# MANUAL TECNICO
# PROYECTO #2

Luis Manuel Chay Marroquín

202000343

Huehuetenango, 23 de septiembre de 2021

## DESCRIPCION GENERAL

El siguiente programa fue diseñado para el almacenamiento, análisis y realización de reportes de datos para la toma de ediciones futuras en cualquier tipo de negocio, permitiendo mostrar en una interfaz grafica reportes de un distinto conjunto de operaciones básicas y exportar estos reportes en formato .html, todo a través de la carga de un archivo de texto de formato .lfp

## DATOS TECNICOS

LENGUAJE UTILIZADO

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

IDE UTILIZADO

Atom

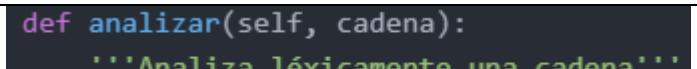
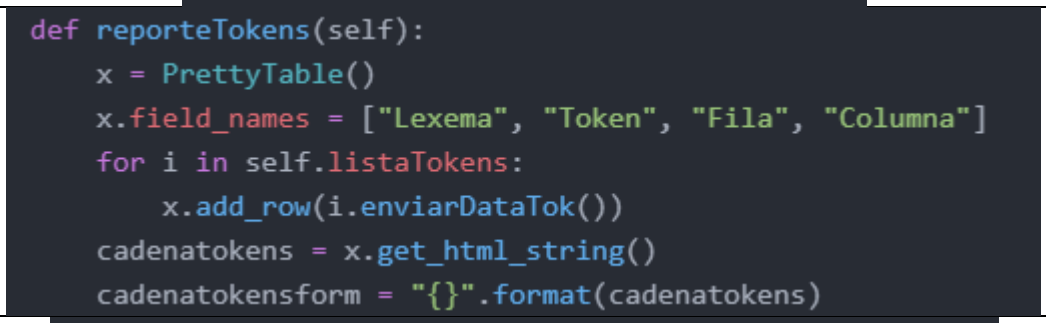
SISTEMA OPERATIVO

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

Analizador Léxico

| METODO | DESCRIPCION |
|---|---|
| “init” | `def __init__(self):`<br>`  self.listaTokens = []`<br>`  self.listaErrores = []`<br>`  self.linea = 1`<br>`  self.columna = 1`<br>`  self.buffer = ''`<br>`  self.estado = 0`<br>`  self.i = 0` |
| “agregar_token” | `def agregar_token(self,caracter,token,linea,columna):`<br>`  self.listaTokens.append(Token(caracter,token,linea,columna))`<br>`  self.buffer = ''` |
| “agregar_error” | `def agregar_error(self,caracter,linea,columna):`<br>`  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):` |
| “estado14” | `def estado14(self,caracter):` |
| “estado15” | `def estado15(self,caracter):` |

| | |
|---|---|
| “analizar” |  |
| “reporteTokens” |  |
| “reporteErrores” |  |
| “leerArchivo” |  |

```
def analizar(self, cadena):
    '''Analiza léxicamente una cadena'''
```

```
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 LFP", "*.lfp"), ("All Files", "*.*")])
    archivo = open(entrada, 'r')
    global contenido
    contenido = archivo.read()
    Text1.insert(END, contenido)
    archivo.close()
```

