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 Técnico

José Manuel Ibarra Pirir
202001800
Guatemala, 28 de Abril de 2022

Clase Analizador

```
class Analizador:
    tipo = TypeToken.UNKNOWN
    #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 = 0
    #Bool para errores
    generar = False

    def __init__(self, entrada):
        self.lexema = ''
        self.tokens = []
        self.estado = 1
        self.fila = 1
        self.columna = 0
        self.generar = True
        tipos = Token("lexema",-1,-1,-1)

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

        i = 0
```

Funcionamiento del sistema

Funcionamiento Autómata

```
i = 0
while(i < longitud):
    actual = entrada[i]
    i += 1
    #Manejo inicial
    if self.estado == 1:
        if actual.isalpha():
            self.estado = 2
            self.columna += 1
            self.lexema += actual
            continue
        elif actual.isdigit():
            self.estado = 3
            self.columna += 1
            self.lexema += actual
            continue
        elif actual == "<":
            self.estado = 4
            self.columna += 1
            self.lexema += actual
            continue
        elif actual == '"':
            self.estado = 5
```

AddToken

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

Método de Banderas

```
def Banderas(self):
    bandera = self.lexema
    if bandera == "-f":
        self.tipo = TypeToken.F.name
        return True
    if bandera == "-ji":
        self.tipo = TypeToken.JI.name
        return True
    if bandera == "-jf":
        self.tipo = TypeToken.JF.name
        return True
    if bandera == "-n":
        self.tipo = TypeToken.N.name
        return True
    return False
```

Método de Reservadas

```
def Reservada(self):
    palabra = self.lexema.upper();
    if palabra == 'RESULTADO':
        self.tipo = TypeToken.RESULTADO.name
        return True
    if palabra == 'VS':
        self.tipo = TypeToken.VS.name
        return True
    if palabra == 'TEMPORADA':
        self.tipo = TypeToken.TEMPORADA.name
        return True
    if palabra == 'JORNADA':
        self.tipo = TypeToken.JORNADA.name
        return True
    if palabra == 'GOLES':
        self.tipo = TypeToken.GOLES.name
        return True
    if palabra == 'LOCAL':
        self.tipo = TypeToken.CONDICION_GOLES.name
        return True
    if palabra == 'VISITANTE':
        self.tipo = TypeToken.CONDICION_GOLES.name
        return True
    if palabra == 'TOTAL':
        self.tipo = TypeToken.CONDICION_GOLES.name
        return True
    if palabra == 'TABLA':
        self.tipo = TypeToken.TABLA_TEMPORADA.name
        return True
```

Impresión de Errores y Token

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

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

Reporte 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 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 ERRORES LEXICOS</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 str(x.tipo) == "UNKNOWN":
            f.write("<tr>")
```

Reporte 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 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 str(x.tipo) != "UNKNOWN":
            f.write("<tr>")
            f.write("<center><td><h4>" + str(i) + "</td></h4>"+"<td><h4>" + str(x.lexema ) +"</td></h4>"+"<
            f.write("</tr>")
    f.write("</table></center>")
```

Limpiar Errores

```
def limpiarErrores(self):
    messagebox.showinfo(message="Se ha limpiado el log de Errores", title="Reporte")
    self.tokens.clear()

def limpiarTokens(self):
    messagebox.showinfo(message="Se ha limpiado el log de Tokens", title="Reporte")
    self.tokens.clear()
```

