MANUAL TECNICO
PROYECTO #1
“BITXELART”

Luis Manuel Chay Marroquín

202000343

Huehuetenango, 23 de septiembre de 2021

DESCRIPCION GENERAL

El siguiente programa fue diseñado para la empresa “Bitxelart”, el programa permite la elaboración de imágenes digitales en estilo pixel art a partir de un archivo de texto plano con extensión .pxla elaborado por un empleado de la empresa, este archivo contiene los datos necesarios para la creación de archivos .html con la imagen solicitada por el cliente, también incluye una serie de palabras reservadas que indica el tipo de filtro a aplicar a la imagen solicitada, también nos permite la conversión de estos archivos .html a imágenes de extensión .png y mostrarlos tanto en la interfaz gráfica del programa como en nuestra computadora como un archivo nuevo.

DATOS TECNICOS

LENGUAJE UTILIZADO

El programa fue desarrollado en el lenguaje de programación Python 3.

IDE UTILIZADO

Atom

SISTEMA OPERATIVO

El programa puede ser ejecutado en los sistemas operativos Windows, Linux/UNIX, MacOS, AIX, IMB i, iOS, iPadOS, OS/390, z/OS, RISC PS, Solaris, VMS y HP-UX

ALGORITMOS CLAVES

METODO	DESCRIPCION
“init”	`def __init__(self):` `  self.listaTokens = []` `  self.listaErrores = []` `  self.linea = 1` `  self.columna = 1` `  self.buffer = ''` `  self.estado = 0` `  self.i = 0`
“agregar_token”	`def agregar_token(self,caracter,token,linea,columna):` `  self.listaTokens.append(Token(caracter,token,linea,columna))` `  self.buffer = ''`
“agregar_error”	`def agregar_error(self,caracter,linea,columna):` `  self.listaErrores.append(Error('Caracter ' + caracter + ' no reconocido.', linea, columna))`
“estado0”	`def estado0(self,caracter):`
“estado1”	`def estado1(self,caracter):`
“estado2”	`def estado2(self,caracter):`
“estado3”	`def estado3(self,caracter):`
“estado4”	`def estado4(self,caracter):`
“estado5”	`def estado6(self,caracter):`
“estado6”	`def estado6(self,caracter):`
“estado7”	`def estado7(self,caracter):`
“estado8”	`def estado8(self,caracter):`
“estado9”	`def estado9(self,caracter):`
“estado10”	`def estado10(self,caracter):`
“estado12”	`def estado12(self,caracter):`
“estado13”	`def estado13(self,caracter):`
“analizar”	`def analizar(self, cadena):` `  '''Analiza léxicamente una cadena'''`

“reporteTokens”	
“reporteErrores”	
“leerArchivo”	
“analisis”	

```
def reporteTokens(self):
    x = PrettyTable()
    x.field_names = ["Lexema", "Token", "Fila", "Columna"]
    for i in self.listaTokens:
        x.add_row(i.enviarDataTok())
    cadenatokens = x.get_html_string()
    cadenatokensform = "{}".format(cadenatokens)
```

```
def reporteErrores(self):
    x = PrettyTable()
    x.field_names = ["Descripcion", "Fila", "Columna"]
    if len(self.listaErrores)==0:
        print('No hay errores')
    else:
        for i in self.listaErrores:
            x.add_row(i.enviarDataErr())
        cadenaerrores = x.get_html_string()
        cadenaerroresform = "{}".format(cadenaerrores)
```

```
def leerArchivo():
    Tk().withdraw()
    entrada = askopenfilename(filetypes=[("Archivos PXLA", "*.pxla"), ("All Files", "*.*")])
    archivo = open(entrada, 'r')
    global contenido
    contenido = archivo.read()
    archivo.close()
    return contenido
```

```
def analisis():
    scanner = AnalisisLexico()
    scanner.analizar(contenido)
    scanner.reporteTokens()
    scanner.reporteErrores()

    tokens = scanner.listaTokens

    diccionario['TITULO'] = tokens[2].lexema
    diccionario['ANCHO'] = tokens[6].lexema
    diccionario['ALTO'] = tokens[10].lexema
    diccionario['FILAS'] = int(tokens[14].lexema)
    diccionario['COLUMNAS'] = int(tokens[18].lexema)
    diccionario['CELDAS'] = []
    diccionario['FILTROS'] = []
```