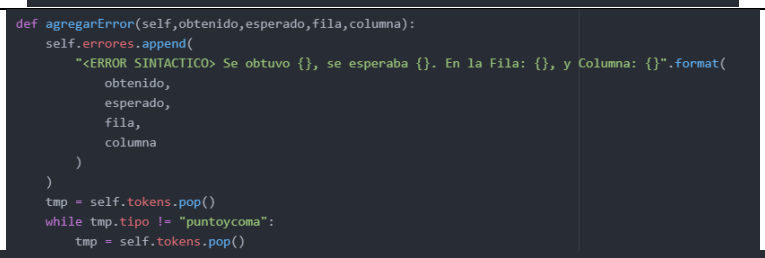
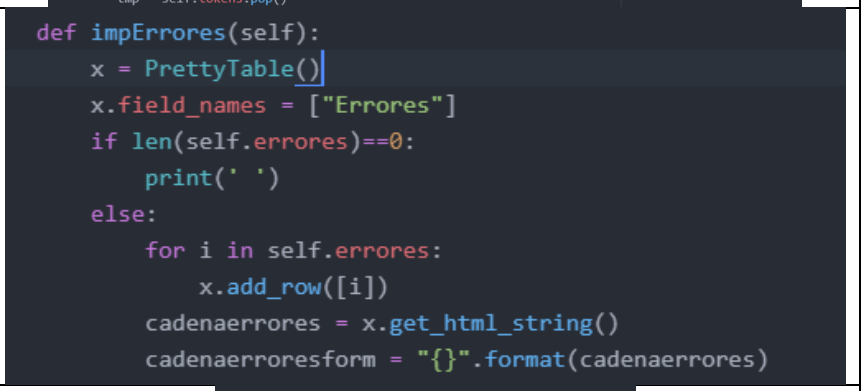
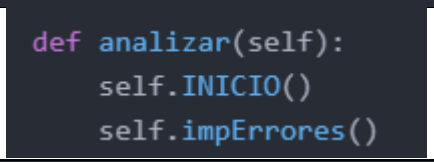
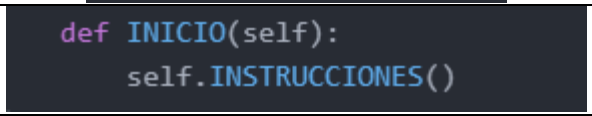
| | |
|---|---|
| “analisis” | |

```
def analisis():
    #Analisis lexico
    scanner = AnalisisLexico()
    scanner.analizar(Text1.get(1.0, END))
    scanner.reporteTokens()
    scanner.reporteErrores()
    #Analisis sintactico
    sintactico = AnalizadorSintactico(scanner.listaTokens)
    sintactico.analizar()
    Text2.insert(tk.INSERT, sintactico.cadenaprint)
    Text2.insert(tk.INSERT, sintactico.cadenaprintln)
    Text2.insert(tk.INSERT, sintactico.activador)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
    Text2.insert(tk.INSERT, sintactico.variablefloat)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
    Text2.insert(tk.INSERT, sintactico.activadordatos)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
    Text2.insert(tk.INSERT, sintactico.variablesuma)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
    Text2.insert(tk.INSERT, sintactico.variablemax)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
    Text2.insert(tk.INSERT, sintactico.variable2)
    Text2.insert(tk.INSERT, sintactico.saltolinea)
```

| | |
|---|---|
| “double” | |

```
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()
```

Analizador Sintáctico

| METODO | DESCRIPCION |
| --- | --- |
| “init” |  |
| “agregar_error” |  |
| “impErrores” |  |
| “analizar” |  |
| “INICIO” |  |

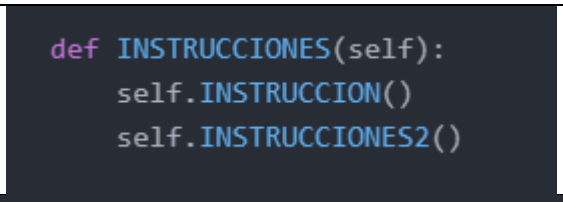
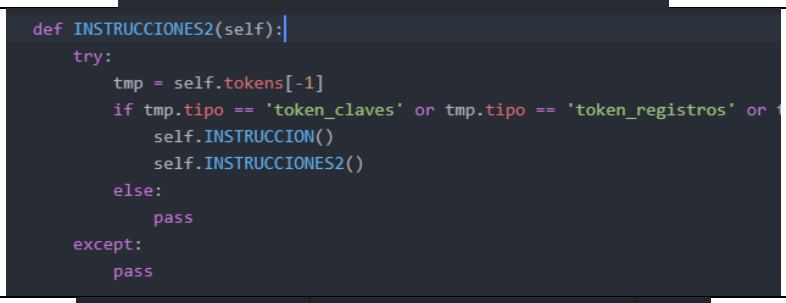
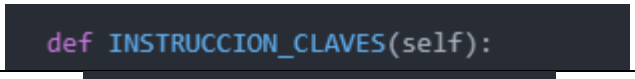
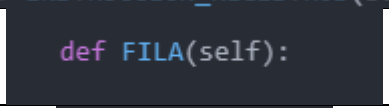
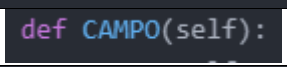
```
def __init__(self,tokens = []):
    self.errores = []
    self.cadenaprint = ""
    self.cadenaprintln = ""
    self.cadenapromedio = ""
    self.cadenasumar = ""
    self.cadenamax = ""
    self.cadenamin = ""
    self.datos = {}
    self.registrosaux = []
    self.contadorreg = 0
    self.activador = 0
    self.variablefloat = 0.00
    self.tokens = tokens
    self.saltolinea = '\n'
    self.activadordatos = {}
    self.variablesuma = 0.00
    self.variablemax = 0.00
    self.variable2  = 0.00
    #Para sacar el elemento esperado le doy vuelta a la lista de tokens
    self.tokens.reverse()
    self.tabla = {}
```

