Proyecto 2

Guatemala, 22 de octubre del 2021

Índice de contenido

1 Definición de la solución

1.1 Objetivos

Que el estudiante:

- Desarrolle una solución de software implementando un analizador léxico
- mediante autómatas.
- Aplique los conocimientos adquiridos en el laboratorio, del lenguaje de
- programación Python
- Desarrolle una interfaz gráfica utilizando el lenguaje Python.

1.2 Alcances del proyecto

- Los entregables para este proyecto se adjunta en el repositorio de Git Hub con toda su documentación, este proyecto de sube a un repositorio privado, para que en un futuro los desarrolladores se le pueda dar mantenimiento al sistema, con toda su documentación.
- El sistema está diseñado para ser ejecutado en consola y por interfaz grafica, utilizando archivos de entrada.
- El sistema permite realizar ciertas operaciones de interés, las cuales son:
    - imprimir: Imprime un String en consola.
    - imprimirln: Imprime un String en consola con salto de linea.
    - conteo: cantidad de registros.
    - promedio: promedio de un campo especifico.
    - Contar si: cuenta las veces que un valor se repite para un campo especifico.
    - Datos: imprime tabla con datos.
    - Sumar: Suma de datos de un campo dado.
    - Max: Valor máximo de un campo dado.
    - Min: Valor mínimo de un campo dado.
    - Exportar reporte: Exportar reporte en HTML.
- La información es almacenada en cada ejecución en la memoria RAM.
- El sistema es capaz de generar reportes en HTML con todos los datos obtenidos.

1.3 Requerimientos del sistema

1.3.1 Requerimientos funcionales

- **Abrir archivo**: Carga un archivo de texto a un panel de texto en el sistema, de tal forma que puede ser editado
- **Analizar archivo**: Lee el texto en el sistema y realiza los análisis léxico y sintáctico.
- **Reporte en HTML**: Reporta el listado de tokens, errores léxicos y sintácticos.
- **Ejecutar**: Ejecuta las operaciones del archivo.

1.3.2 Descripción

Se le solicita a usted como estudiante de ingeniería en ciencias y sistemas una solución de software que permita analizar y realizar reportes de datos para la toma de decisiones futuras y que además se pueda aplicar a cualquier tipo de negocio, generando información interesante para pequeñas empresas. Esta solución debe tener como entrada un lenguaje con cierta estructura por medio de un archivo con extensión ".lfp".

1.3.3 Descripción de lenguajes

1.3.3.1 Importación de datos

La importación de datos, los cuáles se utilizarán más adelante en los reportes viene declarada en dos partes:
Sección de Claves: En esta sección se declaran los claves o campos por los que están construidos los registros, su estructura está formada por la palabra reservada Claves, seguido de signo igual, corchete de apertura, lista de claves y corchete de cierre.

Lista de claves está formada por cadenas de caracteres encerradas entre comillas y separadas por coma.

```
Claves = [
    "clave_1", "clave_2", "clave_3", "clave_4"
]
```

```
Claves = [
    "codigo", "producto", "precio_compra",
    "precio_venta", "stock"
]
```

Sección de Registros: En esta sección se detallan los registros que se quieren analizar y sigue la estructura dada por palabra reservada Registros, signo igual, corchete de apertura, lista de registros y corchete de cierre.

Lista de registros: Cada registro está encerrado entre llave de apertura y llave de cierre y sus valores están separados por comas, estos valores pueden ser cadenas de texto, enteros o decimales.

```
Registros = [
    {valor1, valor2, valor3, valor4}
    {valor1, valor2, valor3, valor4}
    {valor1, valor2, valor3, valor4}
    {valor1, valor2, valor3, valor4}
]
```

```
Registros = [
    {1, "Barbacoa", 10.50, 20.00, 6}
    {2, "Salsa", 13.00, 16.00, 7}
    {3, "Mayonesa", 15.00, 18.00, 8}
    {4, "Mostaza", 14.00, 16.00, 4}
]
```

1.3.3.2 Comentarios

Comentarios de una línea: Se representan con un numeral y finalizan con un salto de línea.

```
# Comentarios
```

Comentarios multilínea: Inicia con tres comillas simples y finaliza con tres comillas simples.

```
'''
comentario multilinea
'''
```

