Opción Analizar:**

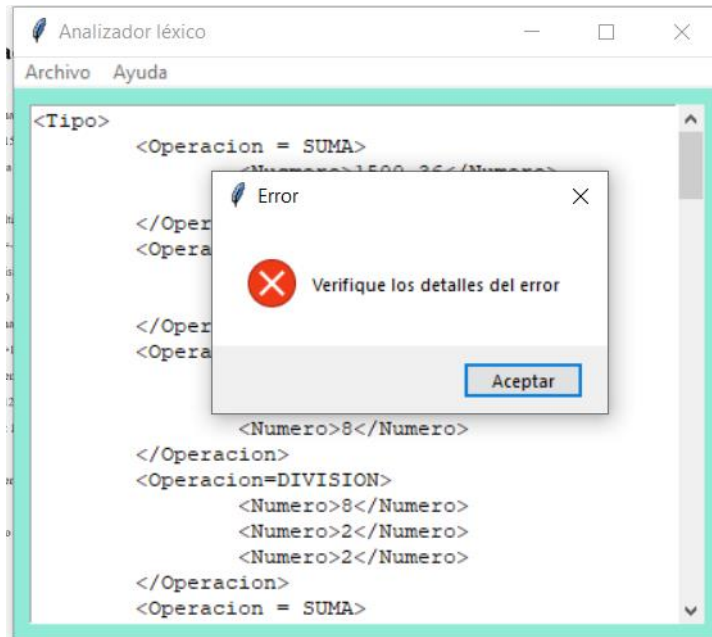


Si el archivo ingresado no contiene ningún tipo de error el sistema nos indicará que se analizó correctamente y se generará un archivo con extensión “html” con los resultados correspondientes:

#### Generacion Archivo HTML

```
Operaciones
Operacion suma 1:
1500.36+1.0=1501.36
Operacion resta 1:
5.0--3.0=8.0
Operacion multiplicacion 1:
5.0*-7.87*8.0=-314.8
Operacion division 1:
8.0/2.0/2.0=2.0
Operacion suma 2:
15.36+100.36=115.72
Operacion potencia 1:
5.0^9.0=1953125.0
Operacion raiz 1:
sqrt(9.0)=1.32
Operacion inverso 1
9.0^-1=0.11
Operacion seno 1:
sin(9.0)=0.41
```

En caso de que el archivo haya presentado errores el sistema lo informará por medio de una ventana emergente:



Seguidamente se generará un archivo con extensión “html” el cual mostrará la ubicación de dicho error:

## Generacion Archivo de Errores HTML

| Tipo  | Fila |
|-------|------|
| Error | 3    |



## PLANIFICACIÓN O ESTIMACIÓN

- **Interfaz Gráfica:** 1 hora.
- **Parte teórica:** 3 días.
- **Programación de AFD's:** 1 semana.
- **Lógica del programa:** 2 semanas.

## GLOSARIO

1. **Extensión de archivo:** Las extensiones indican qué aplicación ha creado el archivo o puede abrirlo, y qué icono se debe utilizar para el archivo. Por ejemplo, la extensión docx indica al equipo que Microsoft Word puede abrir el archivo y que debe mostrar un icono de Word al verlo en el Explorador de archivos.
2. **Explorador de archivos:** El Explorador de archivos o Explorador de Windows, como fue nombrado hasta la edición de Windows 8, es el administrador de archivos oficial del sistema operativo Microsoft Windows.
3. **Software:** Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
4. **Sistema:** En terminología informática el software de sistema, denominado también software de base, consiste en un software que sirve para controlar un proceso.
5. **Paradigma:** El término paradigma es empleado para indicar un patrón, modelo, ejemplo o arquetipo. Por lo general hace referencia a una serie de teorías que son tomadas como modelo a seguir al momento de solucionar cualquier tipo de problemas que puedan surgir en determinadas situaciones.