Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales y de Programación A+
Ingeniero: Otto Rodríguez
Auxiliar: Fernando Chajón

Manual De Técnico

José Manuel Ibarra Pirir
202001800
Guatemala, 20 de Marzo de 2022

Clase analizador:

```
class Analizador:
    #Guarda lo que llevo
    lexema = ''
    #Lista de token
    tokens = []
    #Estado en el que se encuentra
    estado = 1
    #Fila en la que se encuentra
    fila = 1
    #COlumna en la que se encuentra
    columna = 1
    #Bool para errores
    generar = False

    elementos = []
    entrada2 = ''
    def __init__(self, entrada):
```

Métodos

Funcionamiento del automata

```
def __init__(self, entrada):
    self.entrada2 = entrada
    self.lexema = ''
    self.tokens = []
    self.estado = 1
    self.fila = 1
    self.columna = 1
    self.generar = True
    tipos = Token("lexema",-1,-1,-1)

    entrada = entrada + '$'
    actual = ''
    longitud = len(entrada)

    i = 0
    while(i < longitud):
        actual = entrada[i]
        i += 1
        #Manejo inicial
        if self.estado == 1:
            if actual == '~':
                self.estado = 1
                self.columna += 1
                self.lexema += actual
                self.AddToken(tipos.VIRGULILLA)
            elif actual == '>':
                self.estado = 1
                self.columna += 1
                self.lexema += actual
                self.AddToken(tipos.RIGHT_ANGLE)
```

Método de agregar Token y verificar Palabras Reservadas

```
def AddToken(self,tipo):
    self.tokens.append(Token(self.lexema, tipo, self.fila, self.columna))
    self.lexema = ""
    self.estado = 1

def Reservada(self):
    palabra = self.lexema.upper();
    if palabra == 'TIPO':
        self.tipo = Token.TIPO
        return True
    if palabra == 'VALOR':
        self.tipo = Token.VALOR
        return True
    if palabra == 'FONDO':
        self.tipo = Token.FONDO
        return True
    if palabra == 'VALORES':
        self.tipo = Token.VALORES
        return True
    if palabra == 'NOMBRE':
        self.tipo = Token.NOMBRE
        return True
    if palabra == 'EVENTO':
        self.tipo = Token.EVENTO
        return True
    return False
```

Métodos para imprimir

```
def Imprimir(self):
    print("-------------Tokens--------------")
    tipos = Token("lexema", -1, -1, -1)
    for x in self.tokens:
        if x.tipo != tipos.UNKNOWN:
            print(x.getLexema()," --> ",x.getTipo(),' --> ',x.getFila(), ' --> ',x.getColumna())

def ImprimirErrores(self):
    print("-------------Errores--------------")
    tipos = Token("lexema", -1, -1, -1)
    for x in self.tokens:
        if x.tipo == tipos.UNKNOWN:
            print(x.getLexema()," --> ",x.getFila(), ' --> ',x.getColumna(),'--> Error Lexico')
```

Reporte de Token

```
def ReporteToken(self):

    messagebox.showinfo(message="Se ha genera el reporte de token", title="Reporte")
    f = open('Reporte Token.html','w')
    f.write("<!doctype html>")
    f.write("<html lang=\"en\">")
    f.write("<head>")

    f.write(" <meta charset=\"utf-8\">")
    f.write("<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">")
    f.write("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">")
    f.write("<title>Reporte del Tokens</title>")
    f.write("<style>"
        "body {background-color: #F5EFB1;font-family: \"Lucida Console\", \"Courier New\", monospace;}"
        "h1 {background-color: #87DABF;}"
        "table, th, td {border: 1px solid black; text-align: center}""</style>")
    f.write("</head>")
    f.write("<body>")
    f.write("<H1><center>REPORTE DE TOKENS</center></H1>")
    f.write("<center><table><tr><th>No. </th><th>Símbolo</th><th>Tipo</th><th>Fila</th><th>Columna</th>")
    tipos = Token("lexema", -1, -1, -1)
    i=0
    for x in self.tokens:
        i+=1
        if x.tipo != tipos.UNKNOWN:
            f.write("<tr>")
            f.write("<center><td><h4>" + str(i) + "</td></h4>"+"<td><h4>" + str(x.getLexema() ) +"</td></h4>"+
```

Reporte de Errores

```
def ReporteErrores(self):
    messagebox.showinfo(message="Se ha genera el reporte de errores", title="Reporte")
    f = open('Reporte Errores.html','w')
    f.write("<!doctype html>")
    f.write("<html lang=\"en\">")
    f.write("<head>")

    f.write(" <meta charset=\"utf-8\">")
    f.write("<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">")
    f.write("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">")
    f.write("<title>Reporte de Errores</title>")
    f.write("<style>"
        "body {background-color: #F5EFB1;font-family: \"Lucida Console\", \"Courier New\", monospace;}"
        "h1 {background-color: #87DABF;}"
        "table, th, td {border: 1px solid black; text-align: center}""</style>")
    f.write("</head>")
    f.write("<body>")
    f.write("<H1><center>REPORTE DE ERRORES</center></H1>")
    #TABLA DE PRODUCTOS ASCENDENTE
    f.write("<center><table><tr><th>No. </th><th>Símbolo</th><th>Tipo</th><th>Fila</th><th>Columna</th>")
    tipos = Token("lexema", -1, -1, -1)
    i=0
    for x in self.tokens:
        i+=1
        if x.tipo == tipos.UNKNOWN:
            f.write("<tr>")
            f.write("<center><td><h4>" + str(i) + "</td></h4>"+"<td><h4>" + str(x.getLexema() ) +"</td></h
            f.write("</tr>")
    f.write("</table></center>")
    f.write("</body>")
```