1.3.3.3 Instrucciones de reportera

imprimir(cadena): Imprime por consola el valor dado por la cadena.

```
imprimir("Reporte de ");
imprimir("Abarrotería");
>>> Reporte de Abarrotería
```

imprimirln(cadena):

```
imprimirln("Reporte de ");
imprimirln("Abarrotería");
>>> Reporte de Abarrotería
>>> Abarrotería
```

conteo(): Imprime por consola la cantidad de registros en el arreglo de registros.

```
conteo();
>>> 46
```

promedio(“campo”): Imprime por consola el promedio del campo dado.

```
promedio("stock");
>>> 6.25
```

contarsi(“campo”, valor): Imprime por consola la cantidad de registros en la que el campo dado sea igual al valor dado.

```
contarsi("stock", 0);
>>> 0

contarsi("stock", 1);
>>> 18

contarsi("stock", 2);
>>> 7
```

datos(): Imprime por consola los registros leídos.

```
datos();
>>> codigo  producto  precio_compra precio_venta stock
>>> 1       Barbacoa  10.50         20.00        6
>>> 2       Salsa     13.00         16.00        7
>>> 3       Mayonesa  15.00         18.00        8
>>> 4       Mostaza   14.00         16.00        4
```

sumar(“campo”): Imprime en consola la suma todos los valores del campo dado.

```
sumar("stock");
>>> 25
```

max(“campo”): Encuentra el valor máximo del campo dado.

```
max("precio_venta");
>>> 20.00
```

min(“campo”): Encuentra el valor mínimo del campo dado.

```
min("precio_compra");
>>> 10.50
```

exportarReporte(“titulo”): Genera un archivo html con una tabla en donde se encuentren los registros leídos y con el título como parámetro.

```
exportarReporte("Reporte HTML de abarrotería");
```

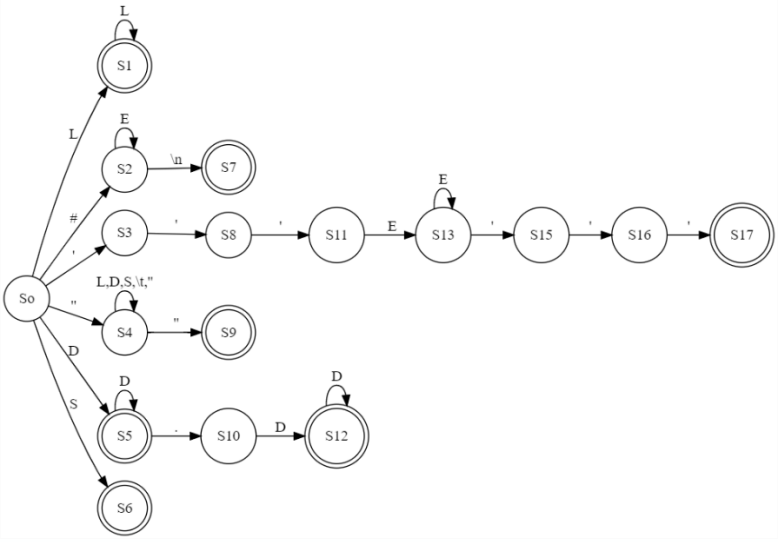
1.4 Componentes

1.4.1 Análisis Léxico

Autómata finito determinista, que realiza el análisis léxico del archivo de entrada, el autómata reconoce los caracteres de la siguiente expresión regular.

(L+$|#E*\n$|'"(E|\n)'"$|"(L|D|S|\t)*"$|D+$|D+.D+$|S$)*

Esta expresión regular es evaluada por el siguiente autómata finito determinista.



Árbol de sintaxis.

