Universidad de San Carlos de Guatemala -USAC-

Facultad de Ingeniería

Ingeniería en Ciencias y Sistemas

Ing. Otto Amilcar Rodríguez Acosta

Aux. Carlos Esteban Godínez Delgado

Lenguajes Formales y de Programación A+

PROYECTO No. 1
ANALIZADOR LÉXICO
(Manual Técnico)

Robin Omar Buezo Díaz

Carné 201944994

Domingo 25 de septiembre de 2022.

El presente documento tiene como finalidad mostrar al usuario la funcionalidad y construcción del software para que entienda su creación y pueda dar solución a cualquier error que pueda presentarse.

Se explican el flujo y las diferentes partes que constituyen el software y como debemos de interactuar con este para que el sistema nos sea de gran ayuda.

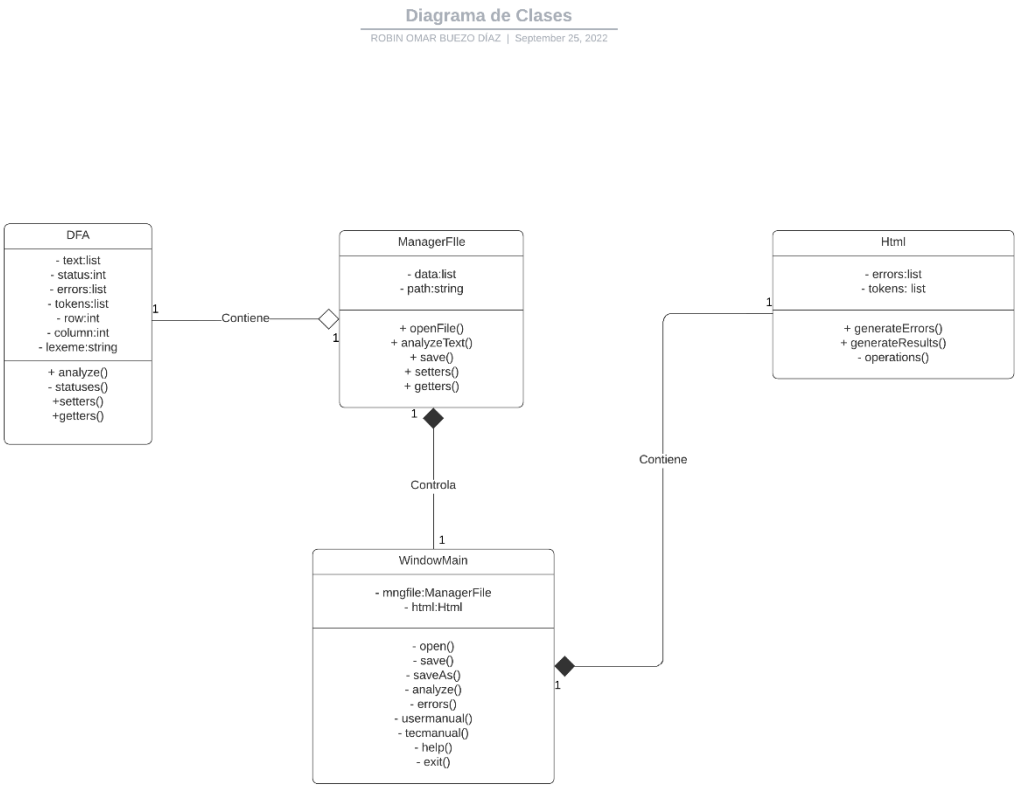
PARADIGMA DE PROGRAMACIÓN

Para la creación se este software se utilizó el paradigma de Programación Orientada a Objetos, ya que esto da una mejor facilidad a la hora de manejar el archivo que se está manipulando a lo largo de toda la ejecución, como también el poder encapsular los objetos y luego poder utilizar los mismos objetos en las diferentes ventanas que se utilizan a lo largo del programa.

