TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Escuela de ingeniería en computación

IC-5701 Compiladores e Intérpretes
PY01 - Análisis léxico

Estudiante(s):

Joselyn Priscilla Jiménez Salgado	2021022576
Dylan Montiel Zúñiga	2023205654

Profesor(a) a cargo:

Allan Rodríguez Dávila

Verano 2024-2025

Tabla de contenidos

Manual de Usuario

Este manual proporciona las instrucciones necesarias para compilar el proyecto, ejecutar y usar el analizador lexico con jflex y cup. Este proyecto fue desarrollado con la herramienta de desarrollo Apache NetBeans.

Configuración del entorno

1. **Instalación de la herramienta de desarrollo y dependencias**

Si no tienes NetBeans instalado, descárgalo desde la página oficial e instálalo en tu equipo. Asegúrate de instalar las herramientas necesarias (por ejemplo, soporte para Java el cual lo puede encontrar en Oracle JDK).

2. **Crear un directorio para el proyecto (opcional)**

```
mkdir AnalizadorLexico
```

3. **Clonar el repositorio (en el cmd)**

```
git clone https://github.com/MITTuu/PP01-Compiladores_E_Interpretes.git
```

4. **Cargar el proyecto con NetBeans**

a) Ve al menú principal y selecciona:

File > Open Project (Archivo > Abrir Proyecto).

b) Navega hasta el directorio donde se encuentra el proyecto.

c) Selecciona la carpeta “programa”.

d) Una vez hecho esto, revisa la ventana de proyectos en NetBeans para asegurarte de que todos los archivos y dependencias están disponibles.

Imagen 1: Abrir el proyecto

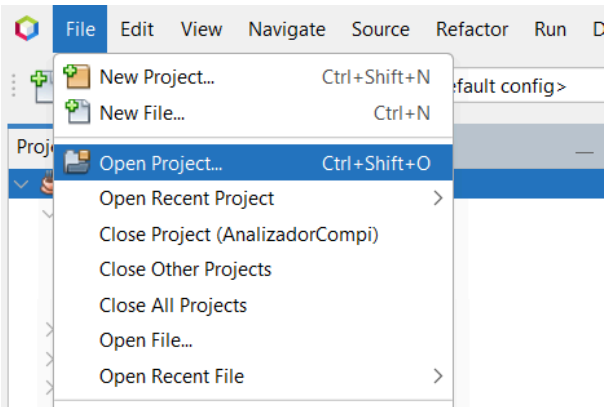


Imagen 2: Seleccionar el directorio programa

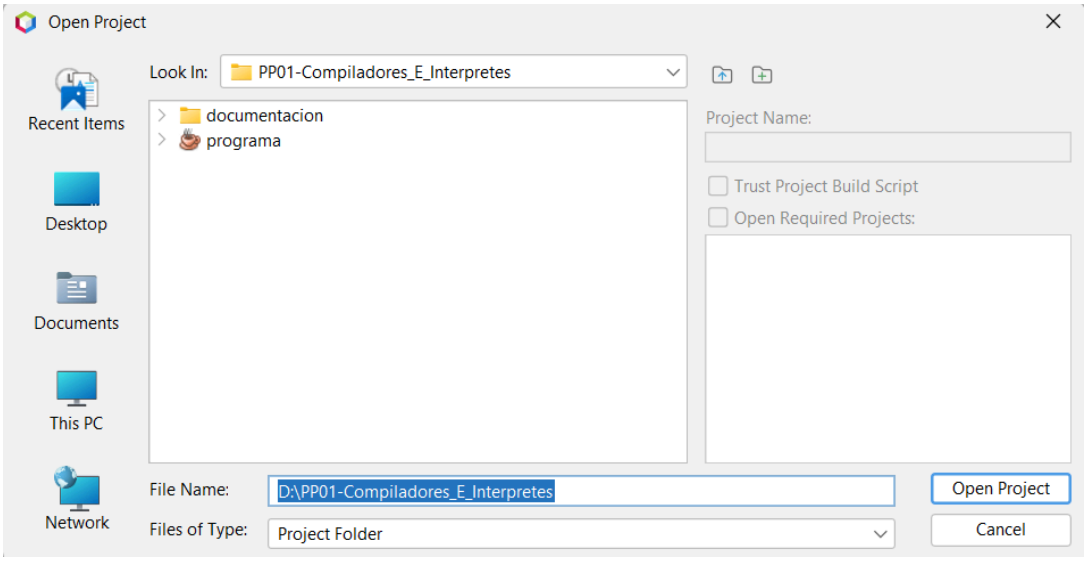
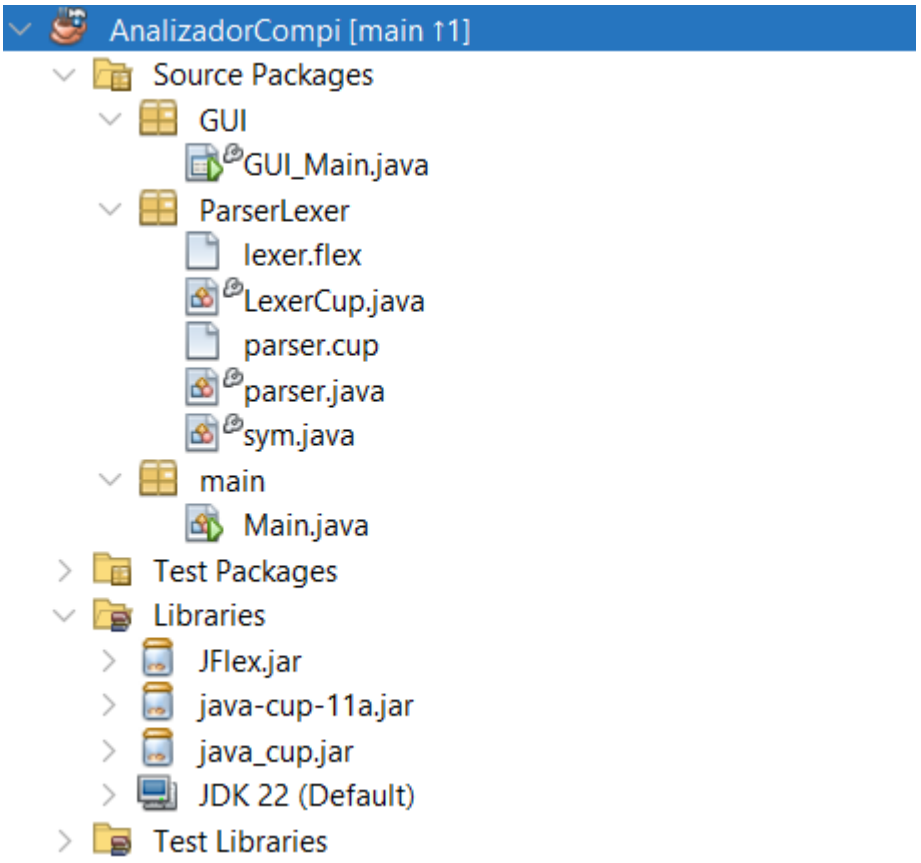


