MANUAL DE USUARIO

Little Code

Elías Abraham Vasquez Soto

201900131

Proyecto 2- Lenguajes Formales y de Programación B-

ÍNDICE

Objetivos

Objetivo General

Brindar al usuario que posee el software *Little Code*, una guía en el uso adecuado de la misma, con el propósito que el uso que se le brinde sea totalmente eficiente, para que la experiencia sea totalmente satisfactoria y cumpla su cometido.

Objetivos Específicos

- Brindar al usuario todos los datos necesarios para comprender el funcionamiento lógico de la aplicación y la manera en que puede interactuar con ella, por medio de ingreso de datos y funciones, así como la generación de reportes de tokens y errores reconocidos en el análisis de los comandos ingresados, así como una representación del análisis sintáctico realizado mediante una forma gráfica y sencilla de entender.
- Entregar al usuario las indicaciones y pasos necesarios a seguir para ingresar datos a la aplicación e ingresar funciones para que el programa realice distintas acciones, así como para que la generación de reportes sea la correcta y evitar que se generen problemas en los resultados por un uso incorrecto de la aplicación.

Introducción

Este manual tiene como propósito el dar a conocer a todos los usuarios que hagan uso del software, las funcionalidades y pasos a seguir para darle el uso más eficaz y lograr el objetivo que este tiene, el cual es procesar el código que el usuario desee ingresar a la aplicación, ya sea para guardar datos como para ejecutar distintas acciones, así como generar los reportes de tokens y errores reconocidos en el analizador léxico y sintáctico implementado en la aplicación, y el árbol de derivación cuando se desee. Para cumplir con el objetivo propuesto se incluye la descripción de las acciones que el usuario puede realizar, los comandos que puede ingresar, y la generación de reportes, todo esto a través de gráficos y capturas de la aplicación para su mayor compresión.

Descripción de la aplicación

Es una aplicación diseñada para el procesamiento de distintas instrucciones ingresadas por el usuario. Esta procesa el código que el usuario ingresa por medio de un analizador léxico implementado, con el propósito de identificar los distintos tokens posibles, y realizar el respectivo análisis sintáctico, para interpretar de manera correcta las acciones que el usuario desea realizar. El usuario tiene libertad total de ingresar los códigos que desee, en el orden que desee, por lo que, debido a todos los posibles errores que pueden darse, existen distintas maneras en que el usuario puede verificar que el código ingresado se encuentre correcto. En caso el usuario desee indagar en el proceso que realizó el analizador léxico para procesar el código, tiene la opción de generar un reporte de Tokens, en donde podrá visualizar todos los tokens reconocidos por la aplicación. En caso el usuario desee indagar en el proceso que realizó el analizador sintáctico para procesar el código, tiene la opción de generar el árbol de derivación creado, donde podrá visualizar este proceso. Si el programa le notifica de errores léxicos/sintácticos, tiene la opción de generar el reporte de Errores, en donde podrá verlos con mayor detalle.

Descripción de las acciones posibles y manejo de la interfaz gráfica

Comandos que el programa reconoce

Antes de empezar a utilizar *Little Code*, se recomienda tener conocimiento de las acciones que se pueden realizar en él, y la estructura correcta de estas, las cuales son:

● **CLAVES**: En esta sección se declaran los claves o campos por los que están construidos los registros (sólo acepta cadenas). Estructura correcta del comando:

```
Claves=["Codigo", "Producto", "Precio_Compra","Precio_venta", "Stock"]
```

● **REGISTROS**: En esta sección se detallan los registros que se quieren analizar y guardar, **según los campos o claves que hayan sido ingresadas previamente**. Estructura correcta:

```
Registros = [
        {1, "Barbacoa", 10.50, 20, 6}{2, "Salsa", 13.00, 16.00, 7}
        {3, "Mayonesa", 15.00, 18.00, 8}{4, "Mostaza", 14, 16, 4}
        {5, "Mayonesa", 15.00, 18.00, 8}{6, "Mayonesa", 14, 16, 4}
]
```

***Importante: Los valores que puede ingresar son enteros, decimales, o cadenas. Se recomienda que la cantidad de datos por registro sea igual a la cantidad de claves ingresadas. El programa acepta que ingrese menos, pero no que ingrese más datos que claves ingresadas.**