“crearhtml”	

```
def crearhtml():
    get_table(diccionario['ANCHO'], diccionario['ALTO'], diccionario['FILAS'], diccionario['COLUMNAS'],
    messagebox.showinfo("Felicidades", "Reporte creado exitosamente")


    import imgkit
    config=imgkit.config(wkhtmltoimage=f'C:\Program Files\wkhtmltopdf\\bin\wkhtmltoimage.exe')
    imgkit.from_file('original.html','original.png', config=config)
    imgkit.from_file('mirrorx.html','mirrorx.png', config=config)
    imgkit.from_file('mirrory.html','mirrory.png', config=config)
    imgkit.from_file('double.html','double.png', config=config)
```

“originalfoto”	

```
def originalfoto():
    global img
    img = Image.open("original.png")
    test = ImageTk.PhotoImage(img)
    resize = img.resize((200,200), Image.ANTIALIAS)
    nueva = ImageTk.PhotoImage(resize)

    label1 = tkinter.Label(image=nueva)
    label1.image = nueva

    T = Text(ventana, height = 5, width = 25)

    l = Label(ventana, text = "Original")
    l.config(font =("Courier", 14))
    l.pack()

    label1.place(x=15, y=25)
```

“mirrorx”	

```
def mirrorx():
    global img2
    img2 = Image.open("mirrorx.png")
    test2 = ImageTk.PhotoImage(img2)
    resize2 = img2.resize((200,200), Image.ANTIALIAS)
    nueva2 = ImageTk.PhotoImage(resize2)

    label2 = tkinter.Label(image=nueva2)
    label2.image = nueva2

    label2.place(x=15, y=300)

    T2 = Text(ventana, height = 5, width = 25)

    l2 = Label(ventana, text = "Mirror X")
    l2.config(font =("Courier", 14))
    l2.pack()
```

“mirrory”	
“double”	
“enviarDataerr”	
“enviarDataTok”	

```
def mirrory():
    global img3
    img3 = Image.open("mirrory.png")
    test3 = ImageTk.PhotoImage(img3)
    resize3 = img3.resize((200,200), Image.ANTIALIAS)
    nueva3 = ImageTk.PhotoImage(resize3)

    label3 = tkinter.Label(image=nueva3)
    label3.image = nueva3

    label3.place(x=600, y=25)

    T3 = Text(ventana, height = 5, width = 25)

    l3 = Label(ventana, text = "Mirror Y")
    l3.config(font =("Courier", 14))
    l3.pack()
```

```
def double():
    global img4
    img4 = Image.open("double.png")
    test4 = ImageTk.PhotoImage(img4)
    resize4 = img4.resize((200,200), Image.ANTIALIAS)
    nueva4 = ImageTk.PhotoImage(resize4)

    label4 = tkinter.Label(image=nueva4)
    label4.image = nueva4

    label4.place(x=600, y=300)

    T4 = Text(ventana, height = 5, width = 25)

    l4 = Label(ventana, text = "Double Mirror")
    l4.config(font =("Courier", 14))
    l4.pack()
```

```
def enviarDataErr(self):
    return [self.descripcion,self.linea, self.columna]
```

```
def enviarDataTok(self):
    return [self.lexema, self.tipo, self.linea, self.columna]
```