Imagen 3: Proyecto cargado correctamente



5. Instrucciones de compilación y ejecución

a. Generación de Archivos Autogenerados (JFlex y CUP)

Aunque los archivos autogenerados ya están disponibles en el directorio *src/ParserLexer/*, se recomienda regenerarlos para garantizar el correcto funcionamiento del programa y evitar posibles errores.

Para generar los archivos de código fuente (sym.java, parser.java y LexerCup.java) utilizados en el análisis léxico y sintáctico, es necesario compilar y ejecutar la clase principal Main.java, ubicada en el directorio *src/main/*.

Imagen 4: Compilar la clase Main.java

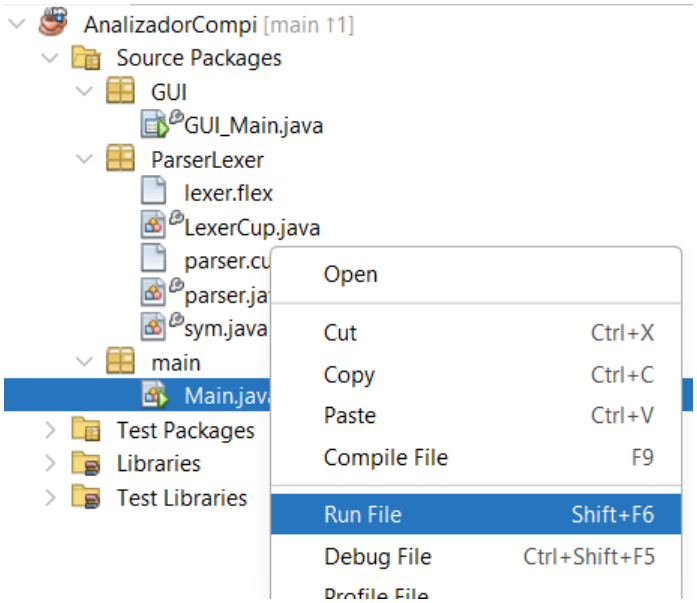
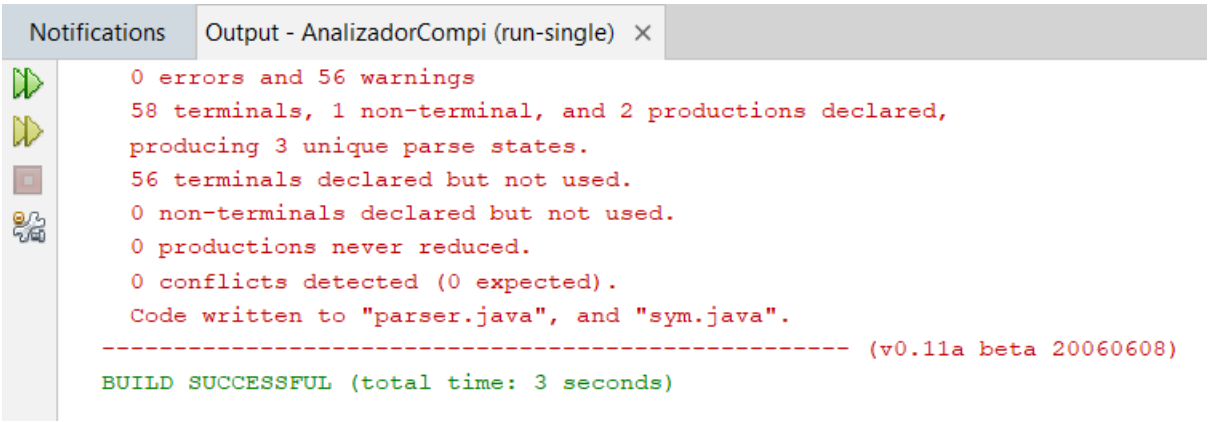


