- **Facultad de Ingeniería**
- **Escuela de Ciencias y Sistemas**
- **Lenguajes Formales y de Programación, 2er. Semestre 2022.**

Manual Técnico

PROYECTO 1
LENGUAJES FORMALES Y DE PROGRAMACION

GENESIS NAHOMI APARICIO ACAN

carne :20211329

Descripcion de la practica

Este programa fue desarrollado con el lenguaje de programacion Python , el objetivo principal de este programa fue el crear un analizador lexico por medio de Tokens. El cual cumple con las reglas establecidas, manejando la lectura y escritura de archivos para el manejo de la información. A través de un entorno gráfico.

En este programa se usaron distintas librerías y se realizó con diferentes métodos La librería principal fue Tkinter, debido a que se tenía que hacer interfaz gráfica

¿paradigma que se utilizo?

para esta practica se utilizo mayormente POO (Programacion orientado a objetos

logica del programa

se utilizaron tresmodulos interfas ,analizador,perfiles y main para realizar el programa . A continuación se presenta una breve descripción de las clases, métodos y funciones que fueron implementadas para la realización del programa :

- main (modulo donde inicia el programa)
- interfas (modulo donde se encuentra la clase Interfaz la cual contiene la parte visual de lo que conlleva el sistema)
- analizador(modulo que contiene el funcionamiento interno del programa)
- perfiles (modulo que tiene el funcionamiento de las operaciones a resolver)

Main

Este modulo es donde se manda a llamar al modulo interfas y a su clase Interfaz para que el programa comienze a funcionar se usa para ejecutar algún código solo si el archivo se ejecutó directamente y no se importó.

```
if __name__ == "__main__":
    Interfaz()
```

INTERFAZ

En este modulo es donde se crea la clase interfaz utilizando la libreria de Tkinter de python , que representa la interfas que esta conformada por una ventana de menu principal la cual muestra el inicio del programa para que el usuario carge archivos con la extencion csv que contengan cursos , lea estos y los modifique .

```
from tkinter import *
from tkinter import filedialog
from tkinter import Scrollbar as scroll
import webbrowser
from analizador import Analizador
from tkinter import END, messagebox
import os

class Interfaz:
    global analizar2
    analizar2=Analizador()
    def __init__(self):
        self.archivo=None

        self.menu_inicio()

    def menu_inicio(self):···

    #-------------------------------METODOS PARA ARCHIVO ----------------------
    def abri(self):···

    def analizado(self):···

#guardar
    def guardar(self):···

#guardar como
    def guardarComo(self):···
    #def mostrararchivo(self):

#-----------------------------------METODOS PARA AYUDA-------------------------
    def manual_usuario(self):···

    def manual_tecnico(self):···

    def ayuda(self):···



    def salir(self):···
```

Clase Interfaz

la clase Cuerpo de este modulo consta de una variable global y de aproximadamente 10 metodos siendo estos los cuales conforman toda la parte visual con la que el usuario puede interactuar.Cabe recalcar desde antes que la unica extencion que permite este programa para los archivos de entrada es .txt
los metodos que contiene esta clase son :

- **__init__**

es el metodo del constructor el cual contiene al metodo de menu de inicio y una variable global

```
def __init__(self):
    self.archivo=None

    self.menu_inicio()
```

- **menu_inicio**

este metodo contiene el menu de inicio , al igual que varios labels y varios botones que lo conforma , ademas de un area te texto donde se visualizara el contenido de el archivo que el usuario abra

```
def menu_inicio(self):
    self.ventana = Tk()
    self.ventana.title("proyecto1")
    self.ventana.geometry("600x330")
    self.ventana.resizable(0,0)
frame archivo
    f1=Frame(self.ventana,bg='#a1f5f1').place(x=0,y=0,width=200,height=200)

    label1=Label(f1,text="Archivo",bg='#1ab3ac',font=('Cooper Black', 12),fg = '#a1f5f1').place(x=0,y=0,width=200,height=30.33)
    Abrir= Button(f1, text="Abrir",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12),command=self.abri).place(x=0, y=30, width=200, height=30)
    guardar= Button(f1, text="Guardar",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12),command=self.guardar).place(x=0, y=60, width=200, heigh
    guardar_como = Button(f1, text="Guardar como",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12),command=self.guardarComo).place(x=0, y=90, w
    Analizar= Button(f1, text="Analizar",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12),command=self.analizado).place(x=0, y=120, width=200,
    Errores= Button(f1, text="Errores",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12)).place(x=0, y=150, width=200, height=30)
    salir= Button(f1, text="Salir",bg= "#a1f5f1",fg='#1ab3ac',font=('Cooper Black', 12),command=self.salir).place(x=0, y=180, width=200, height=30)
```