“crearhtmloriginal”		``` head = '<table>\n' body = '' bottom = '</table>' matriz = [] for i in range(0,filas): fila_temporal = [] for j in range(0,columnas): fila_temporal.append(crearceldavacia(ancho, alto)) matriz.append(fila_temporal) for celda in range(0, len(celdas)-1): if celdas[celda][2] == "FALSE": matriz[int(celdas[celda][1])][int(celdas[celda][0])] = crearceldavacia(ancho, alto) else: matriz[int(celdas[celda][1])][int(celdas[celda][0])] = crearcelda(ancho, alto, celdas[celda][3]) for fila in range(0,filas): body += '\t<tr>' for columna in range(0,columnas): body+=matriz[fila][columna] body += '\t</tr>' html = open(nombre_archivo_og,'w+') html.write(head+body+bottom) ```
“crearhtmlmirrorx”		``` if filtro == "MIRRORX": head = '<table>\n' body = '' bottom = '</table>' matriz = [] for i in range(0,filas): fila_temporal = [] for j in range(0,columnas): fila_temporal.append(crearceldavacia(ancho, alto)) matriz.append(fila_temporal) for celda in range(0, len(celdas)-1): if celdas[celda][2] == "FALSE": matriz[int(celdas[celda][1])][(filas) -1 - int(celdas[celda][0])] = crearceldavacia(ancho, alto) else: matriz[int(celdas[celda][1])][(filas) -1 - int(celdas[celda][0])] = crearcelda(ancho, alto, celdas[celda][3]) for fila in range(0,filas): body += '\t<tr>' for columna in range(0,columnas): body+=matriz[fila][columna] body += '\t</tr>' html = open(nombre_archivo_x,'w+') html.write(head+body+bottom) ```

“crearhtmlmirrory”	
“crearhtmldouble”	
“crearcelda”	
“crearceldavacia”	

```
elif filtro == "MIRRORY":
    head = '<table>\n'
    body = ''
    bottom = '</table>'
    matriz = []


    for i in range(0,filas):
        fila_temporal = []
        for j in range(0,columnas):
            fila_temporal.append(crearceldavacia(ancho, alto))
        matriz.append(fila_temporal)


    for celda in range(0, len(celdas)-1):
        if celdas[celda][2] == "FALSE":
            matriz[(columnas) -1 - int(celdas[celda][1])][int(celdas[celda][0])] = crearceldavacia(ancho, alto)
        else:
            matriz[(columnas) -1 - int(celdas[celda][1])][int(celdas[celda][0])] = crearcelda(ancho, alto, celdas[celda][3])

    for fila in range(0,filas):
        body += '\t<tr>'
        for columna in range(0,columnas):
            body+=matriz[fila][columna]
        body += '\t</tr>'
```

```
elif filtro == "DOUBLEMIRROR":
    head = '<table>\n'
    body = ''
    bottom = '</table>'
    matriz = []


    for i in range(0,filas):
        fila_temporal = []
        for j in range(0,columnas):
            fila_temporal.append(crearceldavacia(ancho, alto))
        matriz.append(fila_temporal)


    for celda in range(0, len(celdas)-1):
        if celdas[celda][2] == "FALSE":
            matriz[(columnas) -1 - int(celdas[celda][1])][(filas) -1 - int(celdas[celda][0])] = crearceldavacia(ancho, alto)
        else:
            matriz[(columnas) -1 - int(celdas[celda][1])][(filas) -1 - int(celdas[celda][0])] = crearcelda(ancho, alto, celdas[celda][3])

    for fila in range(0,filas):
        body += '\t<tr>'
        for columna in range(0,columnas):
            body+=matriz[fila][columna]
        body += '\t</tr>'

    html = open(nombre_archivo_db,'w+')
    html.write(head+body+bottom)
```

```
def crearcelda(texto1, texto2, texto3):
        return '\t\t<td width="{}" height="{}" bgcolor="{}"></td>\n'.format(texto1, texto2, texto3)
```

```
def crearceldavacia(texto1, texto2):
        return '\t\t<td width="{}" height="{}"></td>\n'.format(texto1, texto2)
```