Imagen 5: Verificar la salida en la terminal



b. Probar el análisis léxico

Para probar el análisis léxico utilizando el archivo generado por JFlex (LexerCup.java), el proyecto cuenta con la interfaz gráfica GUI_Main.java, ubicada en el directorio *src/GUI/*. Esta interfaz permite cargar archivos de entrada y visualizar los resultados del análisis léxico de manera interactiva, facilitando el uso y la comprensión del funcionamiento del programa, y si se desea se puede guardar el análisis generado en un archivo TXT.

Imagen 6: Compilar y ejecutar la interfaz para probar el analizador léxico

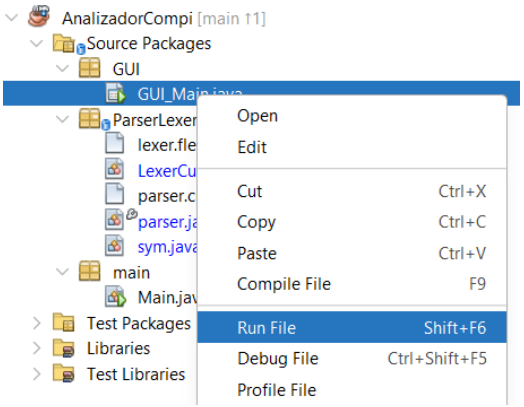
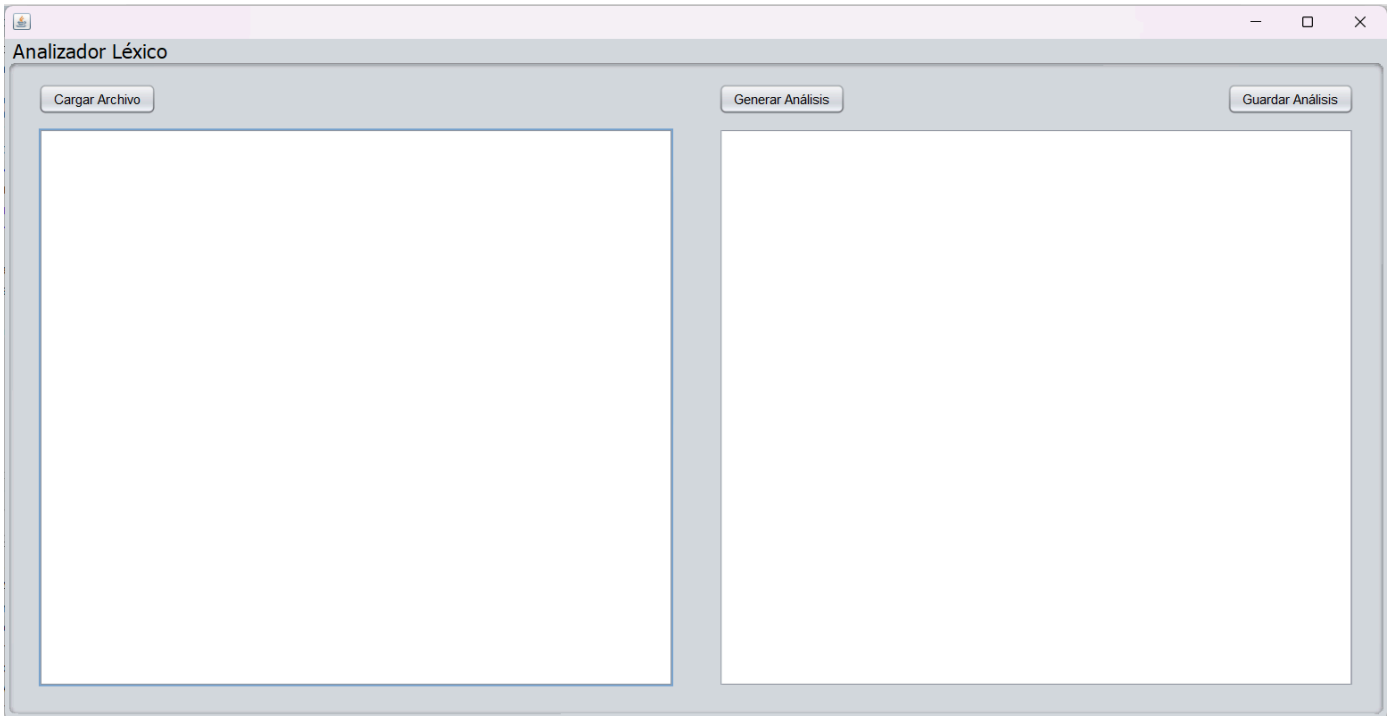