- **abri()**

La funcion de este metodo es el abrir el archivo con extencion txt y leerlo, cuardando el contenido del archivo en una variable

- **analizando()**

este metodo toma la informacion del archivo abierto y lo manda a el modulo analizar para su respectivo analisis

- **guardar()**

Este método guarda el archivo modificado o no con el mismo nombre del original.

- **guardarcomo()**

Este método guarda el archivo con un nuevo nombre

- **Manual_tecnico()**

muestra en el sistema el presente manual en formato pdf

- **manual_usuario()**

muestra en el sistema el manual de usuario en formato pdf

- **Ayuda**

muestra la informacion de el desarollador en el navegador predeterminado

- **salir()**

Este metodo hace que el programa termine automaticamente

ANALIZADOR

En este modulo es donde se encuentra la mayor parte de la logica del programa, donde se desarrolla el analisis del archivo que es abierto, separando y analizando linea por linea el texto que contiene , al igual que las operaciones aritmeticas son realizad para posteriormente mostrarlas en pantalla en formato html. Consta de tres clases y varios metodos

```
class Analizador:
    global  operandopy2,aritmetica,operacionesarit
    operandopy2=OPerandopy()

    operacionesarit=[]

    def __init__(self):···

#quito loa < > de la cadena
    def quitar(self,_cadena:str,_num:int):···

#aumentando las lineas de el archivo de entrada
    def aumentandolineas(self):  ···

    def etiqueta(self,_cadena:str,_etiqueta:str):···

    def Numero(self,_cadena:str):          ···

    def importaroperaciones (self):···

    def importacion2(self):···

#operadores
    def operando (self,_cadena:str):···

    def operacionAri(self):···

    def tipo(self,_cadena:str):···

    def texti(self,_cadena:str):···

    def titulo(self,_cadena:str):···

    def Descrip(self,_cadena:str):···

    def contenid(self,_cadena:str):···

    def funci(self,_cadena:str):···

    def titulo2(self,_cadena:str):···

    def Descripcion2(self,_cadena:str):···
```

Clase Tokens

la clase Tokens de este modulo consta de varios tokens(en total 33 tokens) los cuales son utiles para la lectura del el archivo de entrada, hace uso de la libreria Enum para ello , cada token tiene asignado un valor en espesifico el cual va desde un nombre , numeros , signos y hasta texto

```
class Token(Enum):
    Tk_menor="<"
    Tk_mayor=">"
    Tk_E_numero="Numero"
    Tk_Numero="[0.0-9.0]*"
    Tk_pleca="/"
    Tk_OPERACION="Operacion"
    Tk_igual="="
    Tk_Tipo="Tipo"
    TK_Suma="SUMA"
    Tk_Resta="RESTA"
    Tk_Multiplicacion="MULTIPLICACION"
    Tk_Division="DIVISION"
    Tk_Potencia="POTENCIA"
    TK_RAIZ="RAIZ"
    TK_Inverso="INVERSO"
    TK_Seno="SENO"
    TK_Coseno="COSENO"
    TK_Tangente="TANGENTE"
    TK_Mod="MOD"
#titulo segunda etiqueta
    TK_Texto="[a-zA-Z,À-ÿ\u00f1\u00d1,'+'-'-','*',0.0-9.0*,':','%','=','/','^','√','(',')'*]*"
    TK_TextoPrincipal="Texto"
    TK_Operaciones="[a-zA-Z,À-ÿ\u00f1\u00d1,'+'-'-','*',0.0-9.0*,':','%','=','/','^','√','(',')'*]*"
    Tk_Titulo="Titulo"
    Tk_Descripcion="Descripcion"
    TK_DESCRIPTION2  = "\[TEXTO\]"
    Tk_Contenido="Contenido"
    Tk_Funcion="Funcion"
    Tk_ESCRIBIR="ESCRIBIR"
    TK_CONTENIDO2  = "\[TIPO\]"
    Tk_Color="Color"
    Tk_color2='[AZUL,ROJO,VERDE,AMARILLO,NEGRO,NARANJA,MORADO,ROSADO]*'
    Tk_tamano="Tamanio"
    TK_Estilo="Estilo"
#analisador lexico
#
```