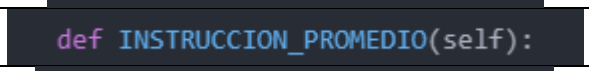
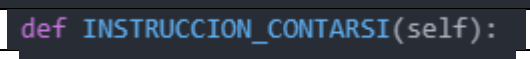
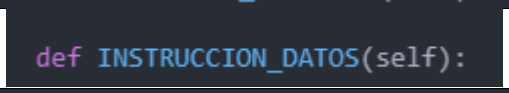
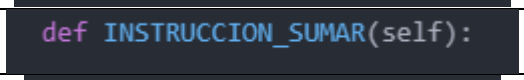
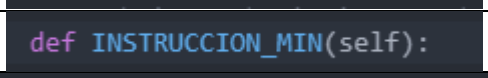
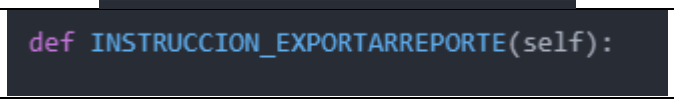
```
def agregarError(self,obtenido,esperado,fila,columna):
    self.errores.append(
        "<ERROR SINTACTICO> Se obtuvo {}, se esperaba {}. En la Fila: {}, y Columna: {}".format(
            obtenido,
            esperado,
            fila,
            columna
        )
    )
    tmp = self.tokens.pop()
    while tmp.tipo != "puntoycoma":
        tmp = self.tokens.pop()
```

```
def impErrores(self):
    x = PrettyTable()
    x.field_names = ["Errores"]
    if len(self.errores)==0:
        print(' ')
    else:
        for i in self.errores:
            x.add_row([i])
        cadenaerrores = x.get_html_string()
        cadenaerroresform = "{}".format(cadenaerrores)
```

```
def analizar(self):
    self.INICIO()
    self.impErrores()
```

```
def INICIO(self):
    self.INSTRUCCIONES()
```

| | |
|---|---|
| “INSTRUCCIONES” |  |
| “INSTRUCCIONES2” |  |
| “INSTRUCCION” |  |
| “INSTRUCCIÓN_CLAVES” |  |
| “LISTACLAVES” |  |
| “CLAVES” |  |
| “INSTRUCCIÓN_REGISTROS” |  |
| “FILA” |  |
| “CAMPO” |  |
| “INSTRUCCIÓN_IMPRIMIR” |  |

| | |
|---|---|
| “INSTRUCCIÓN_IMPRIMIRLN” |  |
| “INSTRUCCIÓN_CONTEO” |  |
| “INSTRUCCIÓN_PROMEDIO” |  |
| “INSTRUCCIÓN_CONTARSI” |  |
| “INSTRUCCIÓN_DATOS” |  |
| “INSTRUCCIÓN_SUMAR” |  |
| “INSTRUCCIÓN_MAX” |  |
| “INSTRUCCIÓN_MIN” |  |
| “INSTRUCCIÓN_EXPORTAREPORTE” |  |