Imagen 7: Interfaz



La interfaz gráfica es muy sencilla, se puede cargar archivos TXT para analizar o se puede escribir directamente en el cuadro de la izquierda, el botón generar muestra el resultado en el cuadro de la derecha. En el siguiente apartado se muestra a detalle su uso completo.

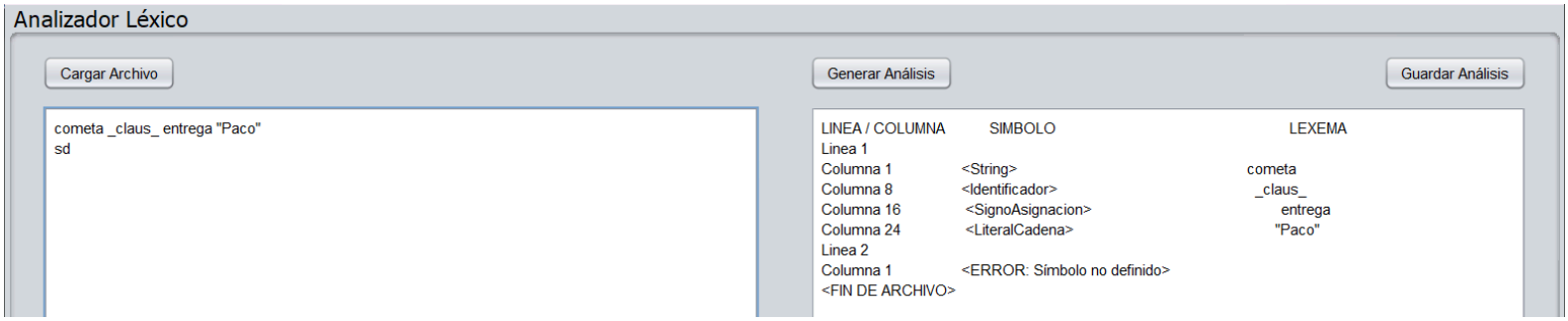
Pruebas de funcionalidad

La interfaz es muy sencilla, se divide en dos secciones (parte izquierda y parte derecha). En la parte izquierda encontramos el botón para cargar un archivo de texto para su posterior análisis y también encontramos el panel de texto (habilitado para su escritura) donde se puede escribir y/o mostrar la información cargada. En la parte derecha encontramos el botón para generar el análisis y para guardarlo, al generar un análisis su resultado se muestra en el panel de texto (no habilitado para su escritura).

1. Probar el análisis con texto libre

Como se puede intuir el panel de texto izquierdo está habilitado para su escritura, esto facilita el análisis puntual y rápido.

Imagen 8: Análisis con texto libre



2. Probar el cargador de archivos

El botón “Cargar Archivo” abre un selector de archivos donde se puede cargar el contenido de un archivo de texto, y ver su información en el panel de texto izquierdo.

