

**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**  
**FACULTAD DE INGENIERÍA**  
**ESCUELA DE CIENCIAS Y SISTEMAS**  
**LABORATORIO DE LENGUAJES FORMALES Y DE PROGRAMACIÓN**  
**AUX: DIEGO OBÍN**

**JUSTY SEBASTIAN RODRÍGUEZ DEL CID**  
**202100058**  
**PROYECTO 1**

A decorative graphic on the left side of the page consisting of white lines and circles on a purple background, resembling a circuit board or a stylized tree structure.

# **MANUAL TÉCNICO**

```
from Instrucciones.aritmeticas import *
from Instrucciones.trigonometricas import *
from Abstract.lexema import *
from Abstract.numeros import *
#from Errores.errores import *
import graphviz
from graphviz import *

from tkinter import *
from tkinter import filedialog
from tkinter import messagebox as mb
import webbrowser as wb
from tkinter import scrolledtext as st
import tkinter as tk
import os
```

**La primera parte consiste en importar las librerías que necesitaremos para ejecutar el proyecto.**

```
17
18 reserved = {
19
20     'ROPERACION'      : 'Operacion',
21     'RVALOR1'         : 'Valor1',
22     'RVALOR2'         : 'Valor2',
23     'RSUMA'           : 'Suma',
24     'RRESTA'          : 'Resta',
25     'RMULTIPLICACION' : 'Multiplicacion',
26     'RDIVISION'       : 'Division',
27     'RPOTENCIA'       : 'Potencia',
28     'RRAIZ'           : 'Raiz',
29     'RINVERSO'        : 'Inverso',
30     'RSENO'           : 'Seno',
31     'RCOSENO'         : 'Coseno',
32     'RTANGENTE'       : 'Tangente',
33     'RMODULO'         : 'Modulo',
34     'RTEXTO'          : 'Texto',
35     'RCOLORFONDONODO' : 'Color-Fondo-Nodo',
36     'RCOLORFUENTENODO': 'Color-Fuente-Nodo',
37     'RFORMADNODO'     : 'Forma-Nodo',
38     'COMA'            : ',',
39     'PUNTO'           : '.',
40     'DPUNTOS'         : ':',
41     'CORI'            : '[',
42     'CORD'            : ']',
43     'LLAVEI'          : '{',
44     'LLAVED'          : '}'
```

**Se crearon tokens que servirán para la identificación de las operaciones que podrían venir en el archivo de prueba.**

```
def instruccion(cadena):
    global n_linea
    global n_columna
    global lista_lexemas
    lexema = ''
    puntero = 0

    while cadena:
        char = cadena[puntero]
        puntero += 1

        if char == '\\':
            lexema, cadena = armar_lexema(cadena[puntero:])
            if lexema and cadena:
                n_columna += 1

                l = Lexema(lexema, n_linea, n_columna)

                lista_lexemas.append(l)
                n_columna += len(lexema) + 1
                puntero = 0

        elif char.isdigit():
            token, cadena = armar_numero(cadena)
            if token and cadena:
                n_columna += 1

                n = Numero(token, n_linea, n_columna)
```

En la clase instrucciones. Se lleva a cabo la lectura de las posiciones dentro del archivo de carga y se dictan las instrucciones de la manera que deseamos que el programa ejecute.

```
def armar_numero(cadena):  
    numero = ''  
    puntero = ''  
    is_decimal = False  
    for char in cadena:  
        puntero += char  
        if char == '.':  
            is_decimal = True  
        if char == '"' or char == ' ' or char == '\n' or char == ']' or char == '\t':  
            if is_decimal:  
                return float(numero), cadena[len(puntero):]  
            else:  
                return int(numero), cadena[len(puntero):]  
        else:  
            numero += char  
    return None, None
```

**En armar número el programa detecta los caracteres que van antes y después de un número. Para que pueda reconocerlos.**



```

def operar():

    global lista_lexemas
    global instrucciones
    operacion = ''
    n1 = ''
    n2 = ''

    while lista_lexemas:
        lexema = lista_lexemas.pop(0)
        if lexema.operar(None) == 'Operacion':
            operacion = lista_lexemas.pop(0)
        elif lexema.operar(None) == 'Valor1':
            n1 = lista_lexemas.pop(0)

            if n1.operar(None) == '[':
                n1 = operar()

        elif lexema.operar(None) == 'Valor2':
            n2 = lista_lexemas.pop(0)
            if n2.operar(None) == '[':
                n2 = operar()

        if operacion and n1 and n2:
            return Arimetica(n1,n2,operacion, f'Inicio:{operacion.getFila()}'
        elif operacion and n1 and operacion.operar(None) == ('Seno' or 'Coseno'):
            return Trigonometricas(n1,operacion,f'Inicio:{operacion.getFila()}'
    return None

```

En la función operar se realizará la validación de las operaciones que fueron ingresadas, para verificar si es válido o no realizarlas.

```
def operar_():  
    global instrucciones  
    while True:  
        operacion = operar()  
        if operacion:  
            instrucciones.append(operacion)  
        else:  
            break  
    for instruccion in instrucciones:  
        print(instruccion.operar(None))  
    return instrucciones
```