**METODOS MAS IMPORTANTES**

Análisis Léxico

| METODO | DESCRIPCION | PARAMETROS |
|---|---|---|
| “init” | Inicializa los datos necesarios para la ejecución del autómata | Salida:<br>listaTokens: Lista de los tokens vacía<br>listaErrores Lista de los errores léxicos vacía<br>línea: línea inicial para el análisis del carácter<br>columna: columna inicial para el análisis del carácter<br>buffer: buffer vacío<br>estado: 0 (estado inicial del autómata)<br>i = 0 valor inicial del contador |
| “agregar_token” | Agrega un token a la lista de Tokens | Salida:<br>llistaTokens: Lista de los tokens con uno o más errores |
| “agregar_error” | Agrega un error léxico a la lista de Errores | Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token<br>llistaErrores: Lista de los errores con uno o más errores<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token |
| “estado3” | Analiza si el carácter es un signo “ necesario para el funcionamiento de la aplicación | Entrada:<br>caracter: Caracteres en el archivo de entrada .lfp |

| | | |
|---|---|---|
| | agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis | Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token |
| “estado6” | Analiza si el carácter es un decimal necesario para el funcionamiento de la aplicación agregándolo a la lista de tokens y regresa al estado 0 para continuar con el análisis | Entrada:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada ..lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token |
| “estado11” | 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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token |

| | | |
|---|---|---|
| “estado12” | 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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>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:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token |
| “estado14” | Analiza si el carácter es un simbolo # e ignora todo lo siguiente a este símbolo y regresa al estado 0 para continuar con el análisis | Entrada:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>Nada |
| “estado15” | Analiza si el carácter es un simbolo ‘ e ignora todo lo siguiente a este símbolo y regresa al estado 0 para continuar con el análisis | Entrada:<br>caracter: Caracteres en el archivo de entrada .lfp<br>Salida:<br>Nada |
| “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:<br>caracter: Caracteres en el archivo de entrada .pxla<br>listaTokens: Lista de los tokens vacía<br>listaErrores Lista de los errores léxicos vacía<br>centinela: carácter $ que se agrega al final de la cadena<br>Salida:<br>llistaTokens: Lista de los tokens con un nuevo token<br>llistaErrores: Lista de los errores con uno o más errores<br>Cambios de estado |
| “reporteTokens” | Crea un reporte HTML a partir de la lista de Tokens creada en el analisis | Entrada:<br>listaTokens: Lista de los tokens vacía<br>Salida:<br>Reporte de tokens en formato .htrml |
| “reporteErrores” | Crea un reporte HTML a partir de la lista de errores creada en el analisis | Entrada:<br>listaErrores: Lista de los errores vacía<br>Salida:<br>Reporte de errorres en formato .htrml |
| “leerArchivo” | Abre un explorador de archivos para seleccionar el archivo .pxla | Entrada:<br>Archivo de extensión .pxla |

| | | |
|---|---|---|
| | de entrada ncesario para el funcionamiento del programa | Salida:<br>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:<br>contenido: Contenido del archvio de entrada<br>Salida:<br>Diccionario: Un diccionario nativo de Python con las listas del contenido analizado por el automata |

Análisis Sintáctico

| METODO | DESCRIPCION | PARAMETROS |
|---|---|---|
| “init” | Inicializa los datos necesarios para la ejecución del autómata | Salida:<br>listaTokens: Lista de los tokens vacía y al revés<br>listaErrores Lista de los errores léxicos vacía<br>datos: Diccionario vacio<br>registrosaux: lista auxiliar vacia<br>línea: línea inicial para el análisis del carácter<br>cadenavacia: cadenas vacias que se llenaran con los resultados<br>variablevacia: variable de tipo float que se llenaran con los resultados |
| “agregar_error” | Agrega un error sintactico a la lista de Errores | Salida:<br>llistaErrores: Lista de los errores con uno o más errores |
| “imprimir_error” | Crea un reporte en formato HTML de los errores sintácticos | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Reporte de tokens en formato .htrml |
| “analizar” | Inicializa el analizador sintáctico o muestra errores sintacticos | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambio a instrucción INICIO o impErrores |

