Proyecto 2

Manual Técnico

2022

Lenguajes Formales y de Programación

Damaris Julizza Muralles Véliz

202100953

Detalles de desarrollo

Este software s e desarrollo en el lenguaje python, utilizando como interfaz gráfica la librería de Tkinter.

Descripción general

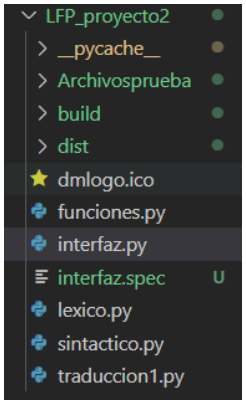
El programa es un software que permite representar las primeras fases de un compilador, el análisis léxico y el sintáctico.

Cuenta con distintas opciones para el manejo de información dentro de la interfaz y también contiene lógica separada para el analizador léxico y sintáctico los cuales se realizaron mediante los métodos correspondientes.

Lógica del programa

Recursos Utilizados:

En el folder “LFP_proyecto2” encontraremos todos los recursos y archivos utilizados, como lo son las imágenes usadas y los archivos de las clases utilizadas, estos ultimas son “funciones”, “interfaz”, “lexico”, “sintactico” y “traduccio1”.



Funciones:

```
class Funcion: ···
```

Este archivo contiene la clase “Funcion” la cual está compuesta por varios métodos que realizan cierta acción para la navegación entre la interfaz y el manejo de los datos que se ingresen en el software, entre estas funciones tenemos:

- ### Función nuevo:

En este método se espera recibir un atributo el cual será el texto contenido en el Text “caja de texto” de nuestra interfaz y antes de realizar su acción correspondiente se genera un mensaje con un messagebox para preguntar al usuario si desea guardar los datos que se tienen, en caso afirmativo esto llama a los métodos save y save_c según corresponda para luego limpiar todo el contenido en esta caja de texto.

➢ Función viewer:

Utiliza la librería FileDialog para navegar entre los archivos de nuestra computadora y seleccionar uno, generando como resultado la ruta en que este se encuentra.

Con la ruta obtenida se hace uso de la librería os para separar la ruta y obtener su nombre, además de abrir el archivo con la instrucción open() para leer el contenido y guardarlo en una variable que posteriormente se retorna a la clase principal para ser colocada como parámetro en la caja de texto.

➢ Función salve:

Esta función hará uso de la ruta obtenida al abrir el archivo y pasando esto como parámetro en open() junto con w que es la instrucción para escribir, se reescribirá el contenido de dicho archivo.

➢ Función salve_c:

Esta función FileDialog.asksaveasfile () para abrir una ventana de navegación de archivos que permitirá escribir nuevo nombre de nuestro archivo a guardar y la dirección o lugar en donde lo colocaremos.

➢ Función run:

Esta función llama a todas las clases correspondientes a los analizadores y para la traducción.

Estos fueron colocados de forma anidada mediante instrucciones if-else para validar si existe errores en cada análisis, de ser retornado un error se manda a llamar al triviewer que contiene la tabla en nuestra interfaz para posteriormente agregar la información concerniente mediante la instrucción insert()

➢ Función tokengenerate:

Se hace uso de una lista la cual se crea al realizar el analizador léxico el cual permite ver todos los tokens que fueron leídos y que lexema le corresponde , según la lógica propuesta esta solo se podrá visualizar si la compilación o los análisis fueron realizador sin ningún error en caso contrario mediante un menssagebox se informa al respecto.

Para visualizar esta lista de tokens se creó una nueva ventana que contiene un label para poner su titulo y un triviewer para crear la tabla.

lexico:

Aquí se encuentra la clase analizadorl el cual está compuesto por funciones que contienen toda la lógica para el análisis léxico.

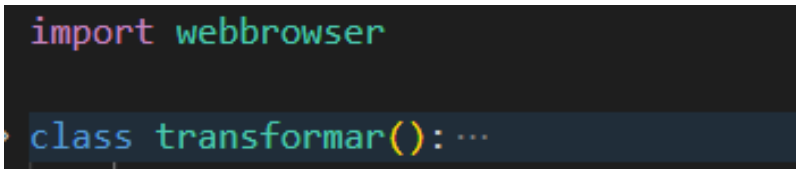
Con anterioridad a la realización de esta parte se realizó un análisis aplicando distintos métodos para obtener así los tokens y sus respectivas expresiones regulares para modelar así la lógica correspondiente.

sintactico:

Esta clase y sus funciones son llamadas desde el analizador léxico para continuar con el proceso de compilación en dado caso no existan errores en la primera parte.