METODOS MAS IMPORTANTES

METODO	DESCRIPCION	PARAMETROS
“init”	Inicializa los datos necesarios para la ejecución del autómata	Salida: listaTokens: Lista de los tokens vacía listaErrores Lista de los errores léxicos vacía línea: línea inicial para el análisis del carácter columna: columna inicial para el análisis del carácter buffer: buffer vacío estado: 0 (estado inicial del autómata) i = 0 valor inicial del contador
“agregar_token”	Agrega un token a la lista de Tokens	Salida: llistaTokens: Lista de los tokens con uno o más errores
“agregar_error”	Agrega un error léxico a la lista de Errores	Salida: llistaErrores: Lista de los errores con uno o más errores
“estado0”	Analiza carácter por carácter hasta encontrar un carácter de cambio de estado, asimismo analiza si hay errores léxicos	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token llistaErrores: Lista de los errores con uno o más errores cambios de estado
“estado1”	Analiza si el carácter es una letra y si forma cualquier palabra reservada necesaria para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado2”	Analiza si el carácter es un signo = necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado3”	Analiza si el carácter es un signo “ necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida:

	y regresa al estado 0 para continuar con el análisis	llistaTokens: Lista de los tokens con un nuevo token
“estado4”	Analiza si el carácter es un signo ; necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado5”	Analiza si el carácter es un numero necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado6”	Analiza si el carácter es un signo { necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado7”	Analiza si el carácter es un signo [necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado8”	Analiza si el carácter es un signo , necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado9”	Analiza si el carácter es un signo] necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado10”	Analiza si el carácter es un signo } necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“estado12”	Analiza si el carácter es un numero o un digito necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token

“estado13”	Analiza si el carácter es un simbolo @ necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis	Entrada: caracter: Caracteres en el archivo de entrada .pxla Salida: llistaTokens: Lista de los tokens con un nuevo token
“analizar”	Inicializa las listas de tokens y de errores y agrega un carácter centinela al final de la cadena de analisis y va creando los cambios de estados necesarios para el automata	Entrada: caracter: Caracteres en el archivo de entrada .pxla listaTokens: Lista de los tokens vacía listaErrores Lista de los errores léxicos vacía centinela: carácter $ que se agrega al final de la cadena Salida: llistaTokens: Lista de los tokens con un nuevo token llistaErrores: Lista de los errores con uno o más errores Cambios de estado
“reporteTokens”	Crea un reporte HTML a partir de la lista de Tokens creada en el analisis	Entrada: listaTokens: Lista de los tokens vacía Salida: Reporte de tokens en formato .htrml
“reporteErrores”	Crea un reporte HTML a partir de la lista de errores creada en el analisis	Entrada: listaErrores: Lista de los errores vacía Salida: Reporte de errorres en formato .htrml
“leerArchivo”	Abre un explorador de archivos para seleccionar el archivo .pxla de entrada ncesario para el funcionamiento del programa	Entrada: Archivo de extensión .pxla Salida: contenido: Contenido del archvio de entrada
“analisis”	Inicializa el analisis del archivo .pxla, mandando su contenido al automata y retornando los datos necesarios en un diccionario para usos futuros	Entrada: contenido: Contenido del archvio de entrada Salida: Diccionario: Un diccionario nativo de Python con las listas del contenido analizado por el automata
“crearhtml”	Manda los datos del diccionario creado por el automata necesarios para la creacion del html y retorna las imágenes solicitadas por el cliente con sus	Entrada: Diccionario: Un diccionario nativo de Python con las listas del contenido analizado por el automata

