Primer proyecto – Compiladores e intérpretes

Analizador léxico

Profesor:

Allan Rodríguez Dávila

Estudiantes:

Jairo González Hidalgo - 2024178535
Rafael Odio Mendoza - 2024083499

26 de diciembre – Verano 2025

Manual de usuario

Compilación

Para que el programa funcione, es necesario tener instalado el JDK de Java (se recomienda la versión 11 o superior). Si se usa VS Code, hay que instalar el entorno de Java, se recomienda el "Extension Pack for Java" de Microsoft. También pueden usar otros IDEs como IntelliJ IDEA o Eclipse sin problema.

El proyecto usa JFlex y CUP para generar el analizador léxico. Estos se encuentran como archivos .jar en la carpeta lib/ y no necesitan instalación adicional.

El proyecto fue diseñado para que no hubiera que estar usando comandos para compilar el Lexer y el Parser cada vez que se cambiara algo. En el menú principal, la Opción 1 se encarga de borrar los archivos .java viejos en la carpeta parserlexer, ejecutar el jar de cup para generar la clase de símbolos (sym.java) y ejecutar el jar de JFlex para generar el motor del scanner (Lexer.java).

Para crearlos manualmente sin usar main, primero hay que estar dentro de la carpeta programa: `\proyecto-compiladores-e-interpretes> cd programa`

Posteriormente se ejecutan

Lexer.java: `java -jar lib/jflex-full-1.9.1.jar parserlexer/Lexer.jflex`

sym.java (que también compila Parser.java): `java -jar lib/java-cup-11b.jar -destdir parserlexer/ -parser Parser parserlexer/Parser.cup`

Los archivos generados quedan en la carpeta parserlexer/

Ejecución

Ejecutar el archivo Main.java, desplegará un menú.

Uso

La opción 0 cierra el programa.

Para crear los archivos Lexer.java y sym.java se usa la opción 1.

Para ejecutar el análisis se usa la opción 2, se escogen los archivos, esta muestra una lista de todos los archivos .txt en la carpeta archivos_prueba/, puede seleccionar un archivo específico o analizar todos mediante la opción 0, los archivos son listados individualmente con un número que permite seleccionarlos.

Por cada archivo analizado se genera un reporte en archivos_salida/ con el formato reporte_nombre.txt

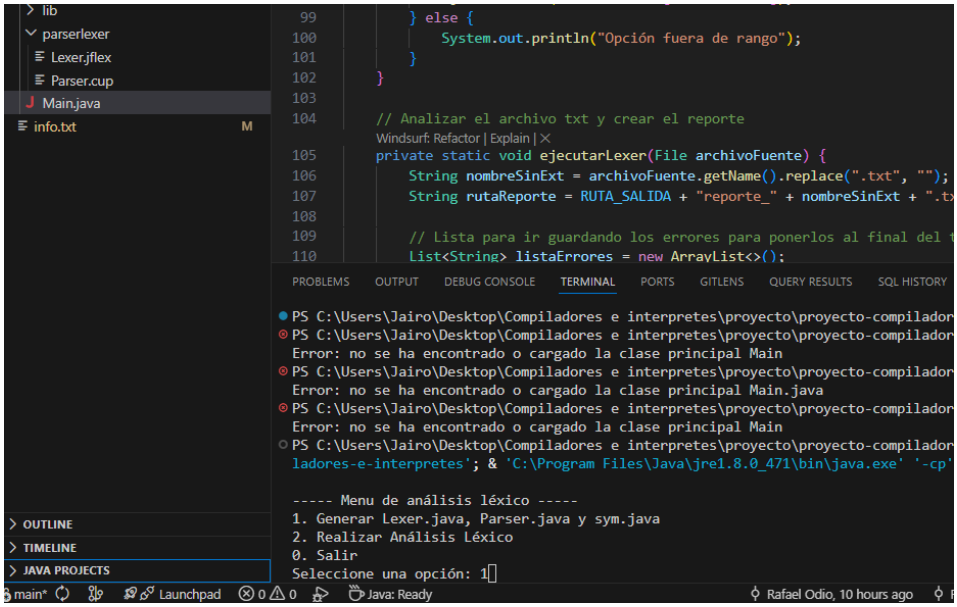
El reporte incluye una tabla con cada token encontrado (identificador, lexema, línea, columna) y una lista de errores léxicos si los hubiera.