| | | |
|---|---|---|
| “INICIO” | Cambia a la instrucción INSTRUCCIONES | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambio a instrucción INICIO o impErrores<br>Salida:<br>Cambia a instrucción INSTRUCCIONES |
| “INSTRUCCIONES” | Cambia a la instrucción INSTRUCCIÓN o INSTRUCCIONES2 | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambia a instrucción INSTRUCCIÓN O INSTRUCCION2 |
| “INSTRUCCIONES2” | Analiza un token de la lista y si coincide con alguna de las claves establecidas cambia a la instrucción INSTRUCCIÓN o INSTRUCCIONES2 | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambia a instrucción INSTRUCCIÓN O INSTRUCCION2 |
| “INSTRUCCIÓN” | Analiza un token de la lista y si coincide con alguna de las claves establecidas cambia a la instrucción requerida para crear un reporte | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambia a instrucción requerida |
| “INSTRUCCION_CLAVES” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción LISTACLAVES y/o ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambia a instrucción requerida |
| “CLAVES” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción LISTACLAVES y/o ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Cambio de instrucción requerida y guarda las claves |
| “LISTACLAVES” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción CLAVES y/o ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida: |

| | | |
|---|---|---|
| | | Cambio de instrucción requerida y guarda las claves |
| “INSTRUCCIÓN _REGISTROS” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción FILA y/o ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Cambia a instrucción requerida y guarda los registros |
| “FILA” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción CAMPO y/o ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Cambia a instrucción requerida y guarda los registros |
| “CAMPO” | Analiza los tokens de la lista si su orden es igual al de la GLC y cambia a la instrucción CAMPO y/o FILA y/o ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Cambia a instrucción requerida y guarda los registros en la lista auxiliar |
| “INSTRUCCIÓN_IMPRIMIR” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Imprime una cadena en consola |
| “INSTRUCCIÓN_IMPRIMIRLN” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Imprime una cadena en consola con un salto de linea |
| “CONTEO” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: Cuenta la cantidad de registros y los pone en consola |
| “PROMEDIO” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada: listaTokens: Lista de los tokens llena Salida: |