	filtros respectivos en formato .html e imagenes en formato .png	Salida: Archivos html y png de la imagen original y con los filtros solicitados por el cliente
“originalfoto”	Muestra la foto original en la interfaz grafica	Entrada: Imagen original Salida: Imagen orginal en interfaz grafica
“mirrorx”	Muestra la foto con el filtro Mirror X en la interfaz grafica	Entrada: Imagen con el filtro Mirror X Salida: Imagen con el filtro Mirror X en interfaz grafica
“mirrory”	Muestra la foto con el filtro Mirror Y en la interfaz grafica	Entrada: Imagen con el filtro Mirror Y Salida: Imagen con el filtro Mirror Y en interfaz grafica
“double”	Muestra la foto con el filtro Double Mirror en la interfaz grafica	Entrada: Imagen con el filtro Double Mirror Salida: Imagen con el filtro Double Mirror en interfaz grafica
“enviarDataerr”	Retorna los datos de la lista de Errores cuando sea necesario	Entrada: Errores lexicos en en analisis del contenido del archivo Salida: Lista de errores
“enviarDataTok”	Retorna los datos de la lista de Tokens cuando sea necesario	Entrada: Tokens en en analisis del contenido del archivo Salida: Lista de Tokens
“crearhtmloriginal”	Crea la imagen solicitada por el cliente en formato .html y .png	Entrada: Diccionario con listgas del analisis lexico del automata Salida: Reporte Html y su conversón a formato png de la imagen original solicitadea por el cliente