Método que guarda los datos en una lista de objetos

```
def GuardarDatos(self):
    tipos = Token("lexema", -1, -1, -1)
    longitud = len(self.tokens)
    for i in range(longitud):
        if self.tokens[i].tipo == tipos.TIPO:
            tipo = self.tokens[i+2].lexema
            tipo = str(tipo).replace('"',"")
            if tipo == "etiqueta":
                valor = None
                if self.tokens[i+4].tipo == tipos.VALOR:
                    valor = self.tokens[i+6].lexema
                    valor = str(valor).replace('"',"")
                self.elementos.append(Elemento(tipo,valor,None,None,None,None))
            if tipo == "texto":
                valor = None
                fondo = None
                if self.tokens[i+4].tipo == tipos.VALOR:
                    valor = self.tokens[i+6].lexema
                    valor = str(valor).replace('"',"")
                if self.tokens[i+4].tipo == tipos.FONDO:
                    fondo = self.tokens[i+6].lexema
                    fondo = str(fondo).replace('"',"")
                if self.tokens[i+8].tipo == tipos.VALOR:
                    valor = self.tokens[i+10].lexema
                    valor = str(valor).replace('"',"")
                if self.tokens[i+8].tipo == tipos.FONDO:
                    fondo = self.tokens[i+10].lexema
                    fondo = str(fondo).replace('"',"")
                self.elementos.append(Elemento(tipo,valor,fondo,None,None,None))
```

Método que genera el formulario en HTML

```
def generarFormulario1(self):
    tipos = Token("lexema", -1, -1, -1)
    longitud = len(self.tokens)
    for i in range(longitud):
        if self.tokens[i].tipo == tipos.UNKNOWN:
            messagebox.showinfo(message="No se ha generado el Formulario", title="Formularios.io")
            generar = False
            break
        else:
            generar = True
    if generar:
        messagebox.showinfo(message="Se ha genera el Formulario", title="Formularios.io")
        f = open('Formulario.html','w')
        f.write("<!doctype html>")
        f.write("<html lang=\"en\">")
        f.write("<head>")

        f.write(" <meta charset=\"utf-8\">")

        f.write("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">")
        f.write("<title>Formulario</title>")
        f.write("<style>"
            "@import url('https://fonts.googleapis.com/css?family=Poppins&display=swap');"

            "* {"
            "box-sizing: border-box;"
            "}"
            "body {background-color: #edeef6;font-family: 'Poppins', sans-serif;display: flex;align-items: center;justify-c
            "h1 {background-color: #87DABF;}"
            "table, th, td {border: 1px solid black; text-align: center}"
            "form {margin: 0 auto; width: 400px;padding: 1em;border: 1px solid #CCC; border-radius: 1em;}"
```

Interfaz gráfica

```
def boton_cargaArchivo_command():
    texto = Lector_Archivos()
    text_area.insert(tk.INSERT,texto)


def boton_analizar_command():
    texto = text_area.get(1.0,'end')
    lexico = Analizador(texto)
    lexico.Imprimir()
    lexico.ImprimirErrores()
    lexico.GuardarDatos()
    lexico.generarFormulario1()



def boton_aceptarTexto_command():
    Tk().withdraw()
    filedir = filedialog.askopenfilename(filetypes=[("Archivo data","*.form")])
    #print(direccion)
    texto = text_area.get(1.0,'end')
    #print(texto)
    with open(filedir,"r+",encoding = "utf-8") as f:
        f.truncate(0)
        f.write(texto)

def boton_buscar_reporte():
    opcion = lista_reportes.get()
    if opcion == "Reporte de token":
        texto = text_area.get(1.0,'end')
        lexico = Analizador(texto)
```

Clase elemento

```
class Elemento:

    def __init__(self,tipo, valor, fondo, valores, evento, nombre):
        self.tipo = tipo
        self.valor = valor
        self.fondo = fondo
        self.valores = valores
        self.evento = evento
        self.nombre = nombre


    def __repr__(self):
        return f'\n Tipo {self.tipo} Valor {self.valor} Fondo {self.fondo} Valores {self.valores} Evento {self.evento} Nombre {self
```