Imagen 9: Cargar archivo

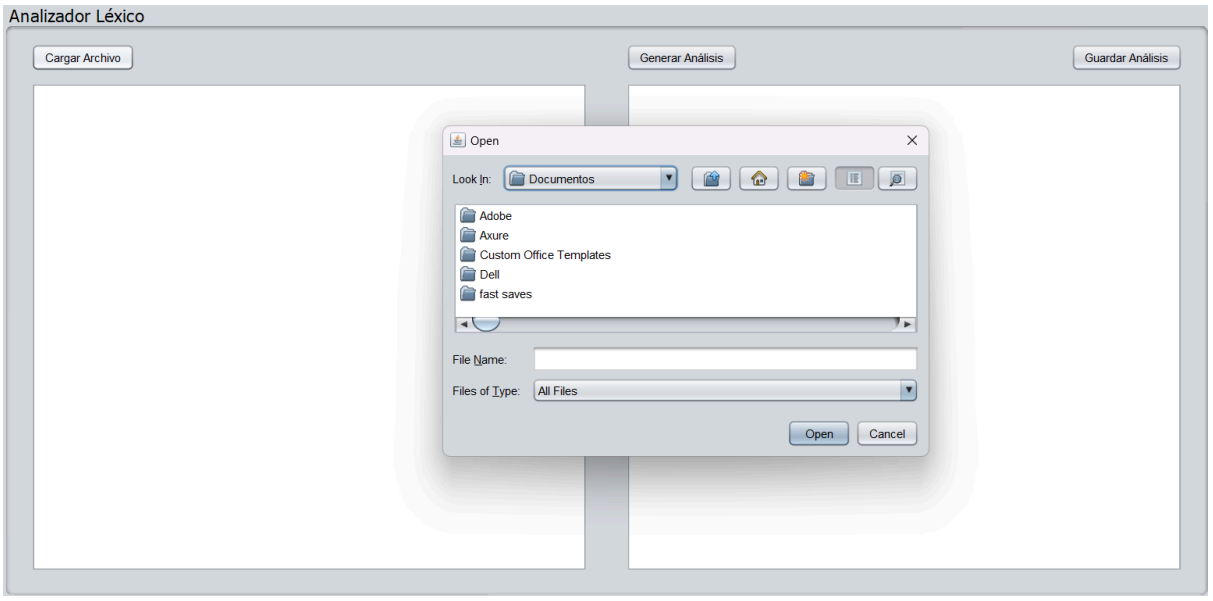
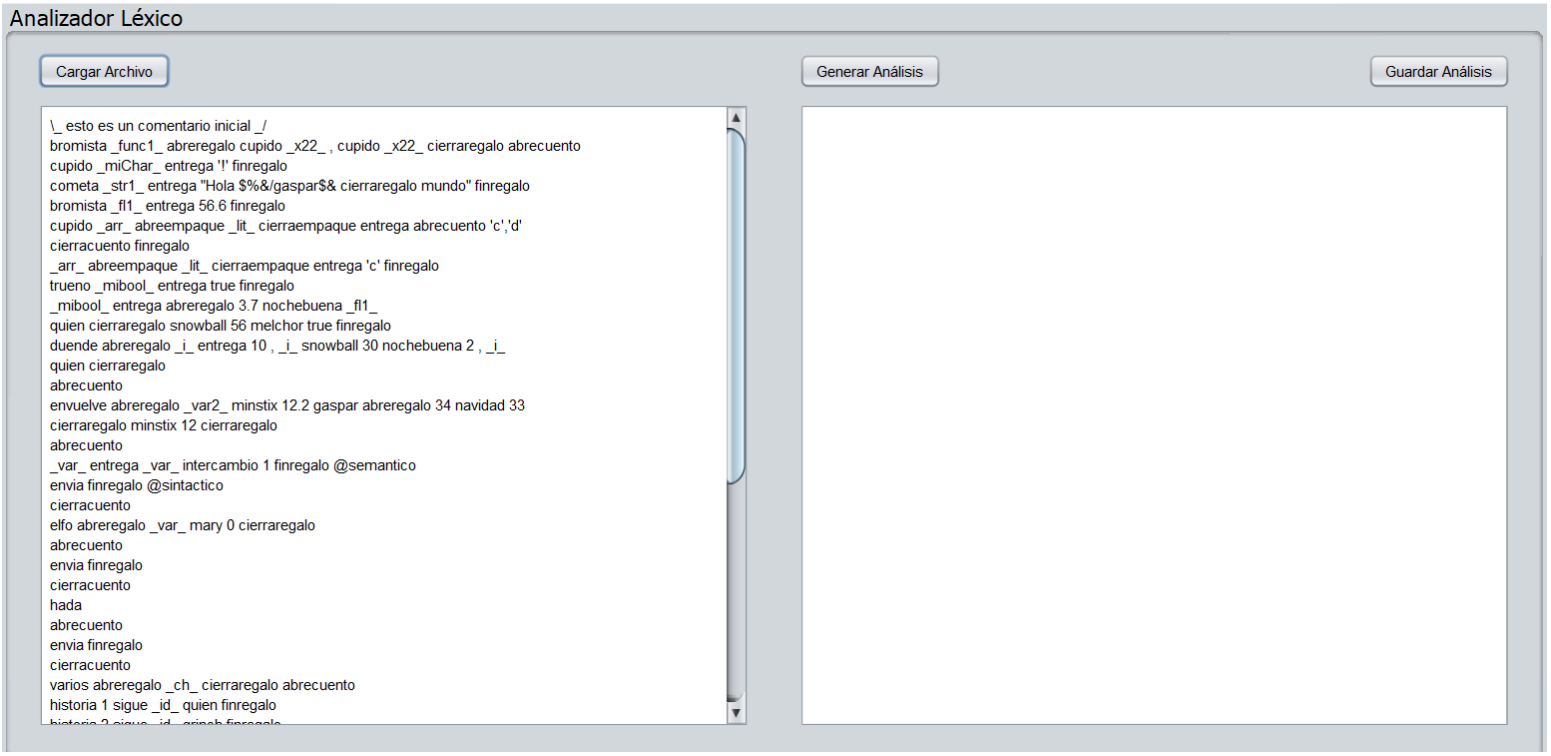