**En la función `operar_()`, se realiza la ejecución de la operación que se requiere.**



```
def buscador():  
    filename=filedialog.askopenfilename(filetype="/", title="Select  
    if filename!="":  
        global contenido  
        archi1=open(filename, "r")  
        contenido=archi1.read()  
        global entrada  
        entrada=contenido  
        archi1.close()  
        scrolledtext1.delete("1.0", tk.END)  
        scrolledtext1.insert("1.0", contenido)  
    global archivo  
    archivo=filename  
global filename
```

**La función buscador nos permite recorrer los archivos de la computadora en busca del que necesitamos para nuestro proyecto.**

```
def guardar_como():  
    filename=filedialog.asksaveasfilename(initialdir="c:/pythonya",title="Guardar  
    if filename!="":  
        archi1=open(filename,"w")  
        archi1.write(scrolledtext1.get("1.0", tk.END))  
        archi1.close()  
        mb.showinfo("Información", "El archivo ha sido guardado con éxito.")  
        global documento  
        documento=archi1
```

La función `guardar_como()` nos permite guardar el archivo editado en una nueva ubicación o guardar en la computadora un nuevo archivo creado.

```
def guardar():  
    documento=open(archivo,"w")  
    documento.write(scrolledtext1.get("1.0", tk.END))  
    documento.close()  
    mb.showinfo("Información", "Información guardada con éxito.")
```

**La función guardar nos permite guardar el archivo en ejecución con los cambios realizados.**

```
def manualusuario():  
    wb.open_new(r"C:\Users\JUSTO\Desktop\Proyecto\Manual usuario.pdf")  
  
def manualtecnico():  
    wb.open_new(r"C:\Users\JUSTO\Desktop\Proyecto\Manual tecnico.pdf")
```

**Las funciones de los manuales nos permiten buscar el manual que queremos dentro de la computadora y ejecutarlo.**

```
0
1  ventanaprincipal=Tk()
2  ventanaprincipal.title("Ventana principal")
3  ventanaprincipal.geometry("1300x500")
4  ventanaprincipal.resizable(0,0)
5  ventanaprincipal.config(background="purple")
6  scrolledtext1=st.ScrolledText(ventanaprincipal, width=80, height=20)
7  scrolledtext1.place(x=475, y=10, width=700, height=450)
8
9
0  botonexplorar=Button(ventanaprincipal, text="Abrir", command=buscador)
1  botonexplorar.pack()
2  botonexplorar.place(x=100, y=100, width=150, height=50)
3
4  botonguardar=Button(ventanaprincipal, text="Guardar", command=guardar)
5  botonguardar.pack()
6  botonguardar.place(x=100, y=150, width=150, height=50)
7
8
9  botonguardarcomo=Button(ventanaprincipal, text="Guardar Como", command=guardar_como)
0  botonguardarcomo.pack()
1  botonguardarcomo.place(x=100, y=200, width=150, height=50)
2
```

**Las ventanas y botones de la pantalla inicial creados mediante la librería tkinter. Para un fácil uso de estos.**

```
ct / lexema.py / ...  
from Abstract.abstract import Expression  
from math import *  
  
class Lexema(Expression):  
    def __init__(self, lexema, fila, columna):  
        self.lexema = lexema  
        super().__init__(fila, columna)  
  
    def operar(self, arbol):  
        return self.lexema  
    def getFila(self):  
        return super().getFila()  
    def getColumna(self):  
        return super().getColumna()
```

La clase lexema nos ayuda a recorrer el archivo de prueba y armar los lexemas encontrados.

```
def operar(self, arbol):
    leftValue = ''
    rightValue = ''
    if self.tipo.operar(arbol) != None:
        leftValue = self.left.operar(arbol)
        rightValue = self.right.operar(arbol)
    elif self.right != None:
        rightValue = self.right.operar(arbol)

    if self.tipo.operar(arbol) == 'Suma':
        return leftValue + rightValue
    elif self.tipo.operar(arbol) == 'Resta':
        return leftValue - rightValue
    elif self.tipo.operar(arbol) == 'Multiplicacion':
        return leftValue * rightValue
    elif self.tipo.operar(arbol) == 'Division':
        return leftValue / rightValue
    elif self.tipo.operar(arbol) == 'Mod':
        return leftValue % rightValue
    elif self.tipo.operar(arbol) == 'Potencia':
        return leftValue ** rightValue
    elif self.tipo.operar(arbol) == "Inverso":
        return 1/leftValue
    elif self.tipo.operar(arbol) == "Raiz":
        return sqrt(leftValue)
    else:
        return None

def getFila(self):
    return super().getFila()

def getColumna(self):
    return super().getColumna()
```

En la clase aritméticas se realizan todas las operaciones aritméticas tales como la suma, resta, multiplicación. División, potencia, raíz e inverso.



```
ss Trigonometricas(Expression):  
    def __init__(self, left, tipo, fila, columna):  
        self.left = left  
        self.tipo = tipo  
        super().__init__(fila, columna)  
  
    def operar(self, arbol):  
        leftValue = ''  
        if self.left != None:  
            leftValue = self.left.operar(arbol)  
  
        if self.tipo.operar(arbol) == 'Seno':  
            return sin(leftValue)  
        elif self.tipo.operar(arbol) == 'Coseno':  
            return cos(leftValue)  
        elif self.tipo.operar(arbol) == 'Tangente':  
            return tan(leftValue)  
        else:  
            return None  
  
    def getFila(self):  
        return super().getFila()  
    def getColumna(self):  
        return super().getColumna()
```

En la clase trigonométricas se realizan todas las operaciones aritméticas tales como el seno, coseno y tangente.