NOMENCLATURA

Identificador	Regla	Ejemplo
Clases	Sustantivos, primera letra mayúscula	class MyClass
Métodos	Verbos, primera letra minúscula y la del resto de palabras en mayúscula	ejecutar() cargarArchivo()
Variables	En minúsculas y deben empezar con una letra	variableuno variabledos

DIAGRAMA DE CLASES



TOKENS

Proyecto 1

Tokens

1 = Apertura = <(/|ε)

2 = Palabra = LL*

3 = Igual = =

4 = Numero = DD*(.DD*)

5 = Cierre = (/|ε)>

6 = Coma = ,

7 = Punto = .

8 = CorcheteA = [

9 = CorcheteC =]

10 = Suma = SUMA

11 = Resta = RESTA

12 = Multiplicacion = MULTIPLICACION

13 = Division = DIVISION

14 = Potencia = POTENCIA

15 = Raiz = RAIZ

16 = Inverso = INVERSO

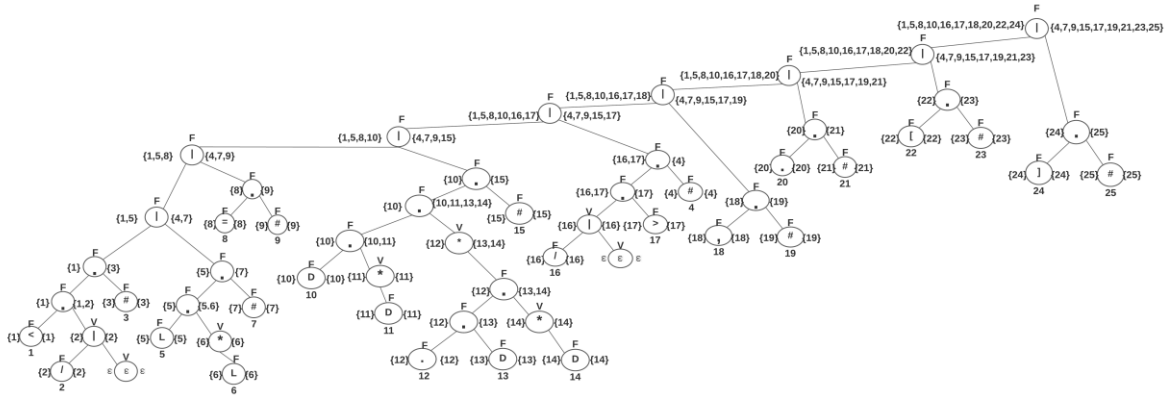
17 = Seno = SENO

18 = Coseno = COSENO

19 = Tangente = TANGENTE

20 = Modulo = MOD

CONSTRUCCIÓN DE AFD



Nodo I	Sig(I)
1,<	2,3
2,/	3
3,#	
4,#	
5,L	6,7
6,L	6,7
7,#	
8,=	9
9,#	
10,D	11,12,15
11,D	11,12,15
12,.	13,14
13,D	14,15
14,D	14,15
15,#	
16,/	17
17,>	4
18,.	19
19,#	
20,.	21
21,#	
22,[	23
23,#	
24,]	25
25,#	

S_0={1,5,8,10,16,17,18,20,22,24}={<,<,L,=,D,/,>,.,.,[,]}

Sig(<)=Sig(1)={2,3}=S_1

Sig(L)=Sig(5)={6,7}=S_3

Sig(=)=Sig(8)={9}=S_4

Sig(D)=Sig(10)={11,12,15}=S_6

Sig(/)=Sig(16)={17}=S_6

Sig(>)=Sig(17)={4}=S_{13}

Sig(,)=Sig(18)={19}=S_7

Sig(.)=Sig(20)={21}=S_8

Sig([)=Sig(22)={23}=S_9

Sig(])=Sig(24)={25}=S_{10}

S_1={2,3}={/,#}

Sig(/)=Sig(2)={3}=S_2

S_3={6,7}={L,#}

Sig(L)=Sig(6)={6,7}=S_3

S_5={11,12,15}={D,.,#}

Sig(D)=Sig(11)={11,12,15}=S_5

Sig(.)=Sig(12)={13,14}=S_{11}

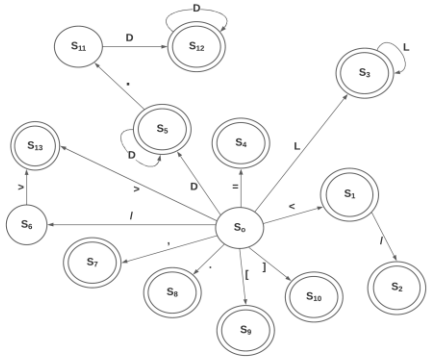
S_{11}={13,14}={D,D}

Sig(D)=Sig(13)={14,15}=S_{12}

Sig(D)=Sig(14)={14,15}=S_{12}

S_{12}={14,15}={D,#}

Sig(D)=Sig(14)={14,15}=S_{12}



METODOS PRINCIPALES

openFile:

Este método nos permite cargar el archivo que se quiere editar o manipular desde el programa.

analyzeText:

Este método se encarga de correr el AFD sobre el programa para poder determinar si el texto esta correctamente apegado al lenguaje y si es así poder ejecutar las instrucciones, también es el método que se encarga de encontrar los errores en nuestro código.

save:

Este método nos permite guardar nuestro código ya sea sobrescribiendo el mismo archivo o bien generando uno nuevo.

generateErrors:

Este método se encarga de generar nuestro archivo html de salida con los errores encontrados por el análisis del AFD.

generateResults:

Este método se encarga de generar nuestro archivo html de salida con los resultados generados mediante las instrucciones dadas por el código a nuestro programa.

operations:

Este método sirve de apoyo para los métodos ‘generateResults’ y ‘generateErrors’ ya que se encarga de depurar la información generada por nuestro AFD para luego ser trasladada al html.

HERRAMIENTAS

Para poder dar solución a los requerimientos anteriores se utilizó el lenguaje de programación Python versión 3 y su documentación por su versatilidad y fácil programación.

Como herramienta de programación se utilizó el programa Visual Studio Code por su amplia funcionalidad y herramientas que brinda a los programadores a la hora de programar en cualquier lenguaje.

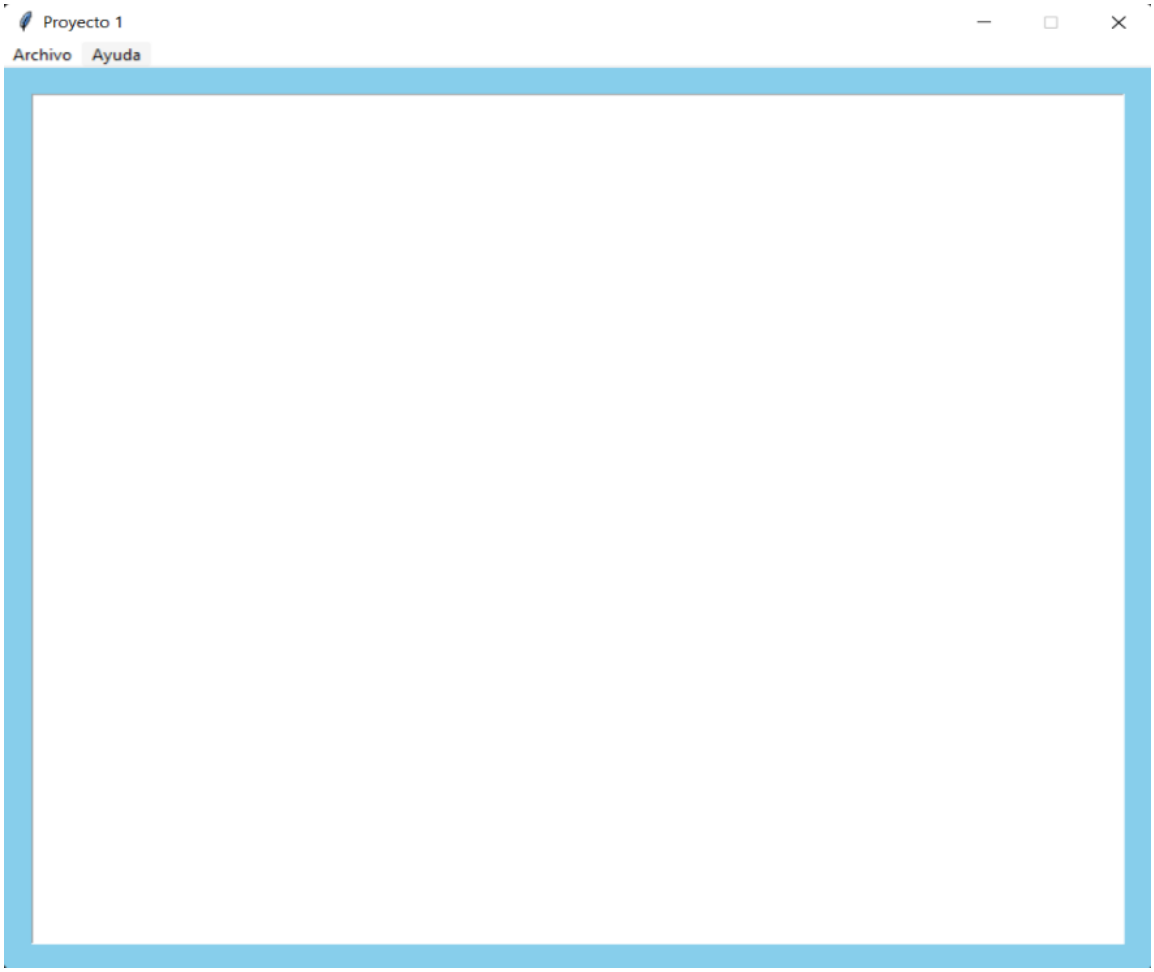
Para poder realizar los diagramas y el AFD se utilizó la herramienta Lucidchart por sus plantillas que nos facilitan el hacer diagramas más profesionales y entendibles.