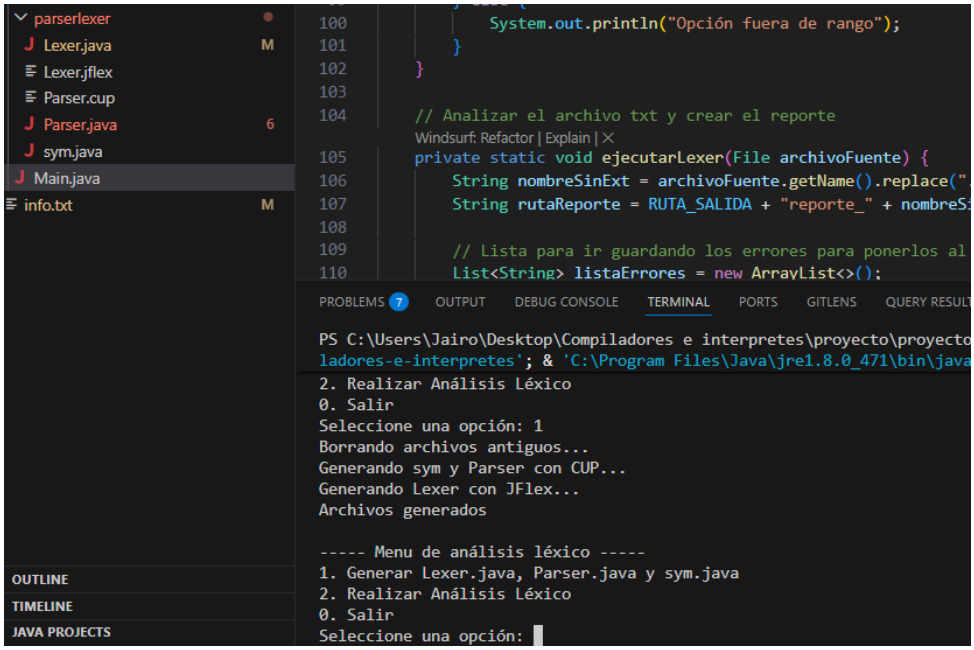
Se pueden ver más pruebas del uso en el siguiente apartado.

Pruebas de funcionalidad

Se pueden apreciar completamente las pruebas en el video: https://www.youtube.com/watch?v=kATyjgQVy2M

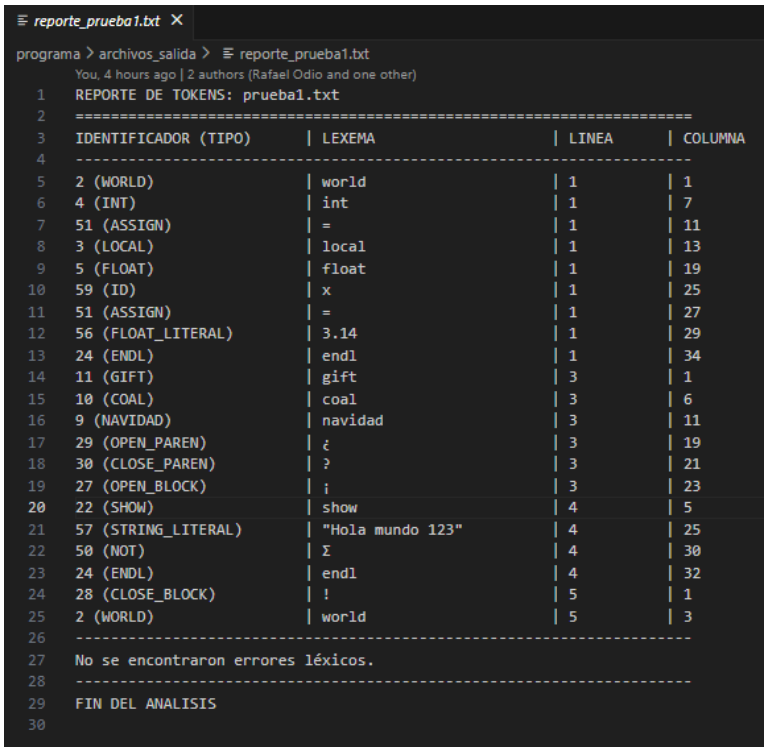
1. Generación de archivos .java con la opción 1





Los errores que marca Parser.java es normal, aún no se desarrolla esa parte porque no es del análisis léxico, no afecta.