Clase Errores_lexicos

Esta clase pertenece a el modulo de analizador, consta de un metodo constructor y el metodo getErrores, su funcion principal es el optener los errores lexticos que pueda contener el archivo de entrada y guardarlos para mostrarlos luego en pantalla para esto se apoya de el retun que se encuentra en el metodo getErrores()

```
class Errorres_lexicos:
    def __init__(self,_tokem,_filas,_columnas,descripcion="") :
        self.tokem=_tokem
        self.filas=_filas
        self.columnas=_columnas
        self.descripcion=descripcion

    def getErrores(self):
        return {"tokem":self.tokem,"filas":self.filas,"columnas":self.columnas,"descrip
```

Clase Analizador

Esta clase pertenece a el modulo de analizador, consta de varios metodos , tres variables globales siendo una lista , su funcion principal es el leer el archivo de entrada del usuario parte por parte deste su primer token hasta el ultimo

- **__init__()**

EStе metodo es el constructor de la clase anallizador contiene 6 listas donde se almacenan los datos que el programa va recolectando a lo largo de su ejecucion, dos valiables sin poseer valor alguno, dos string y dos int que contendran el numero de filas y columnas que se vallan creando a medida que se lee el archivo segun los tokens

- **__init__()**

ESte metodo es el constructor de la clase anallizador contiene 6 listas donde se almacenan los datos que el programa va recolectando a lo largo de su ejecucion, dos valiables sin poseer valor alguno, dos string y dos int que contendran el numero de filas y columnas que se vallan creando a medida que se lee el archivo segun los tokens

- **quitar()**

Este método quita las lineas que se van leyendo a medida que se ejecuta la clase

```
def quitar(self,_cadena:str,_num:int):
    _tmp=""
    contador1=0
    for i in _cadena:
        if contador1>=_num:
            _tmp +=i
        else:
            self.tmp_cadena+=i
        contador1+=1
    return _tmp
```

- **aumentandolineas()**

Este método lee las lineas siguientes aumentando las filas

- **importaoperaciones ()**

este metodo importa envia los valores de las operaciones a otro modulo el cual realiza las operaciones segun sea su valor.

- **Numero()**

Este método interpreta los tokens por orden comenzando por Token.Tk_menor.value,(<) y terminando con Token.Tk_mayor.value(>) , el método es para la etiqueta Números, en este método se evalúan los numeros con la operacion obtenida en el metodo Operacion.guardando los numeros optenidos en unalista para usarlos luego al operar estos

- **Operando()**

Este metodo interpreta los tokens por onden al igual que numero empezando y terminando con los mismos , este metodo evaluan las etiquetas de operaciones que contiene el archivo de entrada.
ejemplo : operando=RESTA

- **Tipo()**

Este metodo interpreta todos los tokens por orden que se encuentran dentro de la etiqueta tipo,este metodo hace que los metodos numero() y Operando() se ejecuten dentro de el ya que el archivo de entrada contiene estas etiquetas dentro de la etiqueta tipo

```
def tipo(self,_cadena:str):
    tokens=[···
    _numero=""
    for i in tokens:
        try :
            if "operacion" == i:···

            else:
                jefe=re.compile(f'^{i}')
                s=jefe.search(_cadena)
                print("|",self.linea,"|", self.columna,"|",s.group())
                self.columna+=int(s.end())
                _cadena=self.quitar(_cadena,s.end())
            self.aumentandolineas()

        except:
            print('hay un erroren tipo ')
            return{'result': _numero,'cadena':_cadena,"Error":True}

    return{'result': _numero,'cadena':_cadena,"Error":False}
```