Por último, se utilizó la herramienta de versionamiento GitHub. Para poder tener un mejor control sobre los cambios que se iban realizando en nuestro código y no tener el problema de perder funcionalidad si en caso algún cambio ocasionaba erros.

GUI PRINCIPALES

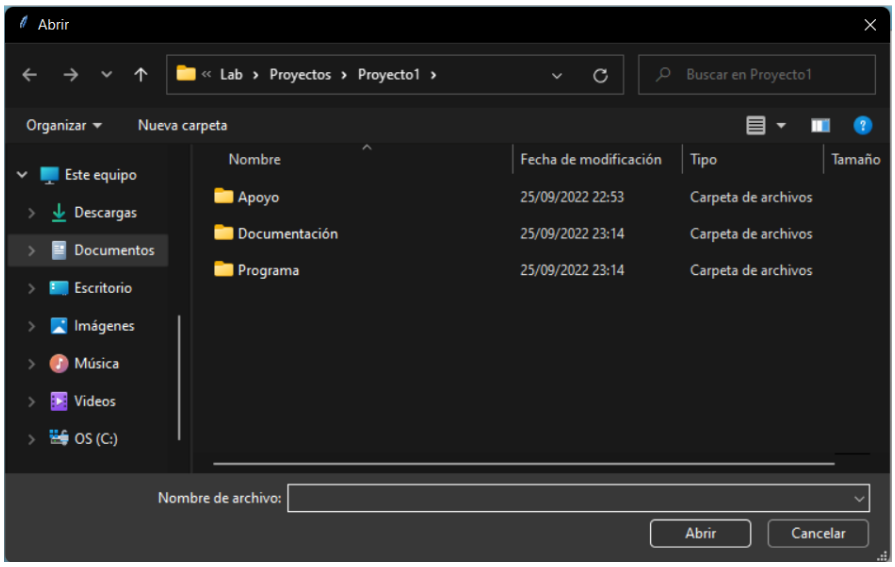
Practica 1 (Ventana Principal)

Esta es la ventana principal y en donde tendremos a nuestro editor de texto para poder manipular nuestro código, como también nuestra barra de menú para el resto de las opciones.



