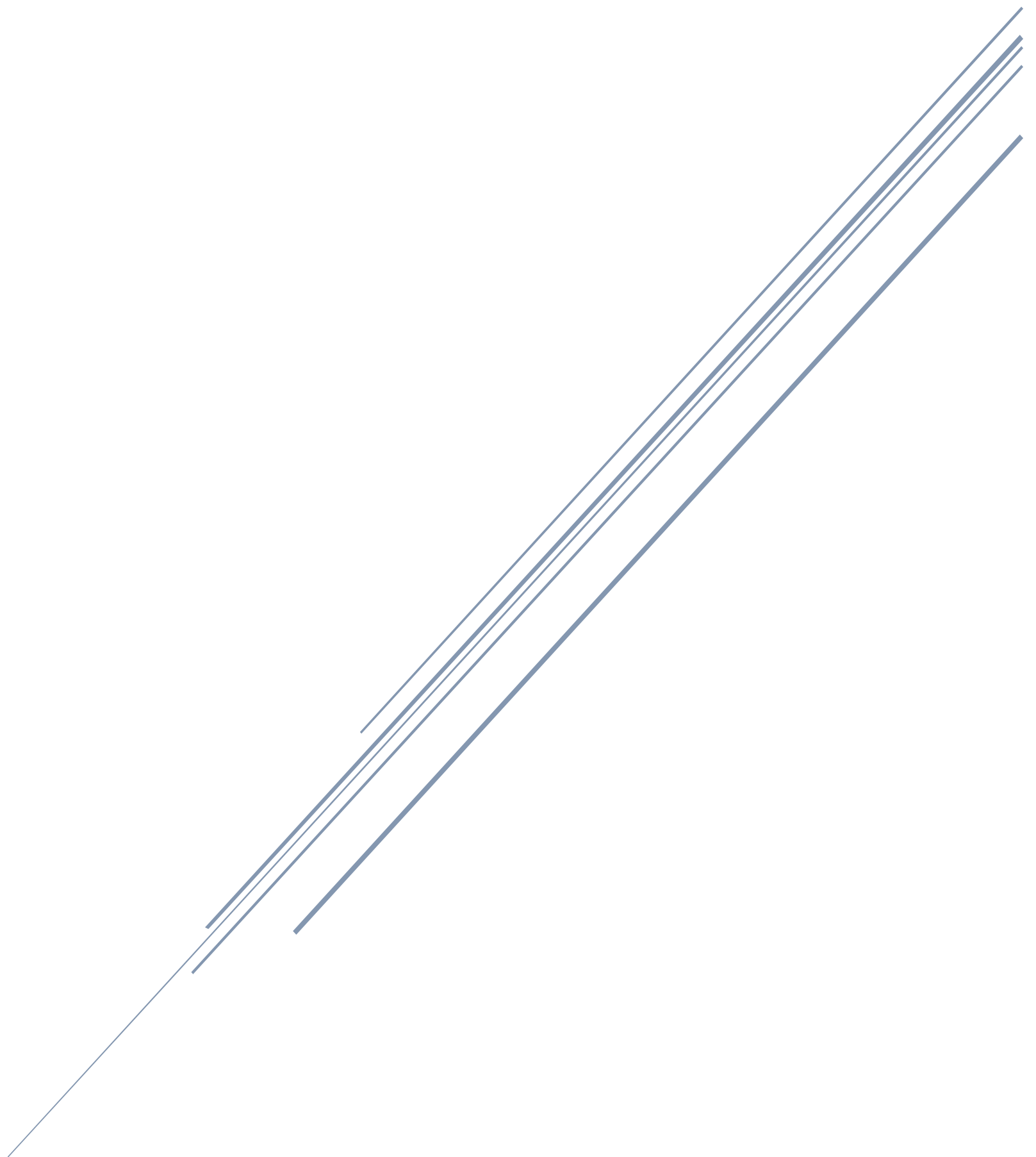


# PROYECTO 2 LFP

## Manual Técnico



USAC – Facultad de ingeniería  
Damián Ignacio Peña Afre

CONTENIDO

ENCABEZADO .....	2
TECNICAS O PARADIGMAS .....	2
Convenciones de nomenclatura.....	2
Diagrama de clases .....	3
metodos principales .....	4
1.  getNextToken y evalCharacter.....	4
2.  evalTokens y evalState.....	5
3.  createSymbolTable y applyProperties.....	7
Requerimientos .....	8
descripción .....	8
nterfaces principales.....	8
1.  Menu principal.....	8
2.  Editor de archivo .....	9
Planificación .....	10
Anexos .....	11

## ENCABEZADO

- Nombre del sistema: Compilador a HTML
- Desarrollador: Damián Ignacio Peña Afre
- Lenguaje utilizado: Python
- Tipo de programa: Desktop
- Cuenta con interfaz gráfica