Para la realización de esta lógica se hizo un análisis previo en donde se crearon los autómatas y diagramas de árbol correspondientes a cada caso, por lo tanto, la lectura de este análisis se realizó mediante estados.

traduccionl:



Este será llamado después de realizar el análisis sintáctico en caso de que no haya errores. Esta clase “transformar ” contiene la lógica que hace posible la traducción del texto a un lenguaje html y css, para esto se hace uso de sentencias if-else y ciclos for o while para realizar los cambios correspondientes y formar una cadena que posteriormente sera pasada a una función, “construirhtml” o “construircss”, que crearan un nuevo archivo con el mismo nombre del archivo de entrada, pero conteniendo los códigos de html y la estructura de css y sus correspondientes extensiones.

También se hizo uso de la librería webbrowser para abrir de manera inmediata el archivo html y visualizar la página creada.

interfaz:

La interfaz esta creada a partir de varias funciones que la modelan:

- **_main_:** que contiene todo el código que sera llamado primero.

- **Ventanaprincipal:** que modela toda nuestra interfaz gráfica, hace uso de triviewer, Text, label, frames, entre otros.

- **Actualizar_index:** esta función esta relacionada a la caja de texto y es la encargada de cambiar constantemente la posición del cursor de texto, es decir la posición en la que estemos.

- **Nuevo, abrir, guardar, guardar_como, generar, g_tokens:** estas son funciones que se encargan de hacer una conexión entre nuestra interfaz y nuestra clase función y sus métodos.

Tabla de tokens

A continuación, se muestra la tabla de todos los tokens con su respectiva expresión regular o patrón que lo define:

No	Token	Patron
1	Token_barra	(/)
2	Token_texto	(A-Za-z0-9 \# | \$ \& \? \¿ \" \% \+ = _ , - . : ; \(\) \[\] \^ \s) +
3	Token_asterisco	(*)
4	Token_mayor	>

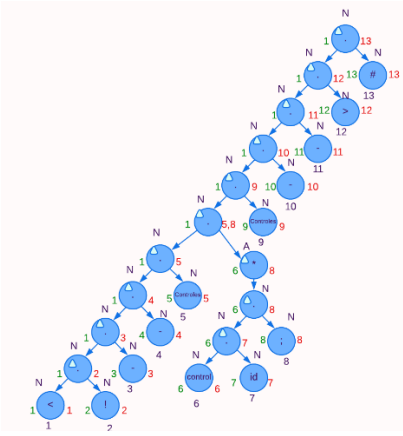
5	Token_menor	<
6	Token_signoE	!
7	Token_guion	-
8	Token_E_Control	(Controles|controles)
9	Token_E_Propiedad	(Propiedades|propiedades)
10	Token_E_Colocar	(Colocacion|colocacion)
11	Token_Control	(Etiqueta)|(Boton)|(Check)|(RadioBoton)|(Texto)|(AreaTexto)|(Clave)|(Contenedor)
12	Token_Id	(A-Za-z0-9_) +
13	Token_pcoma	;
14	Token_punto	.
15	Token_Propiedad	(setColorLetra)|(setTexto)|(setAlineacion)|(setColorFondo)|(setMarcada)|(setGrupo)|(setAncho)|(setAlto)
16	Token_pa	(
17	Token_pc	)
18	Token_coma	,
19	Token_comilla	"
20	Token_numero	pendiente ver lo del signo(0-9)+
21	Token_P_valor	(Centro) | (Izquierdo) | (Derecho)
22	Token_parametro	(true) | (false)
23	Token_elemento	(A-Za-z0-9_) + | (this)
24	Token_posicion	(setPosicion) | (add)

Método del árbol

La estructura a seguir para analizar las distintas partes se planteó de la siguiente forma:

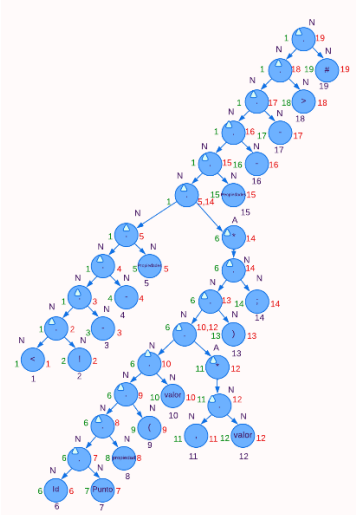
Expresion
< ! - - Controles control Id ; Controles - - >
< ! - - Propiedades Id . propiedad (valor [, valor]*) ; Propiedades - - >
< ! - - Colocacion Id . posicion (Dato [, Dato]?) ; Colocacion - - >

Por lo tanto, se presenta el método del árbol para cada sección indicada en la estructura del texto a ser analizado.



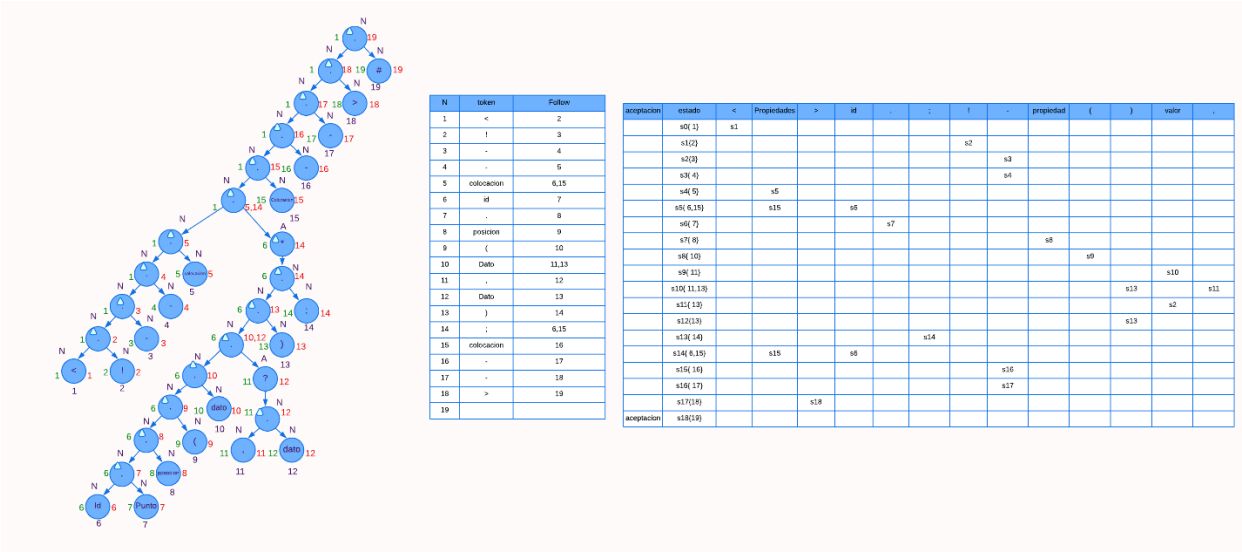
N	token	Follow
1	<	2
2	!	3
3	-	4
4	-	5
5	controles	6,9
6	control	7
7	id	8
8	;	6,9
9	Controles	10
10	-	11
11	-	12
12	>	13
13		

aceptacion	estado	<	Controles	>	control	id	;	!	-
	Q0(1)	Q1							
	Q1(2)							Q2	
	X2(3)								Q3
	Q3(4)								Q4
	Q4(5)		Q5						
	Q5(6,9)		Q8		Q6				
	Q6(7)					Q7			
	Q7(8)						Q5		
	Q8(10)								Q9
	Q9(11)								Q10
	Q10(12)			Q11					
aceptacion	Q11(13)								