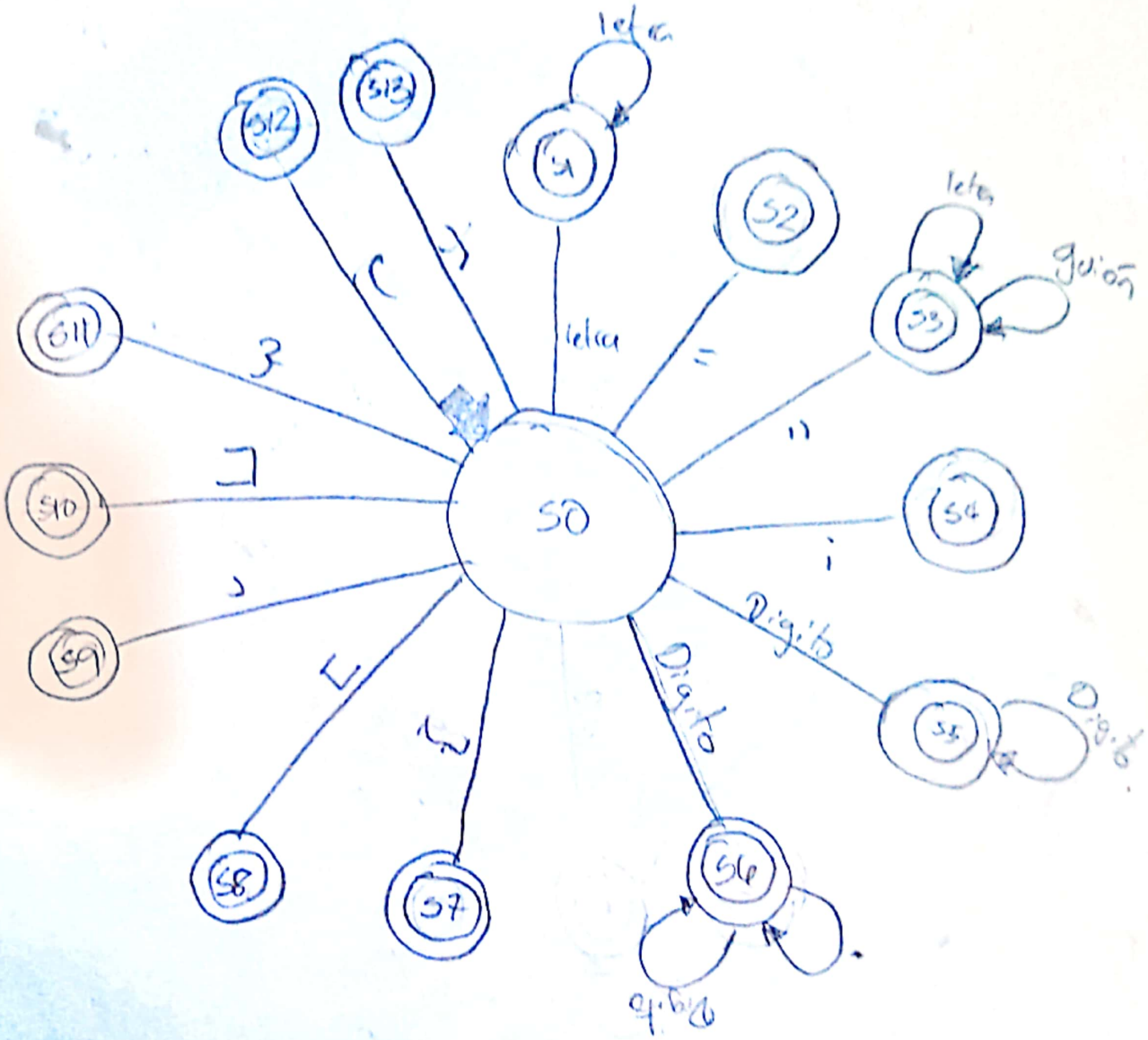
| | | |
|---|---|---|
| | | Cuenta la cantidad de registros y los pone en consola |
| “CONTARSI” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Imprime por consola la cantidad de registros en la que el campo dado sea igual al valor dado |
| “DATOS” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Imprime por consola los datos guardados |
| “SUMAR” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Imprime por consola la suma de los registros dada una clave |
| “MAX” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Imprime por consola el máximo de los registros dada una clave |
| “MIN” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Imprime por consola el mínimo de los registros dada una clave |
| “EXPORTARREPORTE” | Analiza los tokens de la lista si su orden es igual al de la GLC ejecuta la acción | Entrada:<br>listaTokens: Lista de los tokens llena<br>Salida:<br>Crea un reporte en HTML de los registros |

**EXPRESIÓN REGULAR**

(( letra+ | = | “ | ; | digito+ | digito+ ‘.’ digito+ | { | [ | , | ] | } | ( | ‘)’ )+ )#

**TABLA DE TOKENS**

| TOKEN | PATRON | ER |
|---|---|---|
| palabra_reservada | CLAVES, REGISTROS, imprimir, imprimirln, conteo, promedio, contarsi, datos, sumar, max, min, exportarReporte, secuencia de una o mas letras, se incluyen las palabras mencionadas | letra+ |
| signoigual | símbolo = | = |
| comillas | símbolo “ | " |
| puntocoma | símbolo; | ; |
| entero_positivo | secuencia de uno o más dígitos | digito+ |
| decimal_positivo | numero decimal | digito+ ‘.’ digito+ |
| llaveabierta | símbolo { | { |
| corcheteabierto | símbolo [ | [ |
| coma | símbolo , | , |
| corchetecerrado | símbolo ] | ] |
| llavecerrado | símbolo } | } |
| parentesisabierto | símbolo ( | ( |
| parentesiscerrado | símbolo ) | ) |
| ERROR | carácter ajeno al lenguaje | |




AFD