## TECNICAS O PARADIGMAS

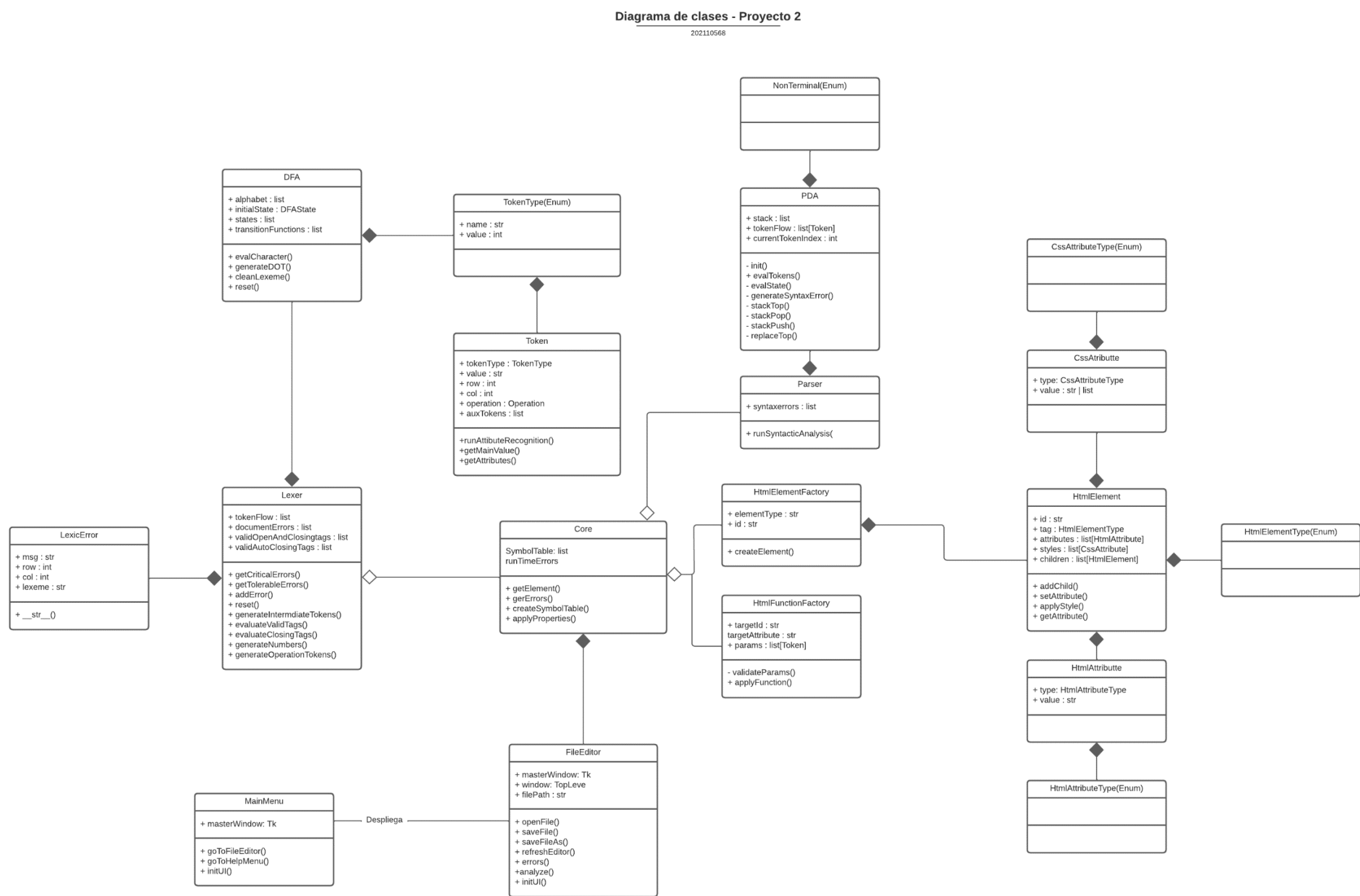
Se utilizo el paradigma de la programación orientada a objetos junto con la separación de los archivos en carpetas en función del tipo de tarea que realizan, de esta manera el proyecto esta segmentado de una forma modular.

Se separó la lógica del aspecto visual, de modo que obtiene una estructura más limpia de código.

## CONVENCIONES DE NOMENCLATURA

Se optó por utilizar '*camel/case*' para nombrar los archivos y variables. Tanto el nombre de la clase como el archivo que lo contiene empiezan con letra mayúscula. Los métodos/funciones y archivos que contienen únicamente métodos/funciones empiezan por letra minúscula.