- **texti()**

Este metodo lee la etiqueta de texto que se encuentra despues de la etiqueta tipo , esta etiqueta contiene el token Token.TK_Texto el cual valida letras mayusculas y minusculas de la 'A' a la 'Z', numeros, signos y caracteres especiales de las letras

- **Descrip()**

Este metodo lee la etiqueta lee todo lo que se encuentra en la etiqueta Descripcion del archivo de entrada de la misma manera que texti

- **Contenid()**

Este metodo lee lo que se encuentra en la etiqueta de contenido del archivo de entrada de la misma forma que texti y Descrip

- **Funci**

Este metodo hace lo mismo que tipo donde interpreta los tokens que se encuentran en el archivo y relaciona cada uno de elloa con sus metodos correspondientes los cuales analizan los tokens que son ingresados respectivamente a ellos

```
tokens=[···
_descrip=""
for i in tokens:
    try:
        if "TITULO" == i:
            if self.etiqueta(_cadena,"<Titulo>"):···

            else:
                print("error en el numero ")
                return{'result': _descrip,'cadena':_cadena,"Error":True}

        elif "DESCRIPCION" == i:···

        elif "CONTENIDO" == i:
            if self.etiqueta(_cadena,"<Contenido>"):
                _result=self.contenid(_cadena)
                _cadena=_result['cadena']
                if _result["Error"]:
                    print('ocurrio un en el resultado')
```

- **Titulo2()**

Este método interpreta los tokens que son correspondientes a la segunda etiqueta titulo , la cual es la responsable de optener los colores y el tamaño que tendra el titulo en la salida del html.

- **Descripcion2()**

Este método interpreta los tokens que son correspondientes a la segunda etiqueta descripcion que se encuentra dentro de la etiqueta funcion , la cual es la responsable de optener los colores y el tamaño que tendra el la descripcion(el segundo texto)en la salida del html, lee cada color ingresado y cambia el nombre de este a su correspondiente en ingles , al igual que valua el tamaño de este guardandolo

- **Contenido2()**

su funcion es la misma que el metodo descripcion2 y Titulo 2, optener colores y tamaño atravez de sus tokens correspondientes

```
def Contenido2(self,_cadena:str):
    Tokene=[
        Token.Tk_menor.value,#<
        Token.Tk_Contenido.value,#titulo
        Token.Tk_Color.value,
        Token.Tk_igual.value,
        Token.Tk_color2.value,
        Token.Tk_tamano.value,
        Token.Tk_igual.value,
        Token.Tk_Numero.value,
        Token.Tk_pleca.value,
        Token.Tk_mayor.value,#>
    ]
    _titulo=""
    _tamanoCC=""
    self.colorCo=None
    self.tamanoCC=None
    for i in Tokene:
        try:
            jefe=re.compile(f'^{i}')
            s=jefe.search(_cadena)
            print("|",self.linea,"|", self.columna,"|",s.group())
            self.columna+=int(s.end())
            if i == Token.Tk_color2.value:···
            if i==Token.Tk_Numero.value:
```

- **estilo()**

al igual que tipo reune los matodos anteriores de titulo2 , descrpcion2 y contenido 2 , interpretando los tokens que se encuentran en este metodo.

- Todo()

Este metodo reune todos los tokens del archivo de entrada y ejecuta los metodos a los cuales correspondientes a las etiquetas proncipales tipo, texto, funcion y estilo

```
def todo(self,_cadena:str):
    tokens=[
        "TIPO",
        "TEXTO",
        "FUNCION",
        "ESTILO"
    ]
    _descrip=""
    for i in tokens:
        try:
            if "TIPO" == i:···

            if "TEXTO" == i:···

            elif "FUNCION" == i:···

            elif "ESTILO" == i:
                salida=True
                if salida:
                    print('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>')
                    _result=self.estilo(_cadena)
                    _cadena=_result['cadena']
                    if _result["Error"]:
                        print('hay un error dentro deloperadorw y tipo')
                        salida=False
                        #break
                    if self.etiqueta(_cadena, "</Funcion>"):
                        salida=False

        except:
            print('hay un erroren tipo ')
            return{'result': _descrip,'cadena':_cadena,"Error":True}
```