● **COMENTARIOS**: El programa admite comentarios, para que el usuario tenga opción de realizar los que desee en el código. Estructura correcta:

```
# Comentario de linea
```

```
'''
Comentario
Multilinea
'''
```

● **INSTRUCCIÓN IMPRIMIR**: Imprime en consola el valor de la cadena que se ingrese. Si ingresa imprimirln, la cadena se mostrará en consola, con un salto de línea al final. Estructura correcta imprimir:

```
imprimir("DATOS");
```

Estructura correcta imprimirln:

```
imprimirln("CONTEO");
```

● **INSTRUCCIÓN CONTEO**: Imprime en consola la cantidad de registros ingresada. Estructura correcta:

```
conteo();
```

Resultado en consola:

```
Registros: 6, Datos: 30
```

● **INSTRUCCIÓN PROMEDIO**: Imprime en consola el promedio del campo dado. Estructura correcta:

```
promedio("Producto");
```

***Importante: La columna que pertenece al campo del que se desee sacar promedio, debe poseer todos los datos de tipo entero o decimal.**

● **INSTRUCCIÓN CONTAR SI**: Imprime en consola la cantidad de registros en la que el campo dado sea igual al valor dado. Estructura correcta:

```
contarsi("Stock", 8);
```

***Importante: El valor que se desee contar puede ser entero, decimal o cadena.**

● **INSTRUCCIÓN DATOS**: Imprime en consola los registros leídos. Estructura correcta:

```
datos();
```

Resultado en consola:

```
[ Codigo ] [ Producto ] [ Precio_Compra ] [ Precio_venta ] [ Stock ]
[   1    ] [ Barbacoa ] [      10.5     ] [      20      ] [   6   ]
[   2    ] [   Salsa  ] [      13.0     ] [      16.0    ] [   7   ]
[   3    ] [ Mayonesa ] [      15.0     ] [      18.0    ] [   8   ]
[   4    ] [ Mostaza  ] [      14       ] [      16      ] [   4   ]
[   5    ] [ Mayonesa ] [      15.0     ] [      18.0    ] [   8   ]
[   6    ] [ Mayonesa ] [      14       ] [      16      ] [   4   ]
```

● **INSTRUCCIÓN SUMAR**: Suma todos los valores del campo dado. Estructura correcta:

```
sumar("Stock");
```

● **INSTRUCCIÓN MAX**: Imprime en consola el valor máximo del campo dado. Estructura correcta:

```
max("Precio_Compra");
```

***Importante: La columna que pertenece al campo del que se desee sacar el valor máximo, debe poseer todos los datos de tipo entero o decimal.**

● **INSTRUCCIÓN MIN**: Imprime en consola el valor mínimo del campo dado. Estructura correcta:

```
min("Precio_Compra");
```

***Importante: La columna que pertenece al campo del que se desee sacar el valor mínimo, debe poseer todos los datos de tipo entero o decimal.**

● **INSTRUCCIÓN EXPORTAR REPORTE**: Genera un archivo HTML con una tabla en donde se encuentren los registros leídos y con el título como parámetro. Estructura correcta:

```
exportarReporte("Reporte de abarroteria");
```