# DIAGRAMA DE CLASES



## METODOS PRINCIPALES

### 1. *getNextToken* y *evalCharacter*

Dentro de la clase que define al autómata encargado del análisis léxico (DFA) se encuentra estos métodos que se encargan de determinar si existe alguna transición con el carácter recibido en la cadena de entrada o programa fuente, llevar un conteo de las filas y columnas, omitir las líneas con comentarios, así como de generar los tokens de los elementos reconocidos.

```
def getNextToken(self):

    while self.currentIndex < len(self.stringFlow):
        token = None
        currentCharacter = self.stringFlow[self.currentIndex]

        # Comments

        # * Skips line
        if self.skipLine:
            self.currentIndex += 1
            self.col += 1

            if currentCharacter == '\n':
                self.row += 1
                self.col = 1
                self.skipLine = False

            continue

        # * Skips line breaks
        if self.skipLineBreaks and currentCharacter == '\n':
            token = self._evalAcceptanceState()
            self.row += 1
            self.col = 1
            self.currentIndex += 1

        if token == None:
            token = self._evalCharacter()

        if token == True:
            self.col += 1
            self.currentIndex += 1
            continue
        if isinstance(token, Token):

            if token.tokenType == TokenType.ONELINE_COMMENT_OPENING:
                self.skipLine = True
                # self.col += 1
                # self.currentIndex += 1
                continue

            if token.tokenType == TokenType.MULTILINE_COMMENT_OPENING:
                self.skipToken = True
                # self.col += 1
                # self.currentIndex += 1
                continue

            if token.tokenType == TokenType.MULTILINE_COMMENT_CLOSING:
                self.skipToken = False
                # self.col += 1
                # self.currentIndex += 1
                continue

            if self.skipToken:
                # self.col += 1
                # self.currentIndex += 1
```

```

        continue

    return token

    if isinstance(token, LexicError):
        return token

    if self.lexeme != '':
        return self._evalAcceptanceState()

    return None

# True -> valid state/character
# False -> invalid state/character
# Token -> valid token/lexeme
# LexicError -> invalid token/lexeme
def _evalCharacter(self):
    currentCharacter = self.stringFlow[self.currentIndex]

    if currentCharacter == ' ':

        if self.skipBlankSpaces:
            token = self._evalAcceptanceState()
            self.col += 1
            self.currentIndex += 1
            return token
        else:
            self.lexeme += currentCharacter
            return True

    # ...

```

## 2. evalTokens y evalState

Luego de haber pasado por el analizar léxico, el PDA o autómatas de pila es el encargado de revisar el orden de los tokens dentro del flujo de tokens. Para ello almacena una serie de símbolos no terminales y terminales dentro de una pila, de manera tal, que al vaciar completamente esta pila y no haber mas tokens validar el flujo.

```

def evalTokens(self):
    while self._stackTop() != EOF_TOKEN:
        resp = self._evalState()

        if isinstance(resp, SyntaxError):
            return resp

    # POP EOF
    self._stackPop()

    if self.currentTokenIndex < len(self.tokenFlow):
        # !
        return SyntaxError(self.tokenFlow[self.currentTokenIndex], [EOF_TOKEN])

    return None

def _evalState(self):
    evaluatedToken = EMPTY_TOKEN
    stackTop = self._stackTop()

    try:
        evaluatedToken = self.tokenFlow[self.currentTokenIndex]
    except IndexError:
        if not isinstance(stackTop, NonTerminal):
            evaluatedToken.row = self.tokenFlow[self.currentTokenIndex-1].row
            evaluatedToken.col = self.tokenFlow[self.currentTokenIndex-1].col
            return SyntaxError(evaluatedToken, [stackTop])

```

```

# * NonTerminal replacement rules

# S -> 4 11 A 11 5 4 11 B 11 5 4 11 C 11 5
if stackTop == NonTerminal.S:

    replace = [
        OPEN_SCOPE_TOKEN,
        CONTROL_SCOPE_TOKEN,
        NonTerminal.A,
        CONTROL_SCOPE_TOKEN,
        CLOSE_SCOPE_TOKEN,
        OPEN_SCOPE_TOKEN,
        PROPERTIES_SCOPE_TOKEN,
        NonTerminal.B,
        PROPERTIES_SCOPE_TOKEN,
        CLOSE_SCOPE_TOKEN,
        OPEN_SCOPE_TOKEN,
        LOCATION_SCOPE_TOKEN,
        NonTerminal.B,
        LOCATION_SCOPE_TOKEN,
        CLOSE_SCOPE_TOKEN,
    ]
    self._replaceTop(replace)
    return True

# A -> 11 11 7 A / epsilon
if stackTop == NonTerminal.A:
    # ! Preanalysis
    if evaluatedToken.tokenType == TokenType.ID and evaluatedToken.lexeme !=
CONTROL_SCOPE_TOKEN.lexeme and evaluatedToken.lexeme != PROPERTIES_SCOPE_TOKEN.lexeme and
evaluatedToken.lexeme != LOCATION_SCOPE_TOKEN.lexeme:
        replace = [
            VALID_CONTROL_TOKEN,
            VALID_VARIABLE_TOKEN,
            SEMICOLON_TOKEN,
            NonTerminal.A
        ]
        self._replaceTop(replace)
        return True

    # / epsilon
    self._stackPop()
    return True

# * Terminal replacement rules

# 4 -> <!--
if stackTop == OPEN_SCOPE_TOKEN and evaluatedToken.tokenType ==
TokenType.DEFINITION_SCOPE_OPENING:
    self._stackPop()
    self.currentTokenIndex += 1
    return True

# 5 -> -->
if stackTop == CLOSE_SCOPE_TOKEN and evaluatedToken.tokenType ==
TokenType.DEFINITION_SCOPE_CLOSING:
    self._stackPop()
    self.currentTokenIndex += 1
    return True