N	token	Follow
1	<	2
2	!	3
3	-	4
4	-	5
5	propiedades	6,15
6	id	7
7	.	8
8	propiedad	9
9	(	10
10	valor	11,13
11	,	12
12	valor	11,13
13	)	14
14	;	6,15
15	propiedades	16
16	-	17
17	-	18
18	>	19
19		

aceptacion	estado	<	Propiedades	>	Id	.	;	!	-	propiedad	(	)	valor	,
	-0(1)	r1												
	r1(2)							r2						
	r2(3)								r3					
	r3(4)								r4					
	r4(5)		r5											
	r5(6,15)		r15		r6									
	r6(7)					r7								
	r7(8)									r8				
	r8(10)										r9			
	r9(11)												r10	
	r10(11,13)											r13		r11
	r11(13)												r2	
	r12(11,13)											r13		r11
	r13(14)						r14							
	r14(6,15)		r15		r6									
	r15(16)								r16					
	r16(17)								r17					
	r17(18)			r18										
aceptacion	r18(19)													



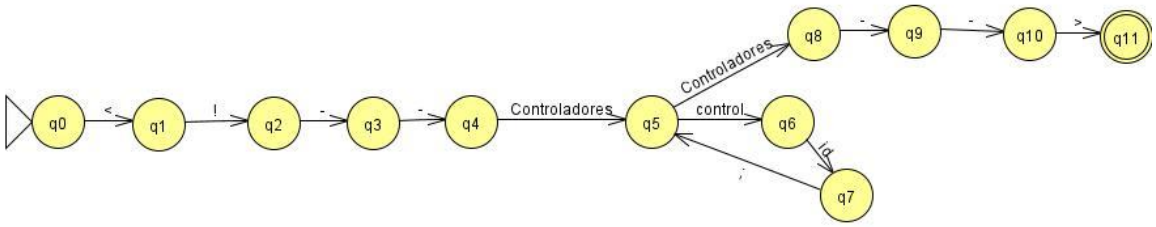
N	token	Follow
1	<	2
2	!	3
3	-	4
4	-	5
5	colocacion	6,15
6	id	7
7	.	8
8	posicion	9
9	(	10
10	Dato	11,13
11	,	12
12	Dato	13
13	)	14
14	;	6,15
15	colocacion	16
16	-	17
17	-	18
18	>	19
19		

aceptacion	estado	<	Propiedades	>	id	.	;	!	-	propiedad	(	)	valor	,
	s0(1)	s1												
	s1(2)							s2						
	s2(3)								s3					
	s3(4)								s4					
	s4(5)		s5											
	s5(6,15)		s15		s6									
	s6(7)					s7								
	s7(8)									s8				
	s8(10)										s9			
	s9(11)												s10	
	s10(11,13)												s13	s11
	s11(13)												s2	
	s12(13)												s13	
	s13(14)						s14							
	s14(6,15)		s15		s6									
	s15(16)								s16					
	s16(17)								s17					
	s17(18)			s18										
aceptacion	s18(19)													

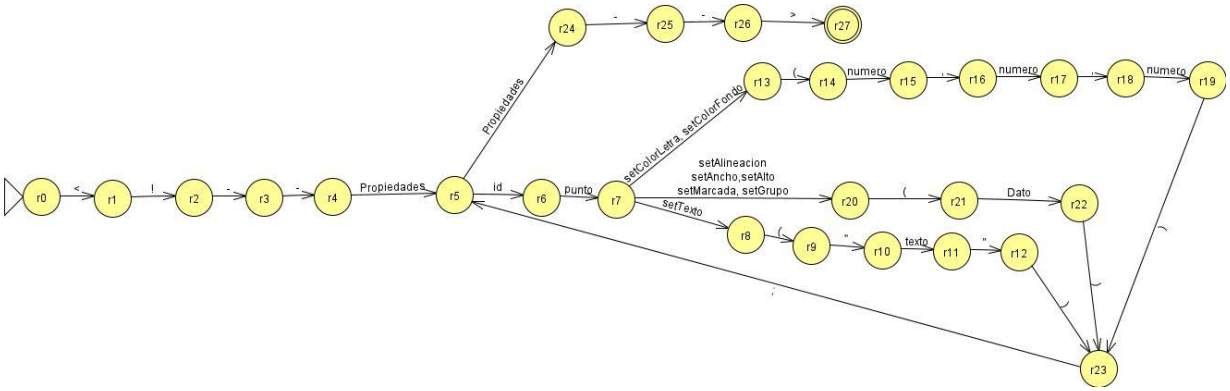
Grafico del autómata finito

A continuación, se presentan los autómatas correspondientes a cada parte o sección a ser analizada.