“crearhtmlmirrorx”	Crea la imagen con el filtro Mirror X solicitada por el cliente en formato .html y .png	Entrada: Diccionario con listgas del analisis lexico del automata Salida: Reporte Html y su conversón a formato png de la imagen con el filtro Mirror X solicitadea por el cliente

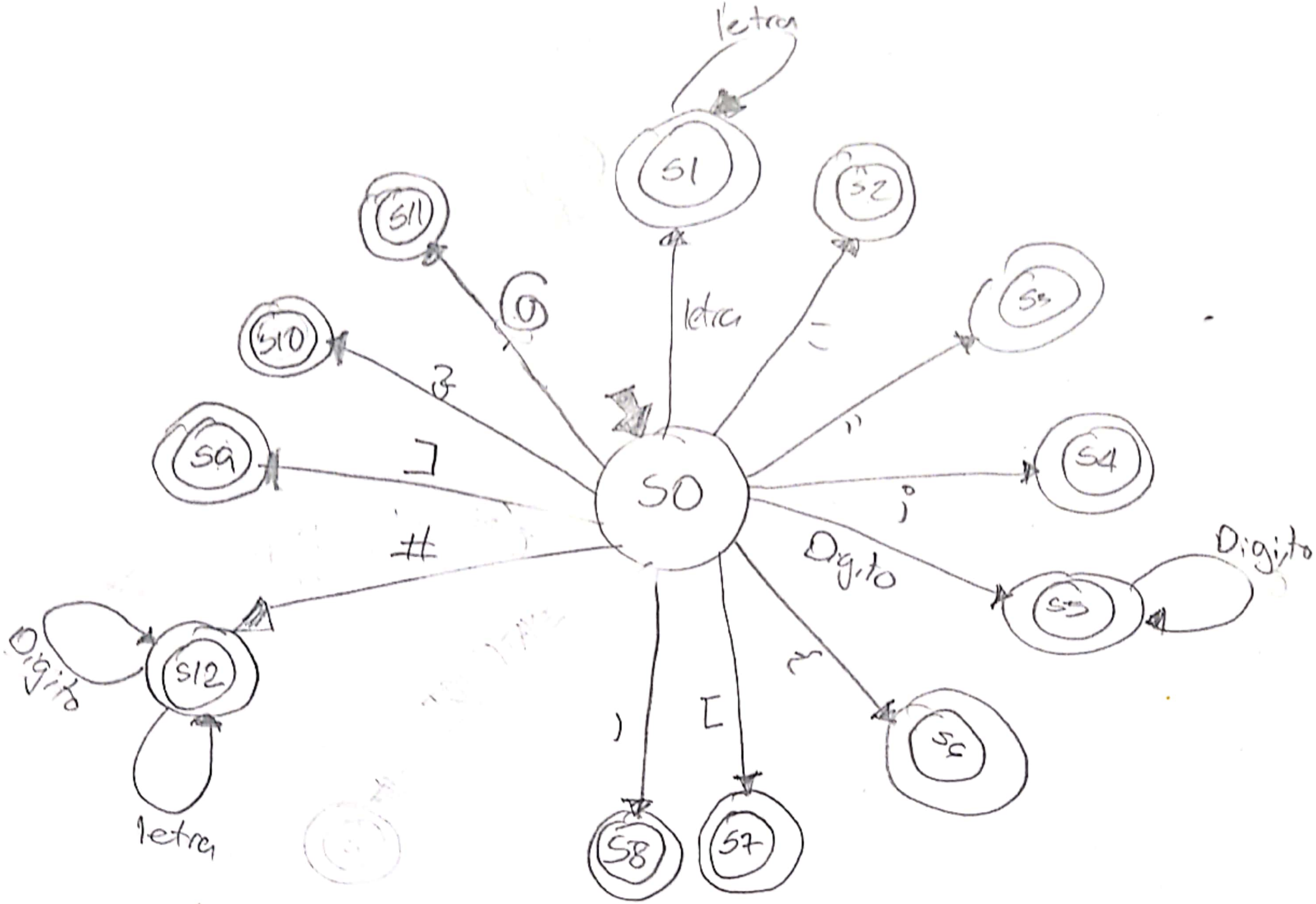
“crearhtmlmirrory”	Crea la imagen con el filtro Mirror Y solicitada por el cliente en formato .html y .png	Entrada: Diccionario con listas del analisis lexico del automata Salida: Reporte Html y su conversón a formato png de la imagen con el filtro Mirror Y solicitadea por el cliente
“crearhtmldouble”	Crea la imagen con el filtro Double Mirror solicitada por el cliente en formato .html y .png	Entrada: Diccionario con listas del analisis lexico del automata Salida: Reporte Html y su conversón a formato png de la imagen con el filtro Double Mirror solicitadea por el cliente
“crearcelda”	Crea una celda de una tabla vacia con los datos proporcionados por el cliente para generar la figura	Entrada: Diccionario con listas del analisis lexico del automata Salida: Celda de una tabla html con el color solicitado por el cliente
“crearceldavacia”	Crea una celda de una tabla con el color proporcionado por el cliente para generar la figura	Entrada: Diccionario con listas del analisis lexico del automata Salida: Celda vacia de una tabla html