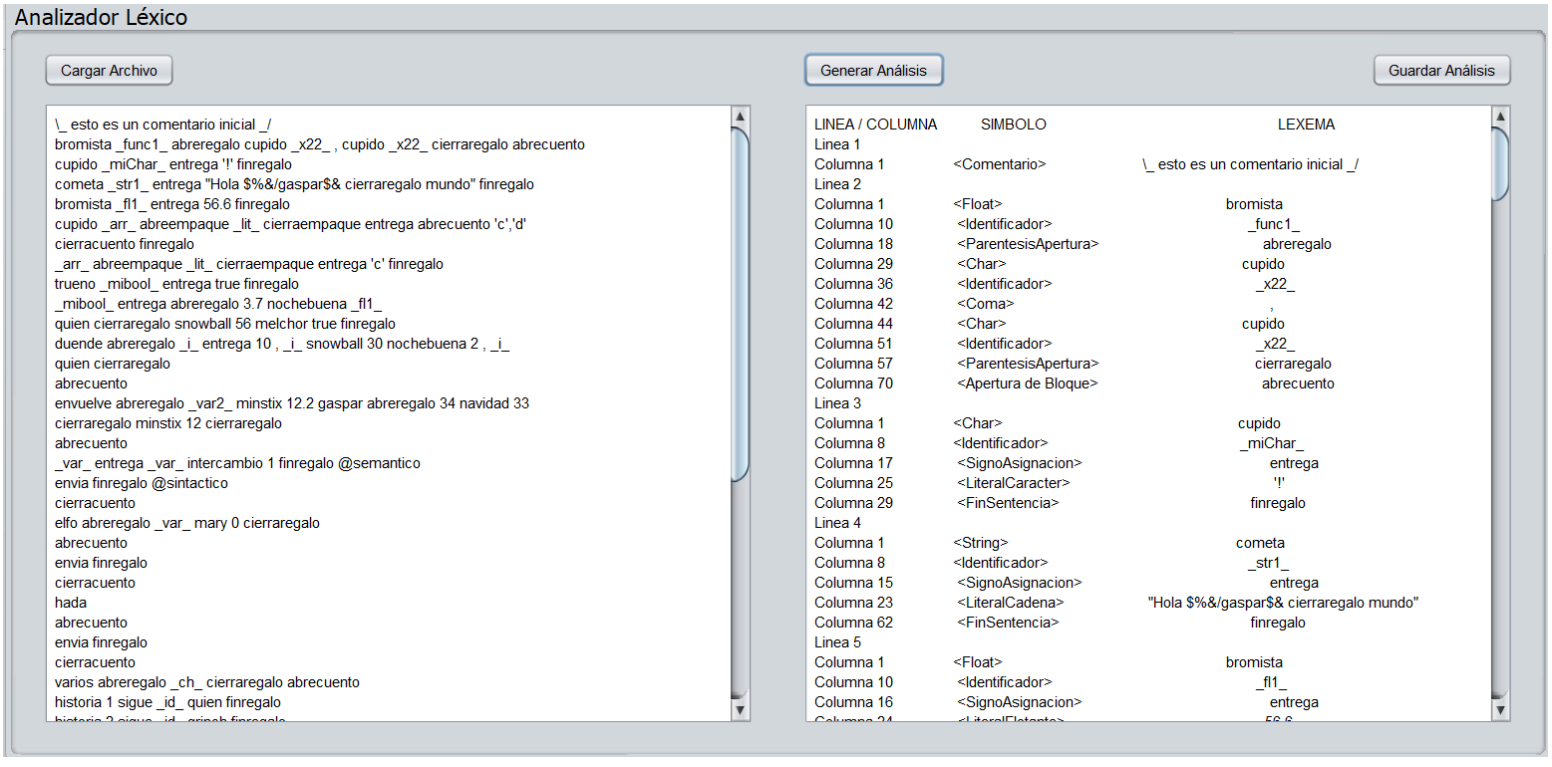
Imagen 10: Contenido cargado



3. Probar la análisis léxico

Con el botón “Generar análisis” se muestra el resultado del análisis léxico donde se puede ver información relevante como la línea/columna donde se encuentra el símbolo y su lexema.