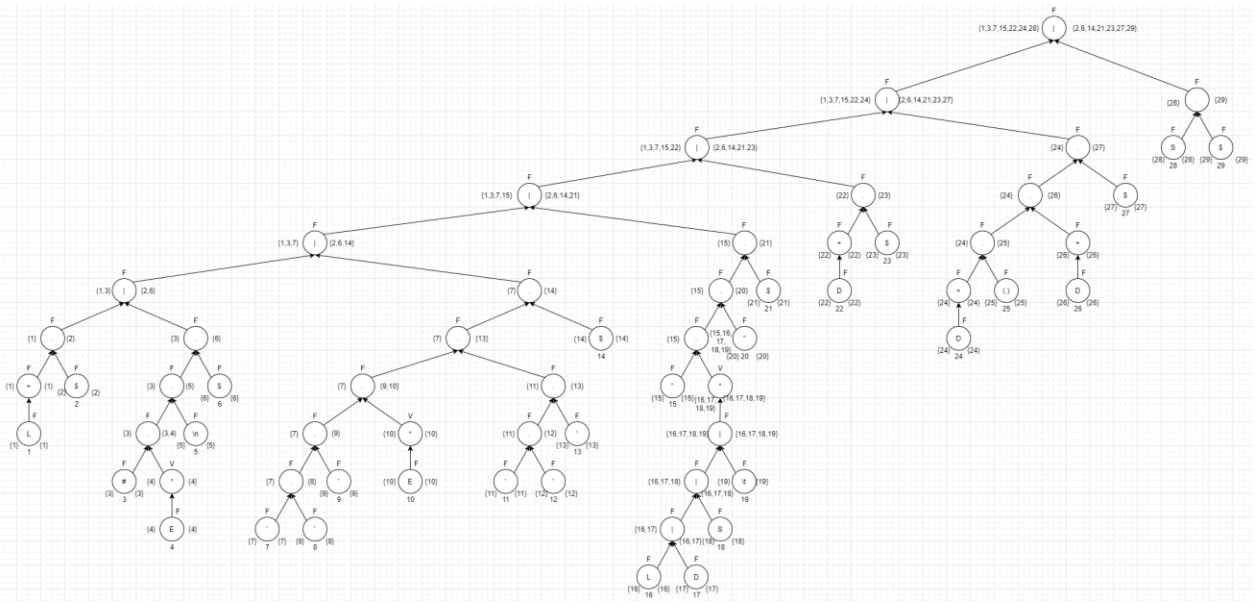


Tabla de siguientes

sym	i	Sig(i)
L	1	1,2
$	2	
#	3	4,5
E	4	4,5
\n	5	6
$	6	
'	7	8
'	8	9
'	9	10,11
E	10	10,11
'	11	12
'	12	13
'	13	14
$	14	
"	15	16,17,18,19,20
L	16	16,17,18,19,20
D	17	16,17,18,19,20
S	18	16,17,18,19,20
\t	19	16,17,18,19,20
"	20	21

$	21	
D	22	22,23
$	23	
D	24	24,25
.	25	26
D	26	26,27
$	27	
S	28	29
$	29	

1.4.2 Análisis Sintáctico

```
<inicio> ::=  <claves> <repetir>
    |       <registros> <repetir>
    |       <imprimir> <repetir>
    |       <imprimirln> <repetir>
    |       <conteo> <repetir>
    |       <promedio> <repetir>
    |       <contarsi> <repetir>
    |       <datos> <repetir>
    |       <sumar> <repetir>
    |       <max> <repetir>
    |       <min> <repetir>
    |       <exportar_reporte> <repetir>
    |       tk_comentario_one_line <repetir>
            tk_comentario_multi_line <repetir>

<repetir> ::= <claves> <repetir>
    |       <registros> <repetir>
    |       <imprimir> <repetir>
    |       <imprimirln> <repetir>
    |       <conteo> <repetir>
    |       <promedio> <repetir>
    |       <contarsi> <repetir>
    |       <datos> <repetir>
    |       <sumar> <repetir>
    |       <max> <repetir>
    |       <min> <repetir>
    |       <exportar_reporte> <repetir>
    |       tk_comentario_one_line <repetir>
    |       tk_comentario_multi_line <repetir>
    |       epsilon

<clave> ::= tk_clave tk_igual tk_abrir_corchete <bloque_claves> tk_cerrar_corchete
<registros> ::= tk_registros tk_igual tk_abrir_corchete <bloque_registros> tk_cerrar_corchete

<imprimir> ::= tk_imprimir tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma
```

```
<imprimirln> ::= tk_imprimirln tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma
<conteo> ::= tk_conteo tk_abrir_parentesis tk_cerrar_parentesis tk_punto_coma
<promedio> ::= tk_promedio tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma

<contarsi> ::= tk_contarsi tk_abrir_parentesis tk_string tk_coma <valor> tk_cerrar_parentesis
tk_punto_coma

<datos> ::= tk_datos tk_abrir_parentesis tk_cerrar_parentesis tk_punto_coma
<sumar> ::= tk_sumar tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma
<max> ::= tk_max tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma
<min> ::= tk_min tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma
<exportar_reporte> ::= tk_promedio tk_abrir_parentesis tk_string tk_cerrar_parentesis tk_punto_coma

<valor> ::= tk_string
   |      tk_double
   |      tk_integer

<bloque_claves> ::= tk_string  tk_coma <bloque_claves>
   |             tk_string

<bloque_registros> ::=  <cuerpo_registros> <bloque_registros>
   |                <cuerpo_registros>

<cuerpo_registros> ::= tk_abrir_llave <valores_registros> tk_cerrar_llave

<valores_registros> ::= <valor> tk_coma <valores_registros>
   |                <valor>
```