Ventana principal *Little Code*

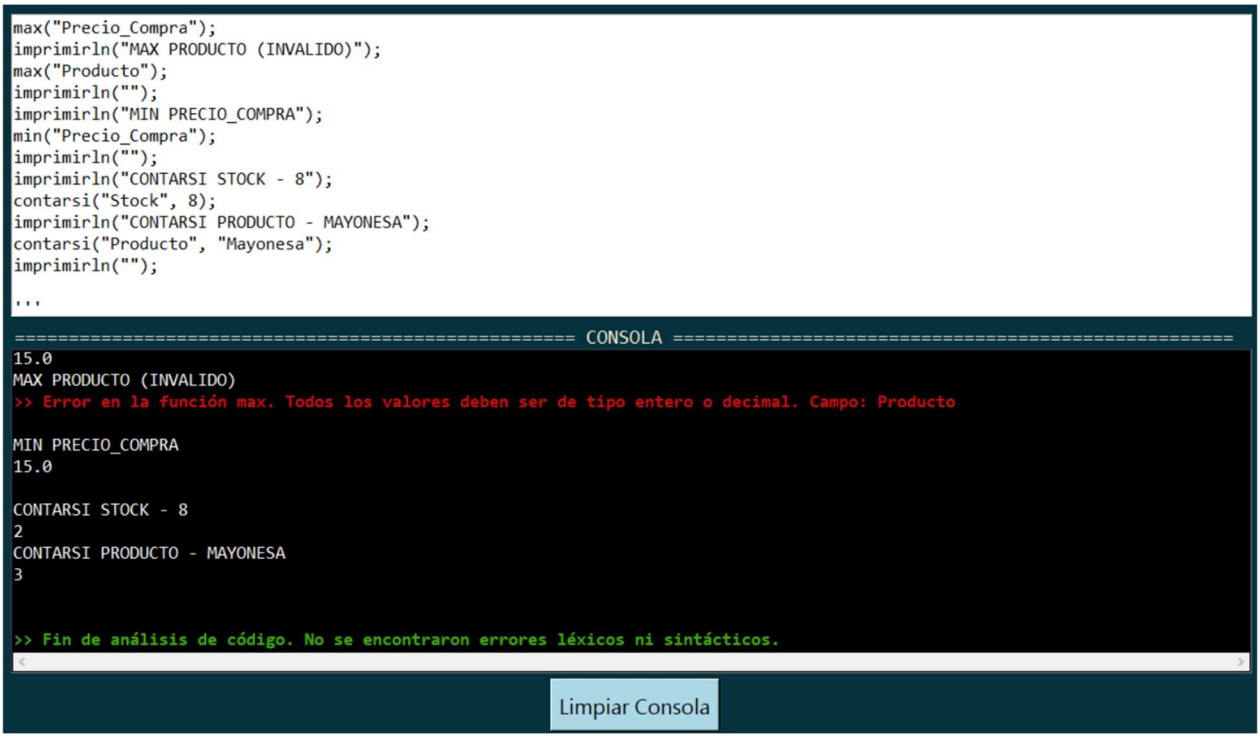
Ventana con la que se encontrará el usuario al ejecutar la aplicación. En ella, el usuario se encontrará con unos botones, con los cuales podrá interactuar, y un área de texto en donde podrá escribir el código que desee, y que se describirán a continuación.

1) Editor de texto

Recuadro blanco de la aplicación, donde el usuario tiene la opción de ingresar el código que desee sea procesado. También, si carga un archivo LFP al programa, todo el texto del archivo se verá reflejado en este apartado.