```

### 3. createSymbolTable y applyProperties

Luego de procesar sintácticamente el flujo de tokens se construye la tabla de símbolos, donde cada token de tipo variable es inicializado como un elemento HTML gracias a la clase HtmlElementFactory. Luego de que se haya inicializado cada uno de estos objetos se procede a leer los 2 bloques restantes para aplicar o modificar sus propiedades.

```
def createSymbolTable(self):  
  
    # this element is the root of the html tree  
    Core.SymbolTable.append(HtmlElement("this", HtmlElementType.BODY))  
  
    # * analyze the controls scope  
  
    index = 0  
    while index < len(Lexer.tokenFlow):  
        token = Lexer.tokenFlow[index]  
  
        if token.idType == idType.CONTROL:  
            nextToken = Lexer.tokenFlow[index + 1]  
            element = HtmlElementFactory(token.lexeme, nextToken.lexeme)  
            Core.SymbolTable.append(element.createElement())  
  
            index += 1  
  
def applyProperties(self):  
  
    # Skip definition scope  
    index = 0  
    while index < len(Lexer.tokenFlow):  
        token = Lexer.tokenFlow[index]  
  
        if token.tokenType == TokenType.DEFINITION_SCOPE_CLOSING:  
            index += 1  
            break  
        index += 1  
  
    while index < len(Lexer.tokenFlow):  
        tokenFlow = Lexer.tokenFlow  
  
        if tokenFlow[index].idType == idType.VARIABLE:  
            targetId = tokenFlow[index].lexeme  
            targetAttribute = tokenFlow[index + 2].lexeme  
            params = []  
  
            # parameters  
            for j in range(index+4, len(tokenFlow)):  
  
                if tokenFlow[j].tokenType == TokenType.COMMA:  
                    continue  
  
                if tokenFlow[j].tokenType == TokenType.PARENTHESIS_CLOSING:  
                    index = j  
                    break  
  
                params.append(tokenFlow[j])  
  
            # Execute function  
            function = HtmlFunctionFactory(targetId, targetAttribute, params)  
            res = function.applyFunction()  
            if res != True:  
                Core.runTimeErrors.append({  
                    'type': 'runtime',  
                    'row': str(tokenFlow[index].row),  
                    'column': str(tokenFlow[index].col),  
                    'lexema': "",  
                    'expected': "",  
                    'description': res  
                })  
  
            index += 1
```



## REQUERIMIENTOS

- Contar con un menú de opciones
- Cargar un archivo con extensión GPW
- Editar el archivo fuente
- Analizar el archivo fuente
- Mostrar los errores del archivo fuente

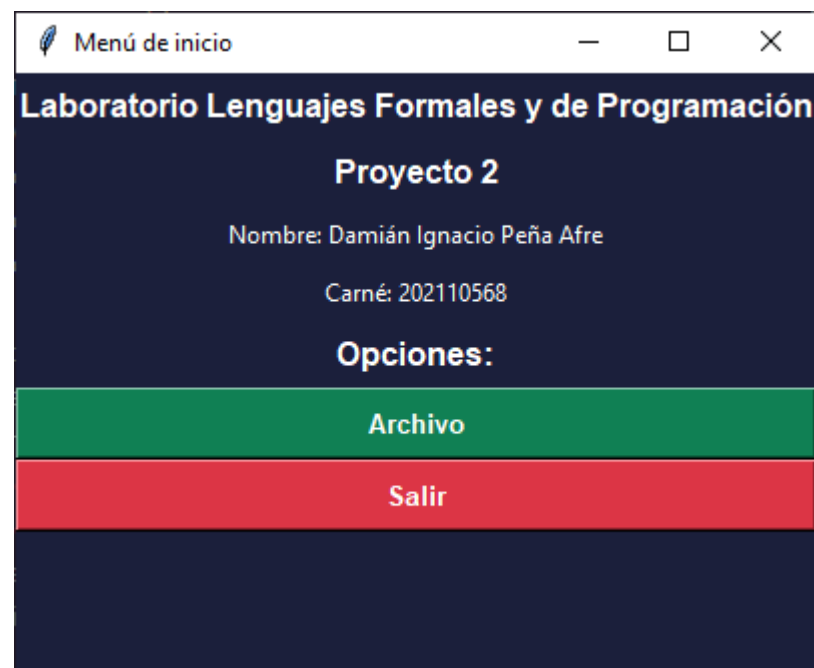
## DESCRIPCIÓN

Para la realización del proyecto se utilizaron las siguientes herramientas:

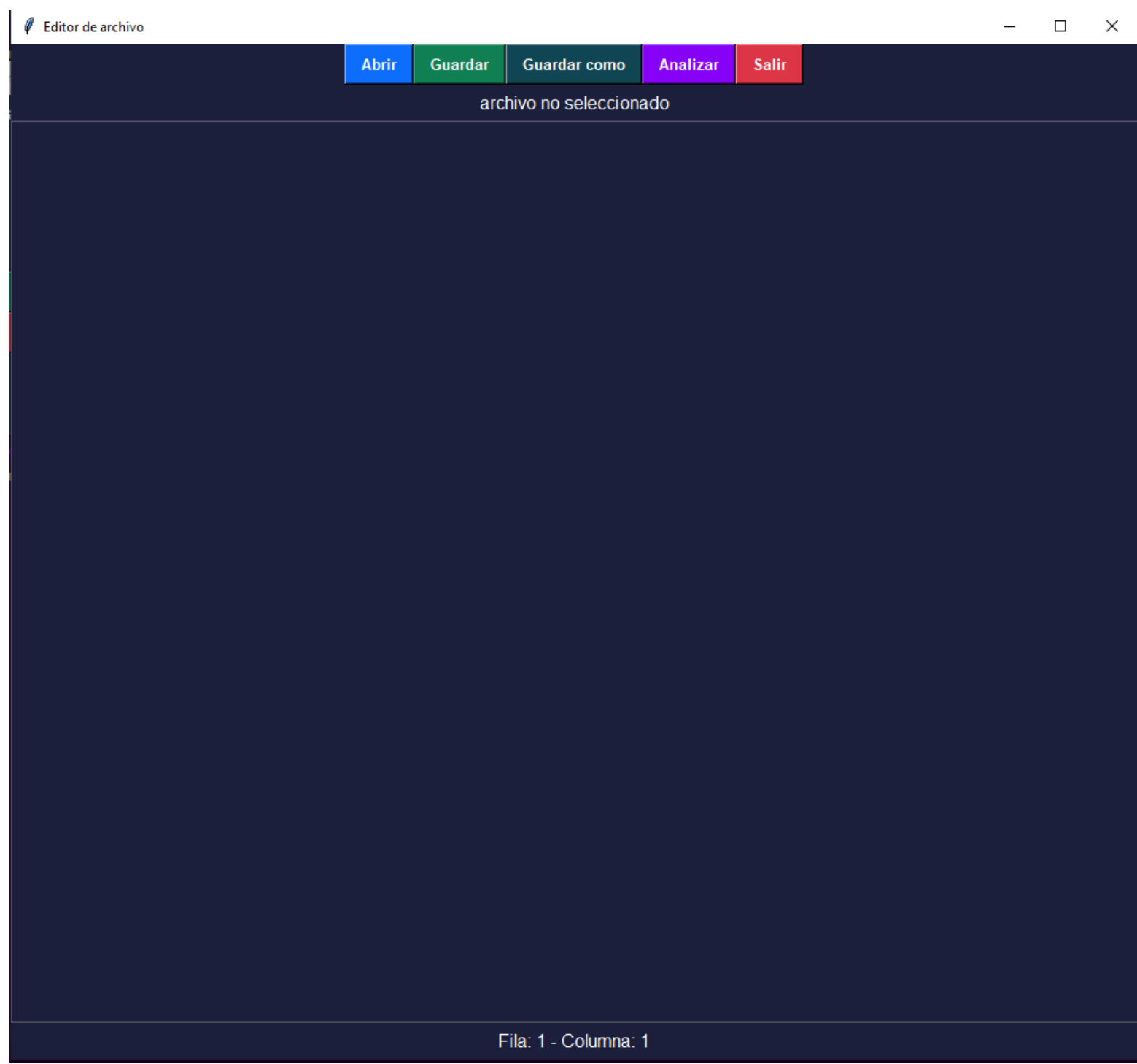
1. Visual Studio Code: Editor de código ligero.
2. Git: Programa para llevar un control de las versiones del proyecto.
3. Github: Repositorio remoto para llevar un control de las versiones del proyecto en la nube.

## NTERFACES PRINCIPALES

### *1. Menu principal*



## 2. Editor de archivo



Se toma el contenido dentro del área de texto para sobrescribir o crear un archivo. Es posible también analizar el contenido de dicho archivo con el botón de analizar, en dado caso se presenten errores, es posible también, presionar el botón de errores para generar dicho reporte.

## PLANIFICACIÓN

Día	Tarea	Tiempo empleado (Horas)
<i>Domingo 16</i>	Diagramación y diseño de autómatas	3
<i>Lunes 17</i>	Analizador léxico	3
<i>Martes 18</i>	Corrección de errores analizador léxico	2
<i>Miércoles 19</i>	Analizador sintáctico	3
<i>Jueves 20</i>	Corrección de errores analizador sintáctico	1
<i>Viernes 21</i>	Generación de elementos HTML	3
<i>Sábado 22</i>	Generación de reportes a partir de elementos Html	2
<i>Jueves 27</i>	Ultimas correcciones, ajustes y documentación	2
<i>Viernes 28</i>	Entrega	-

# ANEXOS

## Tokens

L = Set of letters  
N = Set of numbers  
P = parameters = {ID, string, boolean, number}  
e = épsilon

No	REGEX	Name	Additional Props
1	//	oneline_comment_opening	
2	/*	multiline_comment_opening	
3	*/	multiline_comment_closing	
4	<!--	definition_scope_opening	Scope: Controles <sup>1</sup>   propiedades <sup>2</sup>   Colocacion <sup>3</sup>
5	-->	definition_scope_closing	Scope: Controles <sup>1</sup>   propiedades <sup>2</sup>   Colocacion <sup>3</sup>
6	.	point	
7	;	semicolon	
8	true   false	boolean	
9	N+	number	
10	“(L N) #”	String	
11	L(N L) *	id	Type: <i>Control</i> <sup>1</sup>   variable <sup>2</sup>   this <sup>3</sup>   <i>function</i> <sup>4</sup> / <i>scope</i> <sup>5</sup>
12	(	parenthesis_opening	
13	)	parenthesis_closing	
14	,	Comma	

