TEC | Tecnológico de Costa Rica

COMPILADORES E INTERPRETES

PROYECTO 1

ESTUDIANTES:

ESTEFANI VALVERDE

JORDANO ESCALANTE

KEINGELL MOODIE

PROFESOR

ALLAN RODRIGUEZ

I SEMESTRE 2025

Manual de usuario: instrucciones de compilación, ejecución y uso bien detalladas.

Requisitos previos

Antes de compilar el proyecto, asegúrese de tener instalado:

- Java Development Kit (JDK) 17 o superior.
- Gradle 7.5 o superior.

- Las siguientes librerías:
- **JFlex** (generador de analizadores léxicos para Java).
- **CUP** (Constructor of Useful Parsers, generador de analizadores sintácticos).
- Java CUP Runtime (para ejecutar el parser generado por CUP)

A la hora de crear el proyecto asegúrese de que el proyecto contenga un archivo `build.gradle` y la estructura estándar:

/src

/main

/java

build.gradl

- **Compilar el proyecto**
  - Ejecute el siguiente comando:

    ```
    gradle build
    ```

- **Ejecutar el programa**
  - Una vez compilado, ejecute el proyecto con:

    ```
    gradle run
    ```

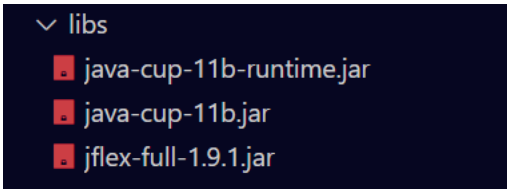
(Opcional) Limpiar archivos compilados

- Si desea limpiar los archivos generados por la compilación:

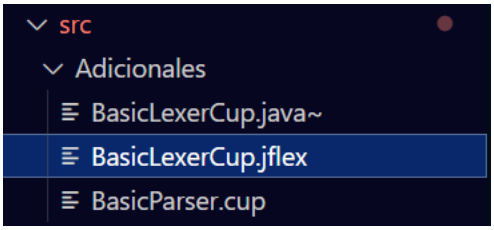
  ```
  gradle clean
  ```

Pruebas de funcionalidad: Incluir screenshots.

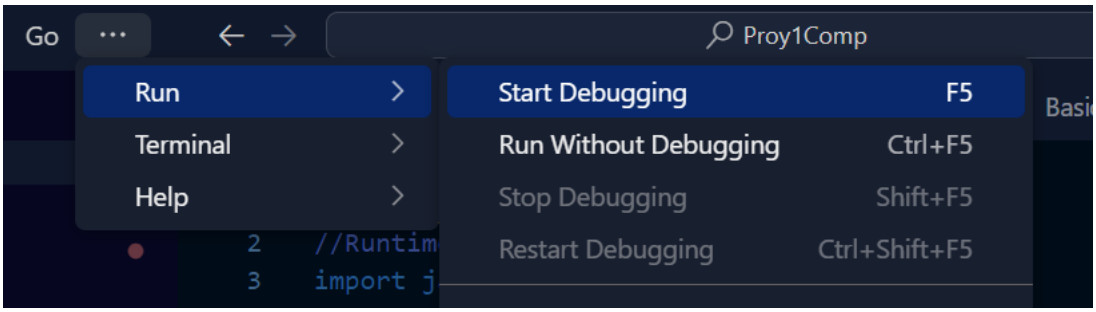
1 Como primer paso se debe tener incluidas las librerías de Jflex y Cup en la carpeta libs.



2 Como segundo paso se debe tener incluida la gramática elaborada en la asignación de la tarea 1 en los archivos .jflex y .cup para generar los archivos creados por el analizador léxico y sintáctico.