- **Complilar()**

Este metodo es el cual es importado al modulo de interfaz para su ejecucion , es el encargado de leer el archivo, guardar esa informacion y remplazar los espacios y saltos de linea que se presenten en el , al igual es donde se ejecutan los otros metodos para el funcionamiento de el programa ademas de ser en donde se genera el html de salida una vez leido el archivo y realizadas la operaciones

```
def compilar(self):
    xml="ejemplo.txt"
    archivo=open(xml,"r",encoding="utf-8")
    contenido_archivo=archivo.readlines()
    archivo.close()
    una_cadena=""
    lista_cadena=[]
    nueval=[]
    otral=[]
    for i in contenido_archivo:···
    for l in contenido_archivo:···
    self.lista_cadena=lista_cadena
    print(self.todo(una_cadena))
    #print(self.listatodo)
    self.importaroperaciones()
    Q=operandopy2.juan()
    self.operacionAri()
    if self.colorTT!=None and self.tamano!=None and self.colord!=None and self.tamanoD!=None a

        f = open('RESULTADOS_202113293.html','w',encoding="utf-8")
        salidaA = """<html>\n<head></head><body bgcolor="#f4cffe">\n<h1 align="center" style="
        salidaA +=str(self.colorTT)
        salidaA +="""  ;font-size:"""
        salidaA +=str(self.tamano)
        salidaA+="""px">"""
        salidaA += str(self.aqui_titulo).replace("[","").replace("]","").replace("'","")
        salidaA +="</font></h1>\n"
        r=0
        for x in range(len(otral)):···
        salidaA+="""<p align="center" style="color:#691d92;font-size:20px">\n"""
        salidaA+='operaciones resueltas '
        salidaA+="</font></p>\n"
        for y in range(len(Q)):
            salidaA+="""<p style="color:"""
            salidaA +=str(self.colorCo)
            salidaA +="""  ;font-size:"""
            salidaA +=str(self.tamanoCC)
            salidaA+="""px">"""
            salidaA+=str(Q[y]).replace("[","").replace("]","").replace("'","")
            salidaA+="</font></p>\n"
        salidaA+="</body>\n </html>"
        f.write(salidaA)
        f.close()
```