Sintético

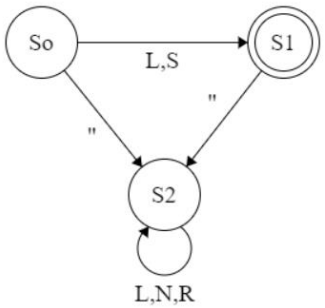
```
class Sintactico:
    preanalisis = TypeToken.UNKNOWN
    pos = 0
    lista = []
    errorSintactico = False

    def __init__(self, lista):
        self.errorSintactico = False
        self.lista = lista
        self.lista.append(Token("#", TypeToken.ULTIMO, 0, 0))
        self.pos = 0
        self.preanalisis = self.lista[self.pos].tipo
        self.Inicio()

    def Match(self, type):
        if self.preanalisis != type:
            print(str(self.lista[self.pos].tipo), "--- Sintactico", "--- Se esperaba "+str(type))
            self.errorSintactico = True
        if self.preanalisis != TypeToken.ULTIMO.name:
            self.pos += 1
            self.preanalisis = self.lista[self.pos].tipo
        if self.preanalisis == TypeToken.ULTIMO.name:
            print("Se ha finalizado el Análisis Sintáctico")

    def Inicio(self):
        print("----- Inicio Análisis Sintactico -----")
        if TypeToken.RESULTADO.name == self.preanalisis:
            self.Resultado()
```

```
def Resultado(self):
    self.Match(TypeToken.RESULTADO.name)
    self.Match(TypeToken.EQUIPO.name)
    self.Match(TypeToken.VS.name)
    self.Match(TypeToken.EQUIPO.name)
    self.Match(TypeToken.TEMPORADA.name)
    self.Match(TypeToken.FECHA.name)

def Jornada(self):
    self.Match(TypeToken.JORNADA.name)
    self.Match(TypeToken.NUMERO.name)
    self.Match(TypeToken.TEMPORADA.name)
    self.Match(TypeToken.FECHA.name)
    self.Match(TypeToken.F.name)
    self.Match(TypeToken.WORDS.name)

def Goles(self):
    self.Match(TypeToken.GOLES.name)
    self.Match(TypeToken.CONDICION_GOLES.name)
    self.Match(TypeToken.EQUIPO.name)
    self.Match(TypeToken.TEMPORADA.name)
    self.Match(TypeToken.FECHA.name)

def Tabla_Temporada(self):
    self.Match(TypeToken.TABLA_TEMPORADA.name)
    self.Match(TypeToken.TEMPORADA.name)
    self.Match(TypeToken.FECHA.name)
    self.Match(TypeToken.F.name)
    self.Match(TypeToken.WORDS.name)
```

Tabla de Token

```
RESULTADO = 1
VS = 2
TEMPORADA = 3
JORNADA = 4
F = 5
GOLES = 6
LOCAL = 7
VISITANTE = 8
TOTAL = 9
TABLA_TEMPORADA = 10
PARTIDOS = 11
JI = 12
JF = 13
TOP = 14
N = 15
SUPERIOR = 16
INFERIOR = 17
ADIOS = 18
EQUIPO = 19
NUMERO = 20
FECHA = 21
WORDS = 22
UNKNOWN = 23
CONDICION_GOLES = 24
CONDICION_TOP = 25
ULTIMO = 26
```

Automata Finito



GLC

<Inicio>::= <Resultado> <Repetir>

| <Jornada> <Repetir>

| <Goles> <Repetir>

| <Tabla Temporada> <Repetir>

| <Partidos> <Repetir>

| <Top> <Repetir>

| <Adios> <Repetir>

<Repetir>::= <Resultado> <Repetir>

| <Jornada> <Repetir>

| <Goles> <Repetir>

| <Tabla Temporada> <Repetir>

| <Partidos> <Repetir>

| <Top> <Repetir>

| <Adios> <Repetir>

| Epsilon

<Resultado>::= tk_Resultado tk_Equipo tk_VS tk_Equipo tk_Temporada tk_Fecha

<Jornada>::= tk_Jornada tk_Numero tk_Temporada tk_Fecha tk_FArchivo tk_words

<Goles>::= tk_Goles <CondionGoles> tk_Equipo tk_Temporada tk_Fecha

<CondionGoles>::= tk_Local

| tk_Visitante

| tk_Total

<Tabla Temporada>::= tk_TablaTemporada tk_Temporada tk_Fecha tk_FArchivo tk_words

<Partidos>::= tk_Partidos tk_Equipo tk_Temporada tk_Fecha tk_FArchivo tk_words tk_JI tk_Numero tK_JF tk_Numero

<Top>::= tk_Top <CondicionTop> tk_Temporada tk_Fecha tk_N tk_Numero

<CondicionTop>::= tk_Superior

| tk_Inferior

<Adios>::= tk_Adios