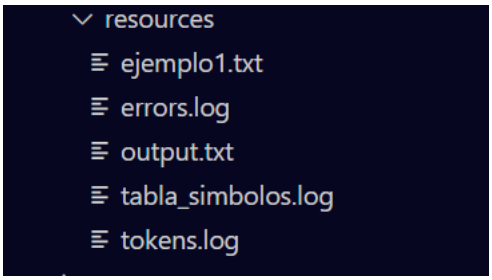


3. Ya con esto se puede pasar al momento de ejecución, ejecutando el programa en el archivo main que es el App.java

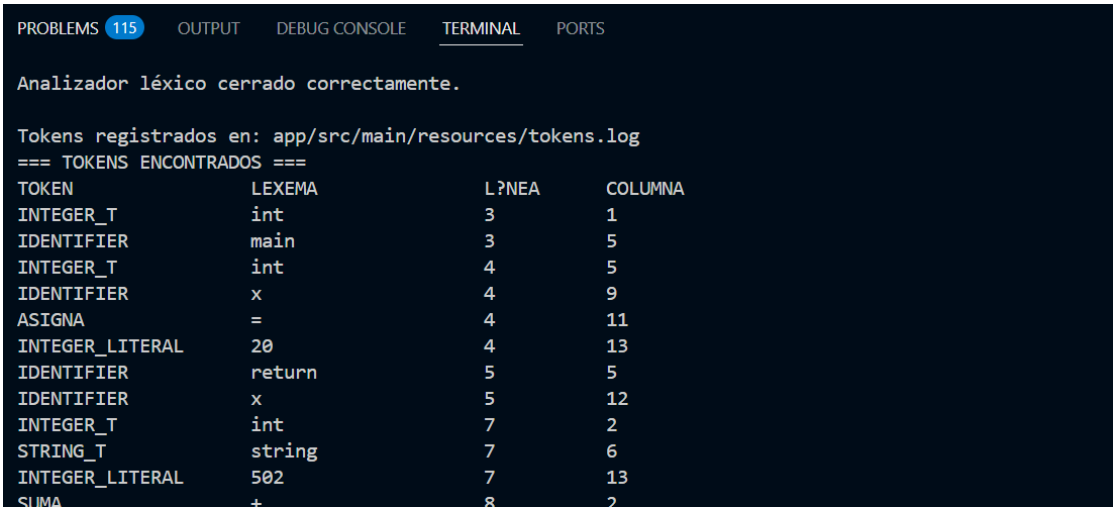


4 Inmediatamente el programa generara los archivos BasicLexerCup.java, parser.java y sym.java quienes son archivos generados por el Jflex y el Cup.

5. El programa en su tiempo de ejecución va generar los archivos de error, tabla de símbolos y los tokens.



6. El programa también mostrara por consola alguna información de los archivos generados.



Descripción del problema.

Este proyecto comprende la fase de análisis léxico y análisis sintáctico para la gramática descrita en el Anexo I (descrita también en Tarea I). Se debe desarrollar un scanner utilizando la herramienta JFlex y se debe desarrollar un parser utilizando la herramienta Cup.

Un programa escrito para este lenguaje está compuesto por una secuencia de declaraciones de procedimientos, que contienen diferentes expresiones y asignaciones de expresiones; todo programa debe contener un único método main.

Para comprobar el desarrollo de ambas fases el programa a presentar deberá tomar un archivo fuente y realizar lo siguiente:

a) El sistema debe leer un archivo fuente.

b) Se debe escribir en un archivo todos los tokens encontrados, identificador asociado con el lexema.

c) Por cada token deberán indicar en cuál tabla de símbolos va y cual información se almacenará.

d) Indicar si el archivo fuente puede o no ser generado por la gramática.

e) Reportar y manejar los errores léxicos y sintácticos encontrados. Debe utilizar la técnica de Recuperación en Modo Pánico (error en línea y continúa con la siguiente). En el reporte de errores es fundamental indicar la línea en que ocurre.

Gramática

La gramática EBNF que reconocerá el lenguaje será la descrita en el Anexo I que fue desarrollada en la tarea I. La gramática desarrollada en la tarea puede ser sujeta a cambios según se avance en la implementación de las dos primeras fases del compilador. La gramática puede tener ideas de gramáticas de referencia, pero en su estructura debe ser original.

Scanner

El desarrollo del Analizador Léxico se debe realizar utilizando la herramienta JFLex. Incluye el reconocimiento de los tokens y recuperación de errores sobre los archivos fuentes analizados. Debe reportar los errores y seguir procesando el código fuente.

Parser

El desarrollo del Analizador Sintáctico se debe realizar utilizando la herramienta Cup. Incluye la comprobación sintáctica y recuperación de errores. Debe reportar los errores y seguir procesando el código fuente.

Diseño del programa: decisiones de diseño, algoritmos usados.

Para el diseño del programa decidimos separar claramente las fases del compilador, utilizando JFlex para el análisis léxico y CUP para el análisis sintáctico. Esto nos permitió manejar de forma más ordenada los tokens y las reglas gramaticales. En el archivo .jflex definimos los patrones léxicos que generan los tokens, y en el archivo .cup definimos la gramática con sus respectivas producciones. Utilizamos la técnica de recuperación en modo pánico para el manejo de errores, lo que permite continuar el análisis aun cuando se detectan errores léxicos o sintácticos. El programa se ejecuta a partir del archivo App.java y genera archivos como la tabla de símbolos, los errores encontrados y los tokens identificados. Estas decisiones facilitaron la implementación modular y el análisis detallado del archivo fuente.

Librerías usadas: creación de archivos, etc.

Para la correcta elaboración del proyecto se utilizaron las siguientes librerías y herramientas:

- **Gradle** (automatización de construcción de proyectos Java).
- **Java Development Kit (JDK)**
- **JFlex** (generador de analizadores léxicos para Java).
- **CUP** (Constructor of Useful Parsers, generador de analizadores sintácticos).
- Java CUP Runtime (para ejecutar el parser generado por CUP)

Análisis de resultados: objetivos alcanzados, objetivos no alcanzados.

Objetivo	Alcanzado	No Alcanzado
El sistema debe leer un archivo fuente.	✓	
Se debe escribir en un archivo todos los tokens encontrados, identificador asociado con el lexema.	✓	
Por cada token deberán indicar en cuál tabla de símbolos va y cual información se almacenará.	✓	
Indicar si el archivo fuente puede o no ser generado por la gramática.		✓
Reportar y manejar los errores léxicos y sintácticos encontrados.	Maso menos (no se muestra cómo se debería de ver).	

Bitácora (autogenerada en git)

https://github.com/Estefani05/Proyecto1Compiladores