2) Consola

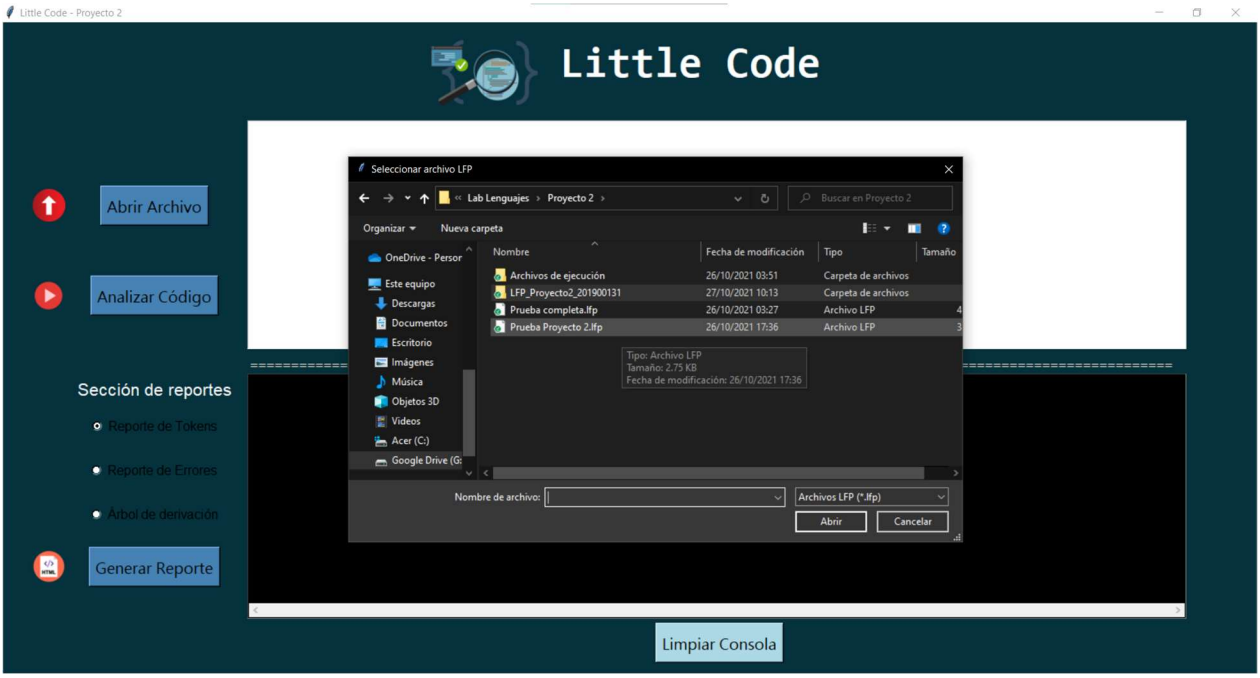
Apartado en donde el usuario podrá visualizar el resultado de las instrucciones que ingrese al programa. Además, se reportarán los errores de ejecución que se vayan encontrando, y al final del análisis, indicará si se encontraron o no errores léxicos o sintácticos.



También cuenta con la opción de limpiar la consola, por si desea eliminar los resultados obtenidos de ejecuciones anteriores.

3) Botón *Abrir Archivo*

Se abre un cuadro de diálogo, en donde el usuario podrá cargar archivos en formato LFP a la aplicación. Esta opción existe, por si el usuario desea escribir el código por separado, y solamente desea cargarlo al programa.



Al cargar un archivo, automáticamente, se cargará todo el texto que en él se encuentre al Editor de texto de la aplicación. Aquí, el usuario puede aún puede realizar las modificaciones que desee.

4) Botón *Analizar Código*

Empieza la ejecución del analizador léxico y sintáctico, de todo el código que el usuario haya ingresado al editor de texto.

5) Sección de Reportes

En este apartado, el usuario podrá generar el **reporte de Tokens** que el analizador léxico reconoció al procesar el código, el **árbol de derivación** que se originó durante el análisis sintáctico, como el **reporte de Errores**, en donde encontrará los errores que el analizador léxico y el analizador sintáctico hayan encontrado al procesar el código, si es que estos ocurrieron.

***Importante: En el reporte de Errores, no se visualizan los errores propios de la ejecución (ver apartado Errores de ejecución)**

Estos se generan en formato HTML, a excepción del Árbol de derivación, que se genera tanto en formato .DOT, que contiene el código del árbol, como en formato .PNG, para que el usuario pueda tener una representación gráfica del mismo.

Esta opción le permite al usuario indagar un poco más en el funcionamiento de la aplicación (con el reporte de Tokens y del árbol de derivación), así como observar los posibles errores que el código posea, para que puedan ser arreglados y puedan ejecutarse los comandos sin problemas.

- **Reporte de Tokens**

#	Id Token	Token	Lexema	Fila	Columna
1	4	Palabra reservada	Claves	1	1
2	17	Simbolo	=	1	7
3	18	Simbolo	[	1	8
4	1	Cadena	Codigo	2	5
5	20	Simbolo	,	2	13
6	1	Cadena	Producto	2	15
7	20	Simbolo	,	2	25
8	1	Cadena	Precio_Compra	2	27
9	20	Simbolo	,	2	42
10	1	Cadena	Precio_venta	3	5