## Gramatica

$\epsilon \rightarrow \text{épsilon}$

### Initial state

$S_0 \rightarrow 4^1 11^5 \text{ A } 11^5 5^1 \quad 4^2 11^5 \text{ B } 11^5 5^2 \quad 4^3 11^5 \text{ C } 11^5 5^3$

### Control scope

$A \rightarrow 11^{\rightarrow 1} 11^{\rightarrow 2} 7 \text{ A}$   
|  $\epsilon$

### Properties Scope

$B \rightarrow 11^2 |^3 6 11^{\rightarrow 4} 12 \text{ D } 13 7 \text{ B}$   
|  $\epsilon$

$D \rightarrow 8 \text{ E}$   
|  $9 \text{ E}$   
|  $10 \text{ E}$   
|  $11 \text{ E}$

$E \rightarrow 14 \text{ D}$   
|  $\epsilon$

### Location Scope

$C \rightarrow 11^2 |^3 6 11^{\rightarrow 4} 12 \text{ D } 13 7 \text{ C}$   
|  $\epsilon$

$\epsilon, E, \epsilon$

$\epsilon, E, 14 D$

$\epsilon, D, 11 E$

$\epsilon, D, 10 E$

$\epsilon, D, 9 E$

$\epsilon, D, 8 E$

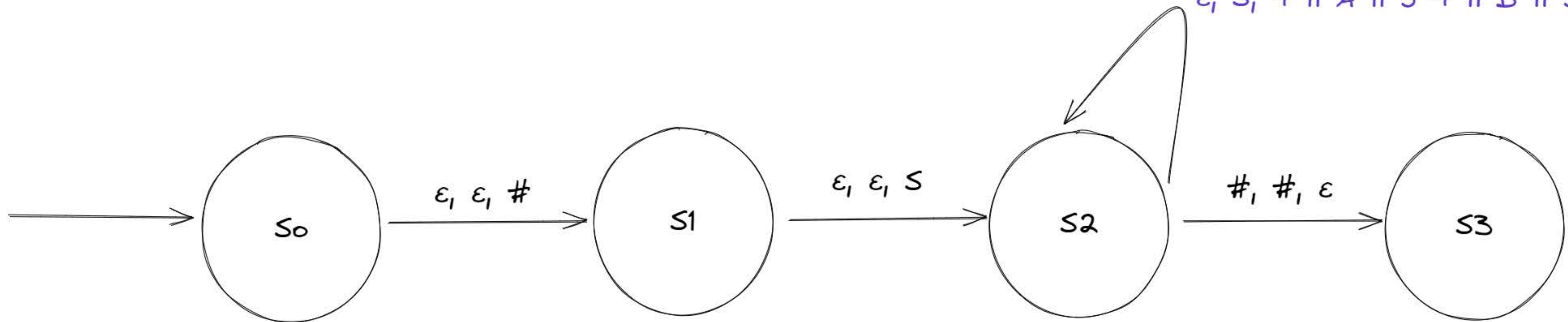
$\epsilon, B, \epsilon$

$\epsilon, B, 11 6 11 12 D 13 7 B$

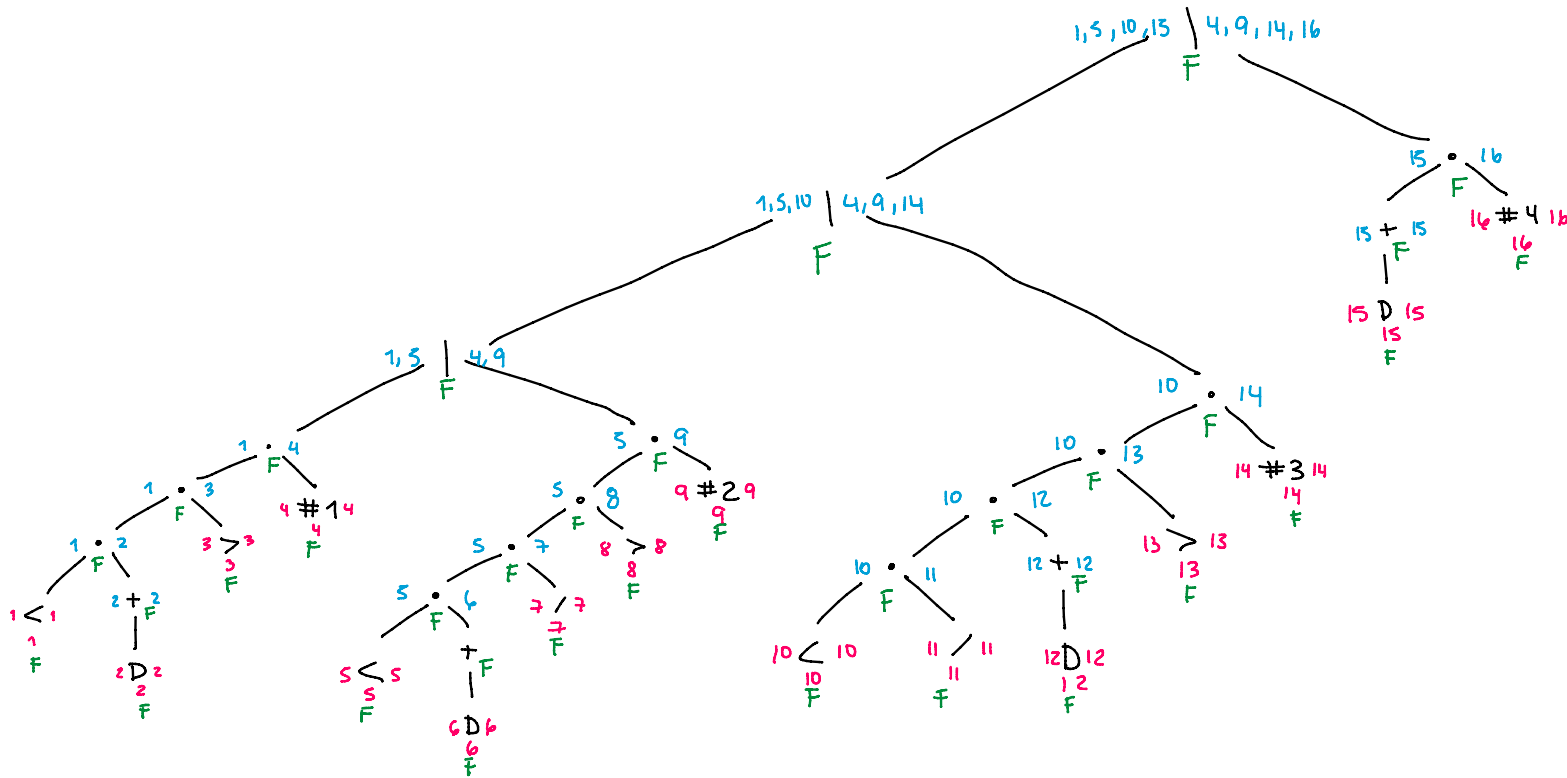
$\epsilon, A, \epsilon$

$\epsilon, A, 11 11 7 A$

$\epsilon, S, 4 11 A 11 5 4 11 B 11 5 4 11 B 11 5$



## Automata 1

$$(\underset{1}{\langle} \underset{2}{D^+} \underset{3}{\rangle} \underset{4}{\#1}) | (\underset{5}{\langle} \underset{6}{D^+} \underset{7}{\rangle} \underset{8}{\#2}) | (\underset{9}{\langle} \underset{10}{D^+} \underset{11}{\rangle} \underset{12}{\#3}) | (\underset{13}{D^+} \underset{14}{\#4})$$


# modo	Simbolo	sig pos
1	<	2
2	D	2, 3
3	>	4
4	# 1	—
5	<	6
6	D	6, 7
7	/	8
8	>	9
9	# 2	—
10	<	11
11	/	12
12	D	12, 13
13	>	14
14	# 3	—
15	D	15, 16
16	# 4	—

