Entrega 1:

Análisis léxico y sintáctico

PROCESADORES DE LENGuAJEs

Carlos Ruiz Ballesteros

Héctor ruiz-poveda coca

GIS-GII

Índice

[AUTORES 2](#_Toc447558436)

[DESCRIPCIÓN DEL CÓDIGO 2](#_Toc447558437)

[Analizador léxico 2](#_Toc447558438)

[Analizador sintáctico 2](#_Toc447558439)

[Interfaz gráfica y su uso 3](#_Toc447558440)

[EJEMPLOS DE EJECUCIÓN 3](#_Toc447558441)

[Página principal 3](#_Toc447558442)

[Ejecución de un análisis 4](#_Toc447558443)

[Ejecución de un análisis con error 4](#_Toc447558444)

# AUTORES

Carlos Ruiz Ballesteros.

Héctor Ruiz-Poveda Coca.

Ambos autores somos del Doble Grado en Ingeniería del Software + Ingeniería Informática.

# DESCRIPCIÓN DEL CÓDIGO

El código se divide en dos partes principales:

## Analizador léxico

Por medio de la herramienta *“Flex”* se ha desarrollado un analizador léxico que permite identificar los distintos elementos: valores numéricos, identificadores, constantes literales, comentarios, retornos de carro…

El analizador léxico transforma estos elementos, así como palabras reservadas, en tokens y símbolos que utilizará el analizador sintáctico.

## Analizador sintáctico

La gramática que se encarga de comprobar si el código es sintácticamente correcto ha sido generada por *“CUP”*.

Este analizador se encarga de comprobar que la estructura y el orden del programa son correctas.

Contiene variables y métodos para el control y detección de errores en la sintáxis del fichero a analizar. Un ejemplo son las variables booleanas ***if\_expression*** y  ***case\_expression***, utilizadas para informar de un token incorrecto en la expresión aritmética de un *IF* o un *CASE*. Por otro lado cuenta con un método que informa la línea y la columna en la que se encuentra el error detectado ( ***syntax\_error(Symbol s)***).

Se encuentran también en el analizador los símbolos terminales, no terminales y las reglas gramaticales necesarias para el correcto análisis gramatical.

Con motivo de una correcta recuperación de errores, se ha incluido la regla ***DEFRANG***, la cual, en caso de un error, busca un corchete cerrado. En caso contrario, se ejecutaría la regla ***ALLTYPES***.

Cabe destacar que se han añadido las modificaciones pertinentes para añadir la gramática que permite el reconocimiento de matrices y registros, así como sentencias*”IF”*, *“WHILE”* y *“FOR”*.

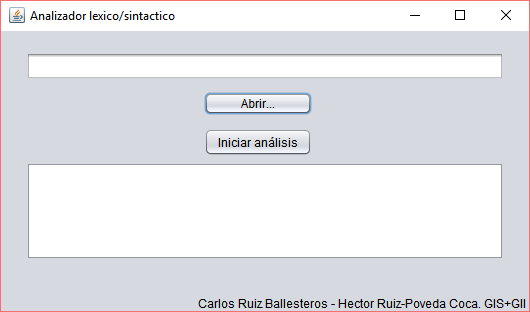
## Interfaz gráfica y su uso

Con motivo de una mejor usabilidad se ha desarrollado una interfaz gráfica, la cual permite seleccionar un fichero, analizarlo, y proporciona un correcto feedback al usuario sobre el resultado de dicho análisis.

# EJEMPLOS DE EJECUCIÓN

A continuación, se muestran las capturas de pantallas correspondientes a la ejecución del código:

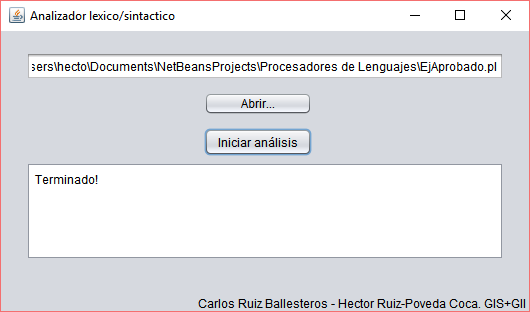
## Página principal



Esta página permite la selección de un fichero, mediante un JFileChooser para mayor comodidad, el cual será analizado.

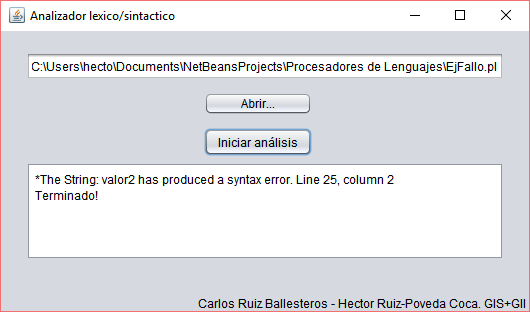
Una vez seleccionado el archivo, y tras pulsar el botón “Iniciar análisis”, la retroalimentación de éste aparecerá en el cuadro de texto inferior.

## Ejecución de un análisis



Aquí se muestra la ejecución del análisis sobre el fichero “EjAprobado.pl”.

## Ejecución de un análisis con error



Se puede apreciar cómo a pesar de detectar el error, el analizador termina su tarea analizando el archivo al completo.