- **Reporte de Errores**

Reporte de Errores

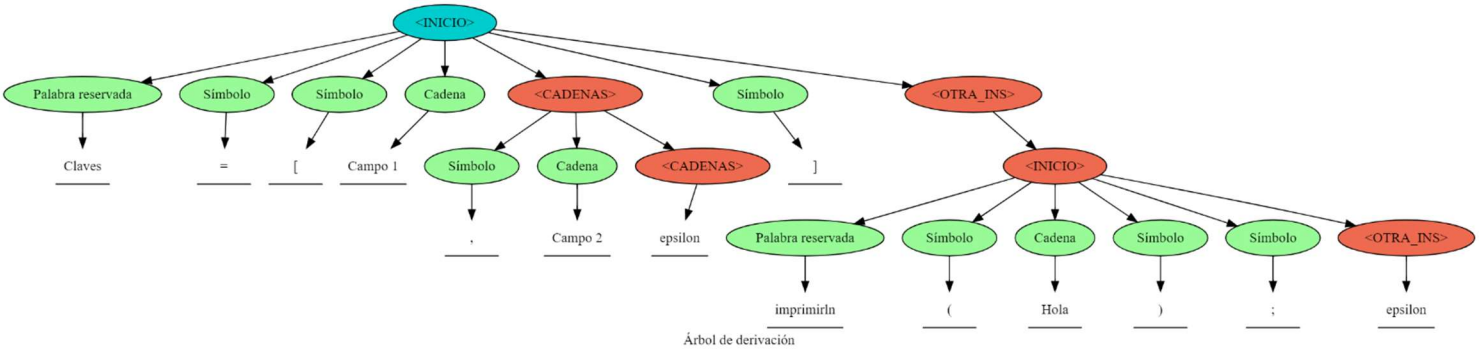
Errores Léxicos

#	Caracter	Descripcion	Fila	Columna
1	&	El caracter no figura como parte del lenguaje del programa.	3	1
2	}	Se esperaba un número en la parte decimal.	9	24

Errores Sintácticos

#	Caracter/Lexema	Descripcion	Recuperado	Fila	Columna
1	ERRORES	Se esperaba una palabra reservada. Vino token ID.	No	1	1
2	}	En Registros, se encontró un token Simbolo, que ocasionó una sintaxis incorrecta.	No	6	24
3	}	En Registros, se encontró un token Simbolo, que ocasionó una	No	9	24

- **Árbol de derivación**



Árbol de derivación

***Importante: Múltiples códigos pueden generar árboles demasiado extensos, por lo que, si se desea tener una mejor apreciación de este, se recomienda generarlo cuando se hayan ingresado pocas instrucciones al programa.**

Errores de ejecución

Estos pueden surgir a lo largo de la ejecución del código que el usuario ingrese, y no se reportan en el reporte de Errores, sino que el programa notificará al usuario de estos mediante mensajes a través de la consola.

Para evitar estos, se le hace al usuario las siguientes observaciones:

- Las claves solo pueden ser cadenas de texto, y no pueden venir vacías.
- Los Registros dependen de las claves ingresadas, por lo que, si no se han ingresado estas, los registros tampoco se ingresaran.
- No se acepta que ingrese más datos por registro que claves ingresadas.
- Los comandos o instrucciones que aceptan un solo argumento tienen como restricción que este debe ser una cadena de texto.
- Existen instrucciones que requieren que todos los datos correspondientes a cierta clave, sean de tipo número (entero o decimal), por lo que se le recomienda que la sección correspondiente a la descripción de cada comando sea estudiada con anterioridad.

Requerimientos del sistema

Hardware

La recomendación es:

- Procesador de 1,6 GHz o con mayor velocidad
- 2 GB de RAM
- Espacio en disco duro: 100MB libre (mínimo)

Software

Las herramientas que se utilizaron para el desarrollo de la aplicación, y que se recomienda se encuentren instaladas en la máquina del usuario, son las siguientes:

- Sistema operativo:
    - Windows 7 (con .NET Framework 4.5.2), 8.0, 8.1 y 10 (32 y 64 bits)
    - Linux (Debian): Ubuntu Desktop 16.04, Debian 9
    - Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 8, Fedora 24
- Python 3.9.7 (cualquier versión de Python 3 puede correr la aplicación)
- Cualquier IDE capaz de ejecutar el código fuente (El IDE con el que se desarrolló el proyecto fue Visual Studio Code versión 1.60.1)