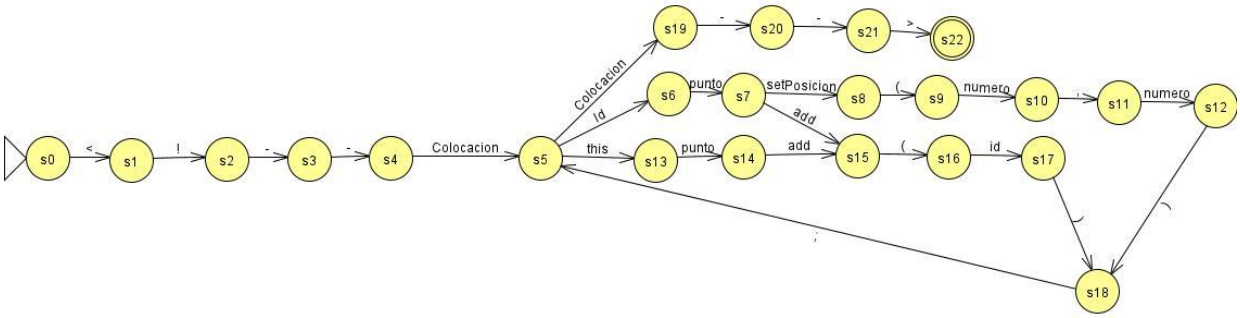
ÁREA DE CONTROLES



ÁREA DE PROPIEDADES



ÁREA DE COLOCACION



Gramática de libre contexto

La gramática libre de contexto es la siguiente:

TERMINALES = { Token_barra, Token_asterisco, Token_mayor, Token_menor, Token_signoE, Token_guion, Token_pcoma, Token_punto, Token_pa, Token_pc, Token_coma, Token_comilla, Token_elemento, Token_E_Control, Token_E_Propiedad, Token_E_Colocar, Token_Control, Token_Propiedad, Token_P_valor, Token_parametro, Token_posicion, Token_Id, Token_numero, Token_texto}

NO TERMINALES={<Inicio> <Comentario> <Texto> <otra línea> <Controles> <Control> <otro control> <Propiedades> <propiedad><valores><valor texto> <numérico> <otro numero> <otra propiedad> <Colocacion> <Colocar> <otrains><posición> }

INICIO= <Inicio>

PRODUCCIONES:

<Inicio> ::= <Comentario> <Controles> <Comentario> <Propiedades> <Comentario>
<Colocación> <Comentario>

<Comentario> ::= Token_barra Token barra Token_texto | Token _barra Token_asterisco
<Texto>

<Texto>::= Token_texto <otra línea> Token_asterisco Token_barra|<otra línea>|épsilon

<otra línea>::= Token_texto <otra línea> Token_asterisco Token_barra I épsilon

<Controles>::= Token_menor Token_signoE Token_guion Token_guion
Token_E_Control <Control> Token_E_Control Token_guion
Token_guion Token_mayor

<Control> ::= Token_Control Token_punto Token_Id Token_pcoma <otro control>
|epsilon

<otro control> ::= <Control> |epsilon

<Propiedades> ::= Token_menor Token_signoE Token_guion Token_guion
Token_E_Propiedad <propiedad> Token_E_Propiedad Token_guion
Token_guion Token_mayor

<propiedad> ::= Token_Id Token_punto Token_Propiedad Token_pa < valores>
Token_pc Token_pcoma <otra propiedad> |épsilon

<valores>::= Token_Id |Token_parametro|Token_p_valor |<valor texto>|<numerico> | épsi

<valor texto> ::= Token_comilla Token_texto Token_comilla

<numérico> ::= Token_numero <otro numero>

<otro numero> ::= Token_coma <numérico> |épsilon

<otra propiedad >:: <propiedad>|épsilon

<Colocacion> ::= Token_menor Token_signoE Token_guion Token_guion
Token_E_Colocar <Colocar> Token_E_Colocar Token_guion
Token_guion Token_mayor

<Colocar> ::= Token_elemento Token_punto Token_posicion Token_pa <posición>
Token_pc Token_pcoma <otra ins > | épsilon

<otra ins >:: <Colocar>|épsilon

<posición> ::= Token_Id | Token_numero Token_coma Token_numero