EXPRESIÓN REGULAR

((letra+ | = | “ | ; | digito+ | { | [| , | boolean | #(letrashex|digitos | letrashex|digitos | letrashex|digitos | letrashex|digitos | letrashex|digitos | letrashex|digitos)|] | } | sep)+)#

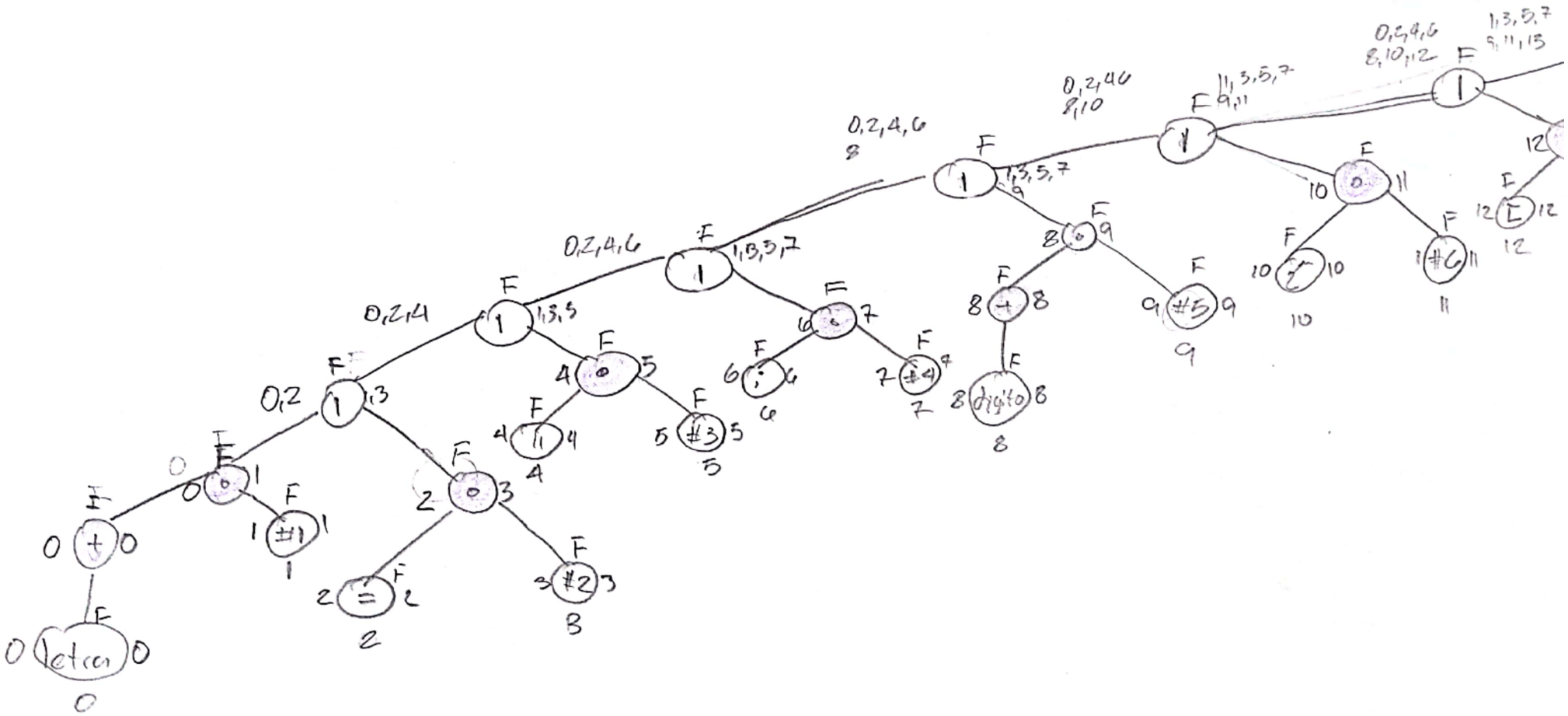
TABLA DE TOKENS

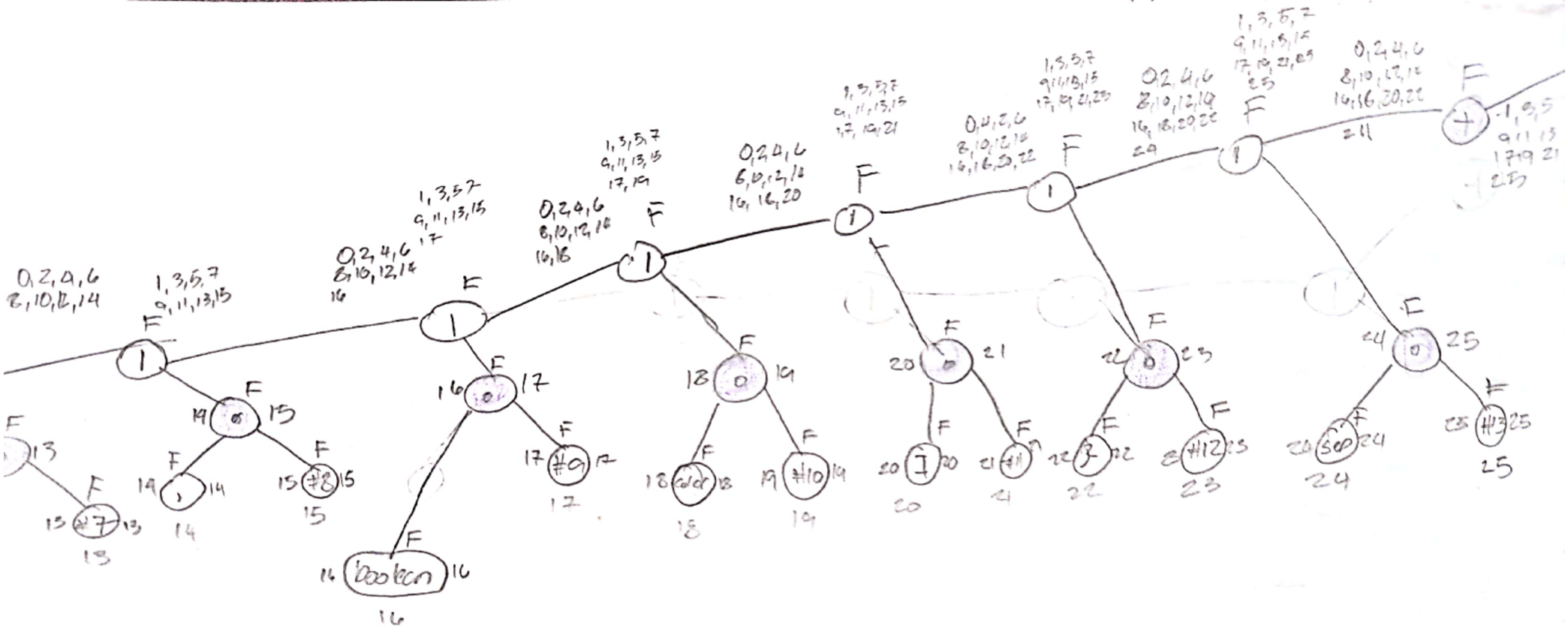
TOKEN	PATRON	ER
palabra_reservada	TITULO, ANCHO, ALTO, FILAS, COLUMNAS, CELDAS, FILTROS, MIRRORX, MIRRORY, DOUBLEMIRROR, secuencia de una o más letras, se incluyen las palabras mencionadas	letra+
signoigual	símbolo =	=
comillas	símbolo “	"
puntocoma	símbolo;	;
entero_positivo	secuencia de uno o más dígitos	digito+
llaveabierta	símbolo {	{
corcheteabierto	símbolo [	[
coma	símbolo ,	,
booleano	true or false	boolean
color	inicia con # y es seguido de letrashex y dígitos #(letrashex|dígitos letrashex|digitos | letrashex|digitos | letrashex|digitos | letrashex|digitos |letrashex|digitos)	color