Clase de Token

```python
class Token():
    lexema = ''
    tipo = 0
    fila = 0
    columna = 0

    #ENUM
    VIRGULILLA = 1
    RIGHT_ANGLE = 2
    LEFT_ANGLE = 3
    CORCHETE_IZQ = 4
    CORCHETE_DER = 5
    COLON = 6
    COMMMA = 7
    CHAIN = 8
    SIMPLE_CHAIN = 9
    TIPO = 10
    VALOR = 11
    FONDO = 12
    VALORES = 13
    EVENTO = 14
    WORDS = 15
    UNKNOWN = 16
    NOMBRE = 17

    #Constructor de la clase
    def __init__(self,lexema,tipo,fila,columna):
        self.lexema = lexema
        self.tipo = tipo
        self.fila = fila
        self.columna = columna
```

Script en Javascript

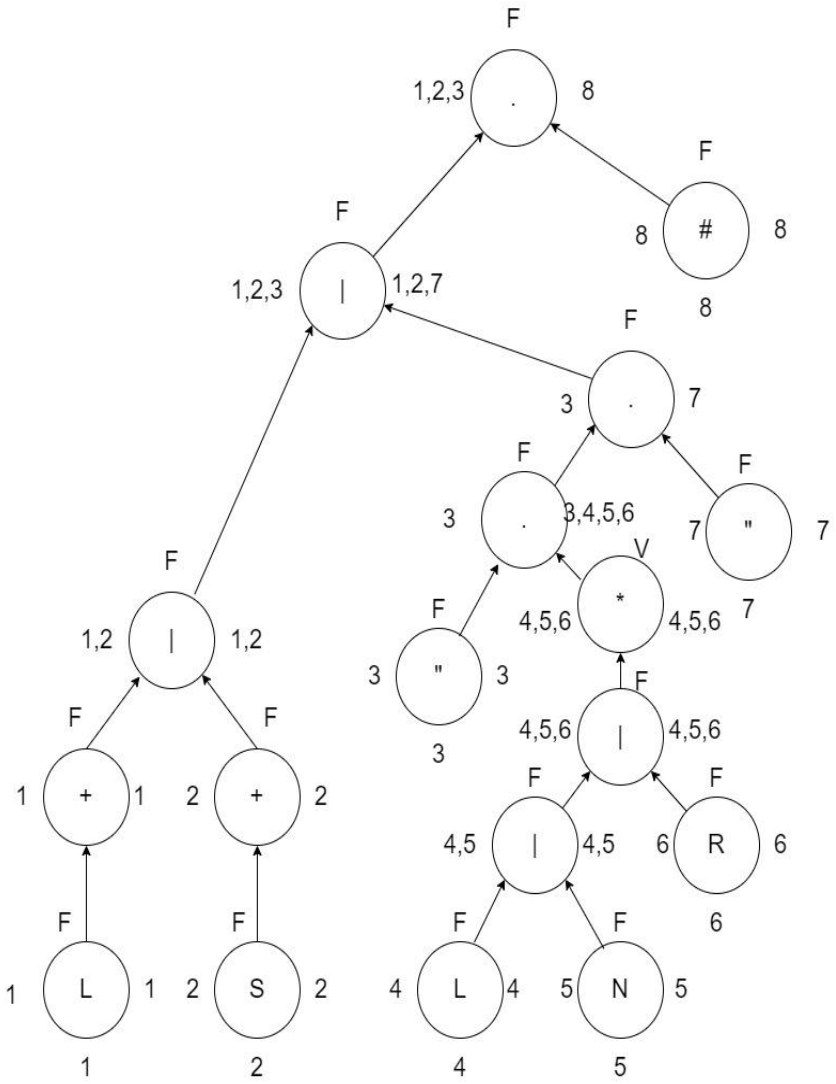
```
cto 1 LFP > JS app.js > ...
const open = document.getElementById('open');
const modal_container = document.getElementById('modal_container');
const close = document.getElementById('close');

open.addEventListener('click', () => {
  modal_container.classList.add('show');
});

close.addEventListener('click', () => {
  modal_container.classList.remove('show');
});
```

Expresiones regulares utilizadas

Símbolos = S = {~.<,>,[,],:,coma}
Números = [0-9]
Letras = L = {a-z,A-Z}
Resto = R = Todos los símbolos que no pertenecen a R
Cadena = C = “(L|N|R)”

Expresión regular Utilizada

ER = [(L)+|(S)+|”(L|N|R)*”]#



No	Terminales	Siguiente
1	L	8
2	S	8
3	"	4,5,6,7
4	L	4,5,6,7
5	N	4,5,6,7
6	R	4,5,6,7
7	"	8
8	#	

So = {1,2,3} -> Sig(L) = Sig(1) = {8} = S1
Sig(S) = Sig(2) = {8} = S1
Sig(“) = Sig(3) = {4,5,6,7} = S2

S1 = {8}->

S2 = {4,5,6,7}-> Sig(L) = Sig(4) = {4,5,6,7} = S2
Sig(N) = Sig(5) = {4,5,6,7} = S2
Sig(R) = Sig(6) = {4,5,6,7} = S2
Sig(“) = Sig(7) = {8} = S2