2. Análisis de un archivo sin errores

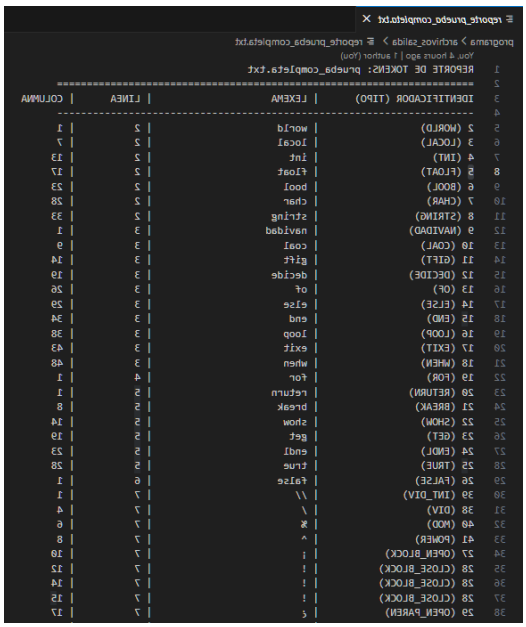


3. Análisis de un archivo con errores léxicos

reporte_prueba2.txt

```
programa > archivos_salida > reporte_prueba2.txt
You, 4 hours ago | 2 authors (You and one other)
REPORTE DE TOKENS: prueba2.txt        You, 8 hours ago • feat: manejo de errores le
================================================================
IDENTIFICADOR (TIPO)      | LEXEMA                    | LINEA      | COLUMNA
----------------------------------------------------------------------------
2 (WORLD)                 | world                     | 1          | 1
4 (INT)                   | int                       | 1          | 7
51 (ASSIGN)               | =                         | 1          | 11
3 (LOCAL)                 | local                     | 1          | 13
5 (FLOAT)                 | float                     | 1          | 19
59 (ID)                   | x                         | 1          | 25
51 (ASSIGN)               | =                         | 1          | 27
56 (FLOAT_LITERAL)        | 3.14                      | 1          | 29
24 (ENDL)                 | endl                      | 1          | 34
11 (GIFT)                 | gift                      | 3          | 1
10 (COAL)                 | coal                      | 3          | 6
60 (ERROR)                | #                         | 3          | 11
22 (SHOW)                 | show                      | 4          | 5
57 (STRING_LITERAL)       | "Hola mundo 123"          | 4          | 25
50 (NOT)                  | Σ                         | 4          | 30
24 (ENDL)                 | endl                      | 4          | 32
60 (ERROR)                | #                         | 5          | 1
28 (CLOSE_BLOCK)          | !                         | 6          | 1
2 (WORLD)                 | world                     | 6          | 3
----------------------------------------------------------------------------
ERRORES ENCONTRADOS:
- Caracter ilegal <#> en línea 3
- Caracter ilegal <#> en línea 5
----------------------------------------------------------------------------
FIN DEL ANALISIS
```

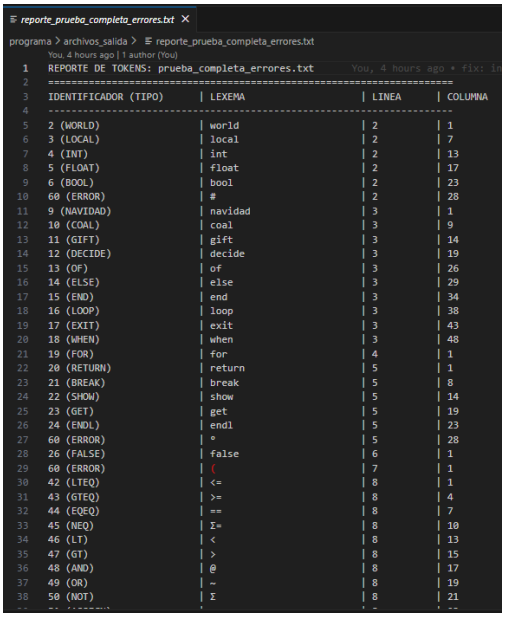
4. Análisis usando todos los tokens del lenguaje