PERFILES

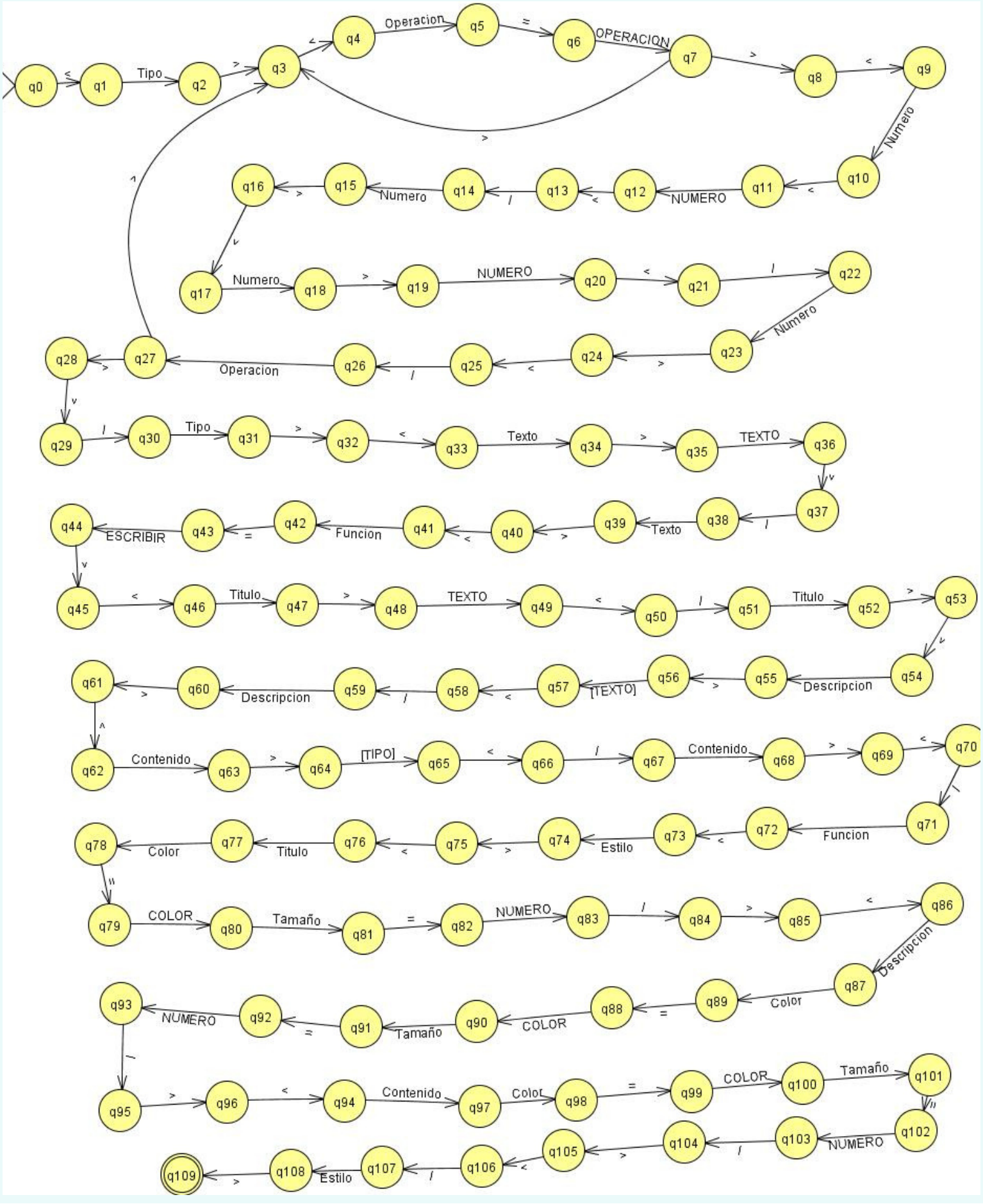
En este modulo es donde se crea la clase OPerandopy,la cual consta de dos metodos, el metodo constructor y el metodo operation, este ultimo es el encargado de hacer las operaciones aritmeticas de el programa , usando listas nativas , desde suma hasta divicion y desde seno hasta tangente operando cada una de ellas , utilizando la libreria math para realizar la soperaciones de seno , coseno y tangente

```python
import math
class OPerandopy:
    def __init__(self) -> None:
        pass

    def operation(self,paso_lista):
        self.listatodo=paso_lista
        unalista=[]
        cafelista=[]
        otralista=[]
        self.listat=[]

        for i in range(len(self.listatodo)):
            listamientras1=int(len(self.listatodo[i]))-3
            listaM2=int(len(self.listatodo[i]))-2
            listaM3=int(len(self.listatodo[i]))-1

            sumando_cont=0
            resta_cont=0
            multiplicacion_cont=0
            raiz_maiz_cont=0
            division_cont=0
            potenciando_cont=0
            residuo_contando=0
        #suma
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'

        #resta
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'
    #MULTIPLICACIO
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'

    #RAIZ
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'


    #DIVISION
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'

#POTENCIA
            if self.listatodo[i][listaM2]!='tangente'and self.listatodo[i][listaM2]!='seno'

    #MODULO
```

AFD

Esta es una muestra gráfica del AFD con el orden de nuestro programa y el uso de los tokens

- **OPERACION=SUMA,RESTA,MULTIPLICACION,DIVICION,POTENCIA,RAIZ,INVERSO,RAIZ,SENO,COSENO, TANGENTE,MOD]**
- **TEXTO=[a-zA-Z,À-ÿ\u00f1\u00d1,'+'-'-','*',0.0-9.0*,':','%','=','/','^','√',(',')'*]*"**
- **NUMERO=[0.0-9.0]**
- **COLOR=[AZUL,ROJO,VERDE,AMARILLO,NEGRO,NARANJA,MORADO,ROSADO]***



- Universidad de San Carlos de Guatemala
- Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
- Lenguajes Formales y de Programación, 2er. Semestre 2022.

GENESIS NAHOMI APARICIO ACAN

carne :20211329