corchetecerrado	símbolo]	]
llavecerrado	símbolo }	}
separador	secuencia de 4 arrobas	sep
ERROR	carácter ajeno al lenguaje	

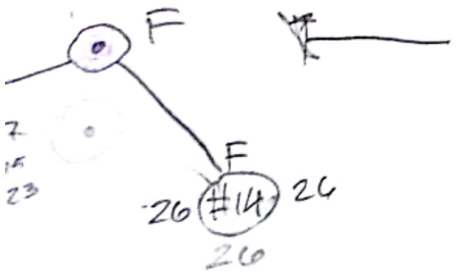

letra
S1
S2
S11
letra
S3
S10
S9
"
S0
S4
;
Digito
Digito
#
S5
S12
Digito
)
[
S6
letra
S8
S7

AFD

METODO DEL ARBOL




0,2,4,6
8,10,12,14
1,3,5,7
9,11,13,15
0,2,4,6
8,10,12,14
16
1,3,5,7
9,11,13,15
17
0,2,4,6
8,10,12,14
16,18
1,3,5,7
9,11,13,15
17,19
0,2,4,6
8,10,12,14
16,16,20
1,3,5,7
9,11,13,15
17,19,21
0,2,4,6
8,10,12,14
16,16,20,22
1,3,5,7
9,11,13,15
17,19,21,23
0,2,4,6
8,10,12,14
16,16,20,22
24
1,3,5,7
9,11,13,15
17,19,21
25
boolean
#7
#8
#9
#10
#11
#12
#13
Sop
I
3
16
17
18
19
20
21
22
23
24
25



First

0, 2, 4, 6

6 10 12 16

10 18 20 22

24

last

26