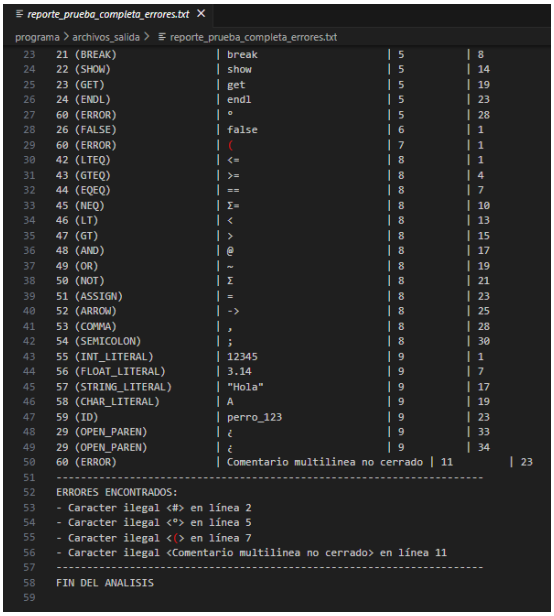
reporte_prueba_completa.txt

```
programa > archivos_salida > reporte_prueba_completa.txt
44 (EQEQ)                 | ==                        | 8          | 7
45 (NEQ)                  | Σ=                        | 8          | 10
46 (LT)                   | <                         | 8          | 13
47 (GT)                   | >                         | 8          | 15
48 (AND)                  | @                         | 8          | 17
49 (OR)                   | ~                         | 8          | 19
50 (NOT)                  | Σ                         | 8          | 21
51 (ASSIGN)               | =                         | 8          | 23
52 (ARROW)                | ->                        | 8          | 25
53 (COMMA)                | ,                         | 8          | 28
54 (SEMICOLON)            | ;                         | 8          | 30
55 (INT_LITERAL)          | 12345                     | 12         | 1
56 (FLOAT_LITERAL)        | 3.14                      | 12         | 7
57 (STRING_LITERAL)       | "Hola"                    | 12         | 17
58 (CHAR_LITERAL)         | A                         | 12         | 19
59 (ID)                   | una_variable_123          | 12         | 23
9 (NAVIDAD)               | navidad                   | 13         | 1
27 (OPEN_BLOCK)           | ¡                         | 13         | 9
12 (DECIDE)               | decide                    | 13         | 11
13 (OF)                   | of                        | 13         | 18
29 (OPEN_PAREN)           | ¿                         | 13         | 21
6 (BOOL)                  | bool                      | 13         | 23
59 (ID)                   | x                         | 13         | 28
51 (ASSIGN)               | =                         | 13         | 30
25 (TRUE)                 | true                      | 13         | 32
30 (CLOSE_PAREN)          | ?                         | 13         | 37
52 (ARROW)                | ->                        | 13         | 39
22 (SHOW)                 | show                      | 13         | 42
57 (STRING_LITERAL)       | "ok"                      | 13         | 50
24 (ENDL)                 | endl                      | 13         | 52
28 (CLOSE_BLOCK)          | !                         | 13         | 57
----------------------------------------------------------------------------
No se encontraron errores léxicos.
----------------------------------------------------------------------------
FIN DEL ANALISIS
```