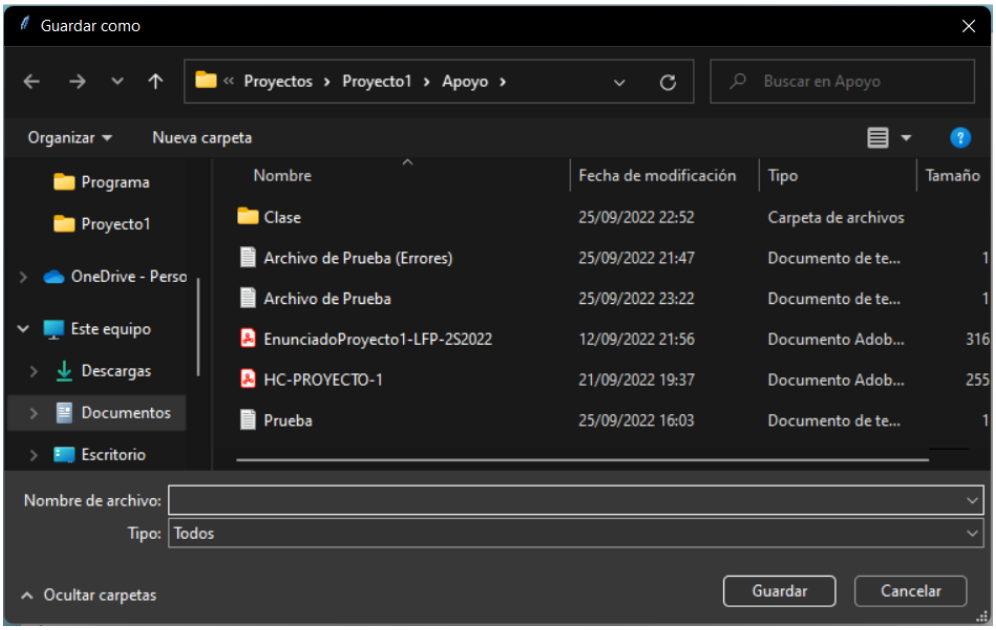
Abrir

Esta opción nos abrirá una ventana emergente desde donde podremos buscar y seleccionar el archivo que queremos cargar.



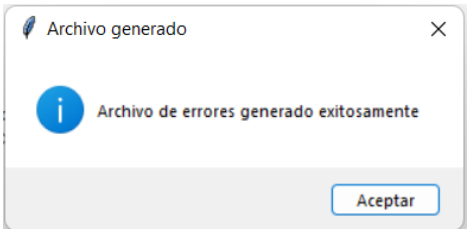
Guardar Como

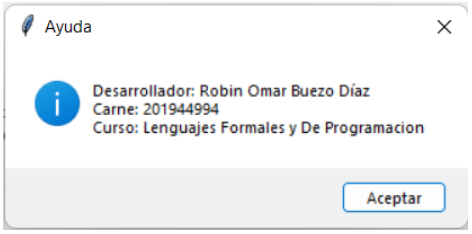
Este botón nos permite guardar nuestro archivo, pero creando otro archivo sin modificar el archivo cargado originalmente. Para ello también tendremos una ventana emergente.

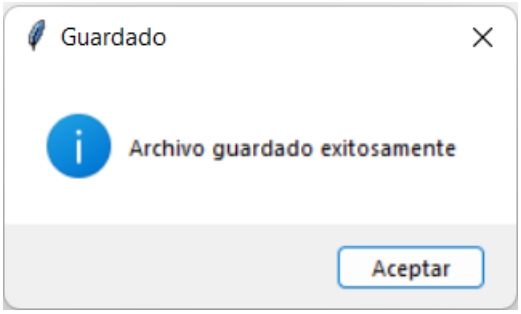


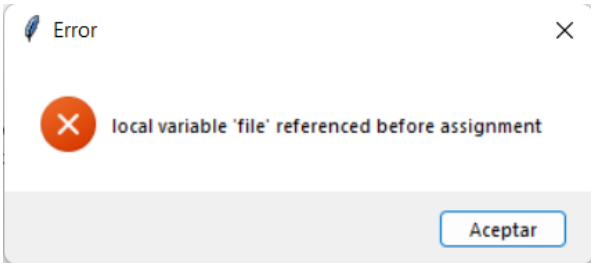
Cuadros de diálogo

Cabe destacar que nuestro sistema nos irá mostrando cuadros de diálogo a lo largo de la ejecución para poder darnos información o bien mostrarnos errores que puedan surgir.









GLOSARIO

HTML: Es un lenguaje de marcado que nos permite indicar la estructura de nuestro documento mediante etiquetas.

AFD: Un autómata finito determinista (abreviado AFD) es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.