$$S_0 = 1, 5, 10, 15$$

$\begin{matrix} < & < & < & & D \end{matrix}$

$$T(S_0, <) = \text{sig}(1) \cup \text{sig}(5) \cup \text{sig}(10)$$

$$= 2, 6, 11 = S_1$$

$\begin{matrix} D & D & / \end{matrix}$

$$T(S_0, D) = \text{sig}(15) = 15, 16 = S_2$$

$\begin{matrix} D & \#4 \end{matrix}$

$$T(S_1, D) = \text{sig}(2) \cup \text{sig}(6) = 2, 3, 6, 7 = S_3$$

$\begin{matrix} D & > & D & / \end{matrix}$

$$T(S_1, /) = \text{sig}(11) = 12 = S_4$$

$\begin{matrix} D \end{matrix}$

$$T(S_2, D) = \text{sig}(15) = S_2$$

$$T(S_3, D) = \text{sig}(2) \cup \text{sig}(6) = S_3$$

$$T(S_3, >) = \text{sig}(3) = 4 = S_5$$

$\begin{matrix} \#1 \end{matrix}$

$$T(S_3, /) = \text{sig}(7) = 8 = S_6$$

$\begin{matrix} > \end{matrix}$

$$T(S_4, D) = \text{sig}(12) = 12, 13 = S_7$$

$\begin{matrix} D & > \end{matrix}$

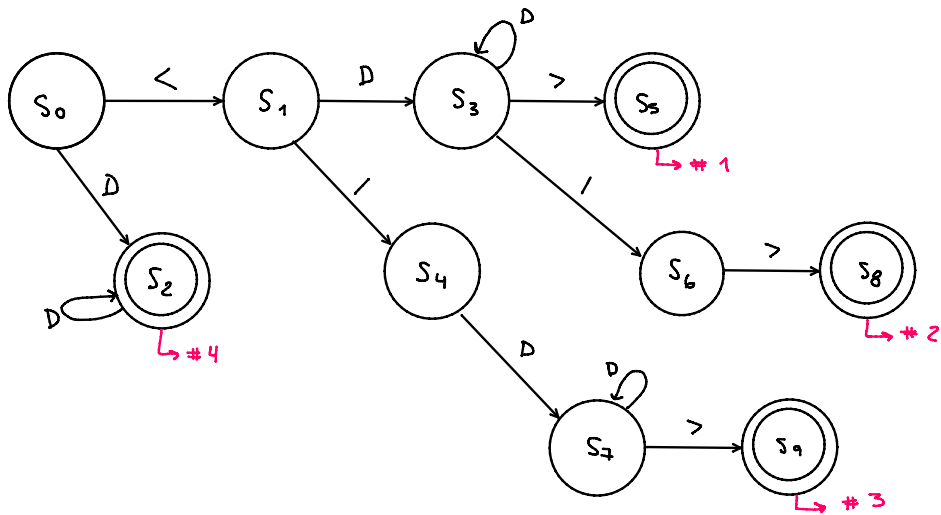
$$T(S_6, >) = \text{sig}(8) = 9 = S_8$$

$\begin{matrix} \#2 \end{matrix}$

$$T(S_7, D) = \text{sig}(12) = S_7$$

$$T(S_7, >) = \text{sig}(13) = 14 = S_9$$

$\begin{matrix} \#3 \end{matrix}$

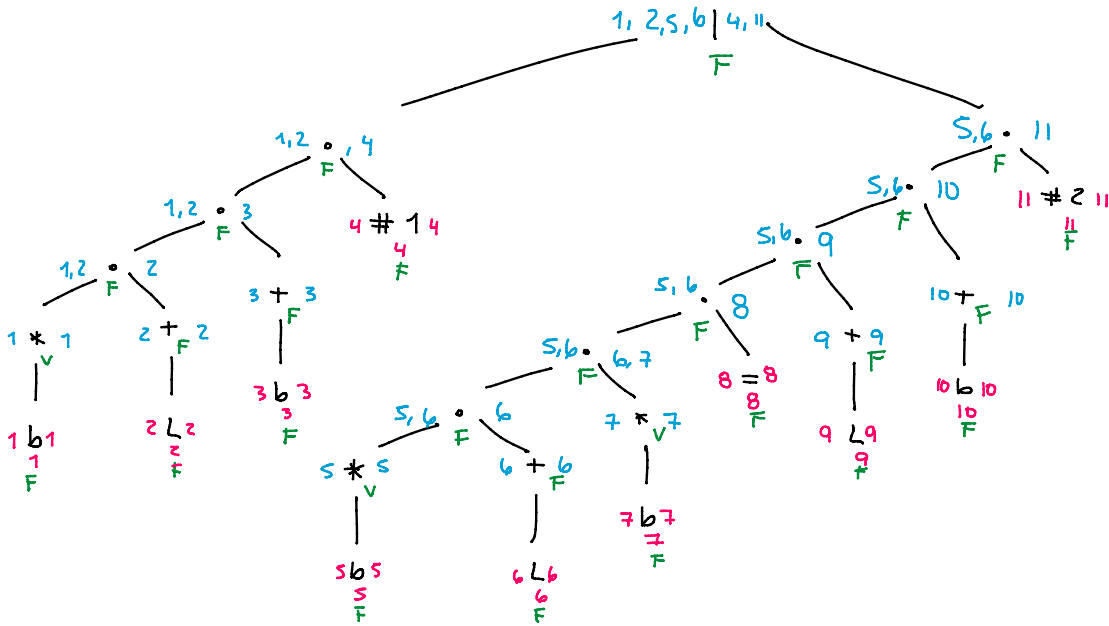




## Automata 2

$$(b^* L^+ b^+ \# 1) \mid (b^* L^+ b^* = L^+)$$

1 2 3 4     5 6 7 8 9



# No	Simb	sig
1	b	1, 2
2	L	2, 3
3	b	3, 4
4	# 1	
5	b	5, 6
6	L	6, 7, 8
7	b	7, 8
8	=	
9	L	9, 10
10	b	10, 11
11	# 2	

$$S_0 = 1, 2, 5, 6$$

b L b L

$$T(S_0, b) = \text{sig}(1) \cup \text{sig}(5) = 1, 2, 5, 6 = S_0$$

b L b L

$$T(S_0, L) = \text{sig}(2) \cup \text{sig}(6) = 2, 3, 6, 7, 8 = S_1$$

L b L b =

$$T(S_1, L) = \text{sig}(2) \cup \text{sig}(6) = S_1$$

$$T(S_1, b) = \text{sig}(3) \cup \text{sig}(7) = 3, 4, 7, 8 = S_2$$

b # 1 b =

$$T(S_1, \epsilon) = \text{sig}(b) = \underset{L}{a} = S_3$$

$$T(S_2, b) = \text{sig}(3) \cup \text{sig}(7) = S_2$$

$$T(S_2, \epsilon) = \text{sig}(b) = a = S_3$$

$$T(S_3, L) = \underset{L}{\text{sig}}(\underset{b}{a}) = \underset{L}{a}, \underset{b}{10} = S_4$$

$$T(S_4, L) = \text{sig}(a) = S_4$$

$$T(S_4, b) = \text{sig}(b) = \underset{b}{10, 11} = S_5$$

$$T(S_5, b) = S_5$$