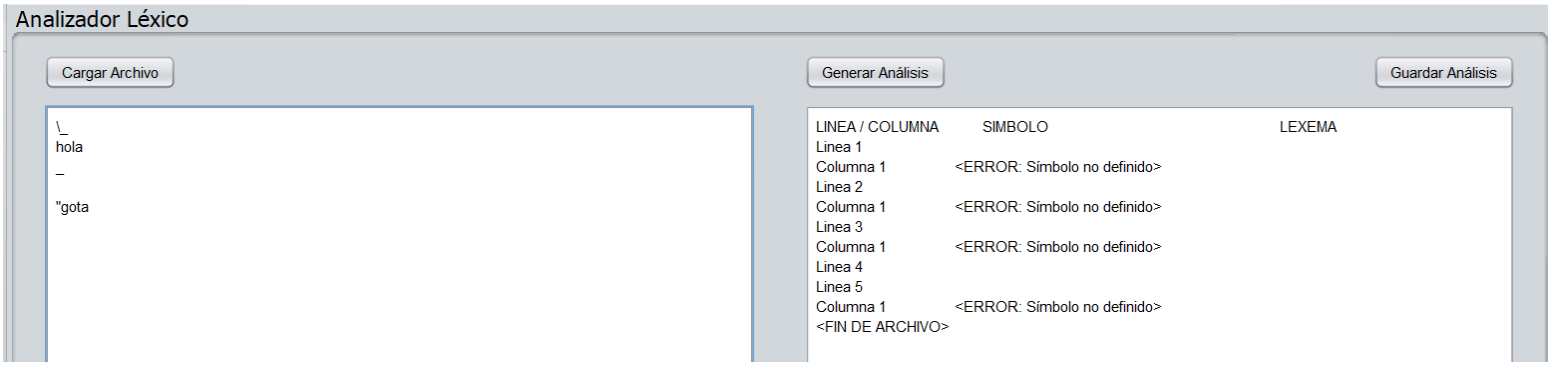
Imagen 11: Análisis léxico



4. Manejo de errores

En caso de que el programa encuentre lexemas no definidos en la gramática analizada, se indica el error correspondiente y donde se encuentra (línea y columna).

Imagen 12: Errores



5. Guardar resultado

Similar al cargado de archivos, el botón “Guardar Análisis” carga un directorio de archivos donde se debe indicar el nombre del archivo y navegar a donde se desea guardar el análisis.

Imagen 13: Guardar resultado

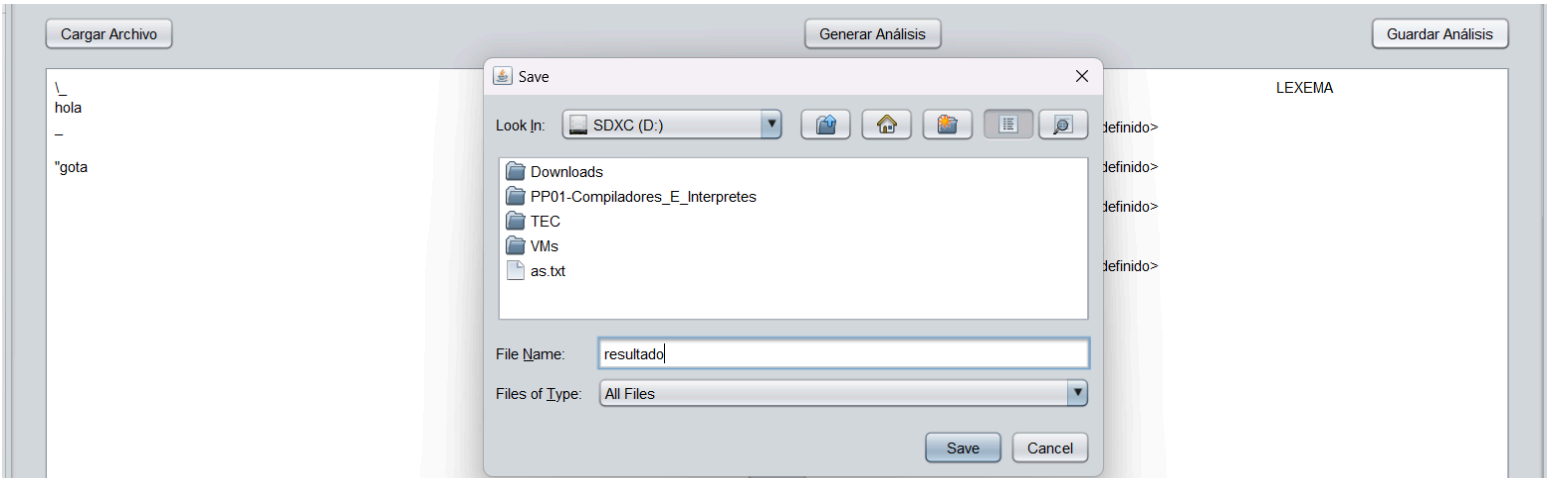
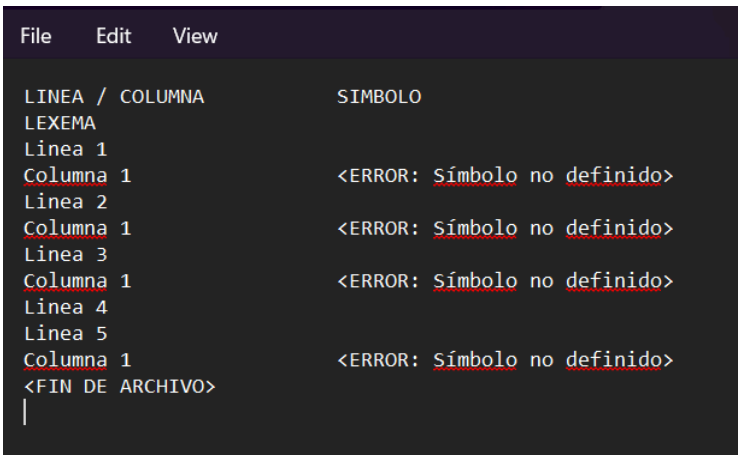


Imagen 14: Archivo de texto generado



Descripción del problema

En este proyecto se nos ha asignado la tarea de desarrollar un scanner (analizador léxico) para un nuevo lenguaje de programación brindado por el profesor. El objetivo principal es procesar programas escritos en este lenguaje, identificando los diferentes componentes léxicos (tokens) a partir de las definiciones gramaticales dadas y reportando cualquier error léxico encontrado durante el análisis.