5. Análisis con todos los tokens pero incluyendo errores



```
reporte_prueba_completa_errores.txt
programa > archivos_salida > reporte_prueba_completa_errores.txt
You, 4 hours ago | 1 author (You)
REPORTE DE TOKENS: prueba_completa_errores.txt        You, 4 hours ago • fix: 10
================================================================================
IDENTIFICADOR (TIPO)      | LEXEMA                    | LINEA      | COLUMNA
--------------------------------------------------------------------------------
2 (WORLD)                 | world                     | 2          | 1
3 (LOCAL)                 | local                     | 2          | 7
4 (INT)                   | int                       | 2          | 13
5 (FLOAT)                 | float                     | 2          | 17
6 (BOOL)                  | bool                      | 2          | 23
60 (ERROR)                | #                         | 2          | 28
9 (NAVIDAD)               | navidad                   | 3          | 1
10 (COAL)                 | coal                      | 3          | 9
11 (GIFT)                 | gift                      | 3          | 14
12 (DECIDE)               | decide                    | 3          | 19
13 (OF)                   | of                        | 3          | 26
14 (ELSE)                 | else                      | 3          | 29
15 (END)                  | end                       | 3          | 34
16 (LOOP)                 | loop                      | 3          | 38
17 (EXIT)                 | exit                      | 3          | 43
18 (WHEN)                 | when                      | 3          | 48
19 (FOR)                  | for                       | 4          | 1
20 (RETURN)               | return                    | 5          | 1
21 (BREAK)                | break                     | 5          | 8
22 (SHOW)                 | show                      | 5          | 14
23 (GET)                  | get                       | 5          | 19
24 (ENDL)                 | endl                      | 5          | 23
60 (ERROR)                | °                         | 5          | 28
26 (FALSE)                | false                     | 6          | 1
60 (ERROR)                | (                         | 7          | 1
42 (LTEQ)                 | <=                        | 8          | 1
43 (GTEQ)                 | >=                        | 8          | 4
44 (EQEQ)                 | ==                        | 8          | 7
45 (NEQ)                  | Σ=                        | 8          | 10
46 (LT)                   | <                         | 8          | 13
47 (GT)                   | >                         | 8          | 15
48 (AND)                  | @                         | 8          | 17
49 (OR)                   | ~                         | 8          | 19
50 (NOT)                  | Σ                         | 8          | 21
```



```
reporte_prueba_completa_errores.txt
programa > archivos_salida > reporte_prueba_completa_errores.txt
21 (BREAK)                | break                     | 5          | 8
22 (SHOW)                 | show                      | 5          | 14
23 (GET)                  | get                       | 5          | 19
24 (ENDL)                 | endl                      | 5          | 23
60 (ERROR)                | °                         | 5          | 28
26 (FALSE)                | false                     | 6          | 1
60 (ERROR)                | (                         | 7          | 1
42 (LTEQ)                 | <=                        | 8          | 1
43 (GTEQ)                 | >=                        | 8          | 4
44 (EQEQ)                 | ==                        | 8          | 7
45 (NEQ)                  | Σ=                        | 8          | 10
46 (LT)                   | <                         | 8          | 13
47 (GT)                   | >                         | 8          | 15
48 (AND)                  | @                         | 8          | 17
49 (OR)                   | ~                         | 8          | 19
50 (NOT)                  | Σ                         | 8          | 21
51 (ASSIGN)               | =                         | 8          | 23
52 (ARROW)                | ->                        | 8          | 25
53 (COMMA)                | ,                         | 8          | 28
54 (SEMICOLON)            | ;                         | 8          | 30
55 (INT_LITERAL)          | 12345                     | 9          | 1
56 (FLOAT_LITERAL)        | 3.14                      | 9          | 7
57 (STRING_LITERAL)       | "Hola"                    | 9          | 17
58 (CHAR_LITERAL)         | A                         | 9          | 19
59 (ID)                   | perro_123                 | 9          | 23
29 (OPEN_PAREN)           | ¿                         | 9          | 33
29 (OPEN_PAREN)           | ¿                         | 9          | 34
60 (ERROR)                | Comentario multilinea no cerrado | 11         | 23
--------------------------------------------------------------------------------
ERRORES ENCONTRADOS:
- Caracter ilegal <#> en línea 2
- Caracter ilegal <°> en línea 5
- Caracter ilegal <(> en línea 7
- Caracter ilegal <Comentario multilinea no cerrado> en línea 11
--------------------------------------------------------------------------------
FIN DEL ANALISIS
```

Descripción del problema

El objetivo de este primer proyecto es desarrollar la fase de análisis léxico para un lenguaje de programación imperativo diseñado para configuración de chips en sistemas empotrados.

El lenguaje funcionará con ciertos símbolos no convencionales, como (¡ y !) para delimitar bloques, (¿ y ?) como paréntesis, también Σ, @ y ~ como operadores NOT, AND y OR, (є y э) como comentarios multilínea, también deben haber operadores de división entera, potencia y diferente.

El analizador léxico debe ser capaz de leer archivos fuente escritos en este lenguaje nuevo que se está desarrollando, debe identificar y clasificar cada elemento del código en tokens, reportar errores léxicos sin que el análisis se detenga y generar reportes detallados con información de los tokens y errores.