Retos a resolver:

1. **Definir los tokens del lenguaje:** Basándonos en la gramática proporcionada, debemos identificar y describir los diferentes elementos del lenguaje, como palabras clave, identificadores, operadores y delimitadores.
2. **Implementar el scanner:** Usar JFlex para diseñar un analizador léxico que pueda procesar un archivo fuente, reconocer los tokens y registrar información como el lexema, el tipo de token, la línea y la columna donde aparece.
3. **Manejo de errores:** Incorporar una técnica de recuperación en modo pánico para que el scanner sea capaz de reportar errores léxicos y continuar procesando el archivo fuente.
4. **Documentación:** Crear una descripción detallada del diseño y funcionamiento del scanner, junto con manuales de usuario, análisis de resultados, y pruebas de funcionalidad.

Este proyecto representa un desafío interesante porque combina conceptos teóricos con herramientas prácticas como JFlex y CUP, y además refuerza nuestras habilidades de trabajo en equipo y documentación. Al completar este proyecto, esperamos adquirir experiencia en el desarrollo de herramientas de análisis léxico y otras esenciales para la continuación futura del proyecto.

Diseño del programa

El diseño del programa se basa en la separación de módulos para mejorar la organización y mantenimiento del código. Se implementaron tres módulos principales: el Lexer, que utiliza JFlex para realizar el análisis léxico mediante expresiones regulares avanzadas; el Parser, que utiliza CUP para el análisis sintáctico mediante un algoritmo LR(1); y la Interfaz Gráfica (GUI), que presenta de forma visual y ordenada los tokens generados, mejorando la experiencia del usuario.

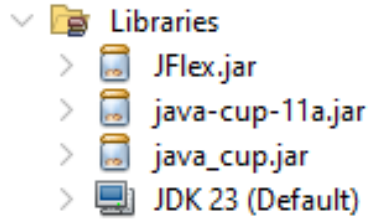
Se emplearon herramientas generadoras como JFlex y CUP, lo que permitió integrar de manera sencilla y eficiente los analizadores léxico y sintáctico. Además, se adoptaron convenciones personalizadas, como palabras clave temáticas (navidad, grinch, melchor), agrupadas por categorías funcionales, para mejorar la legibilidad y modularidad del código. El manejo de errores se realiza mediante un mecanismo de captura de errores en el analizador sintáctico, y el diseño es extensible, permitiendo la inclusión de nuevas construcciones gramaticales.

En cuanto a los algoritmos, el análisis léxico utiliza expresiones regulares para identificar tokens y manejar símbolos con precisión en línea y columna. El análisis sintáctico se realiza mediante un algoritmo LR(1) generado por CUP, y la GUI facilita la presentación de los tokens y captura de errores mediante eventos. El diseño también considera la posibilidad de integrar un análisis semántico en el futuro para validar el uso correcto de variables y tipos de datos.

Las ventajas del diseño incluyen modularidad, flexibilidad y usabilidad, lo que facilita el desarrollo independiente de cada componente. Sin embargo, existen limitaciones como la falta de verificación semántica y la necesidad de optimizar el código generado en futuras fases del proyecto.

Librerías usadas

Las librerías usadas que se necesitaron para la creación de este proyecto:



1. **JFlex.jar:**

   Uso:

   - JFlex es una herramienta para la generación de analizadores léxicos (lexers). Se utiliza para convertir las expresiones regulares definidas en el archivo de especificación en un programa Java que puede leer un flujo de caracteres y reconocer los tokens definidos.

   Aplicación en el proyecto:

   - En este proyecto, se utilizó JFlex para definir las reglas de tokenización de nuestro lenguaje personalizado (por ejemplo, identificar palabras clave, operadores, identificadores, números enteros y flotantes).
   - Los archivos de entrada del programa (fuentes de texto) se procesan mediante JFlex para identificar estos tokens y enviarlos al analizador sintáctico.
   - JFlex genera una clase Lexer que se utiliza para dividir el código fuente en unidades léxicas que el parser puede interpretar.

2. **java-cup-11a.jar / java_cup.jar**

   Uso:

   - CUP es una herramienta de análisis sintáctico que genera parsers en Java. CUP permite definir una gramática para un lenguaje y genera un analizador sintáctico (parser) basado en un análisis LR(1), que se utiliza para construir árboles de sintaxis a partir de las secuencias de tokens generadas por el lexer.

   Aplicación en el proyecto:

   - Se utilizó CUP para generar el analizador sintáctico (parser). CUP toma la gramática definida por el usuario (en nuestro caso, la gramática para nuestro lenguaje específico que brindó el profesor) y genera automáticamente las clases necesarias para analizar el código fuente y comprobar su validez estructural.

- En este proyecto, definimos la gramática para las estructuras del lenguaje (declaraciones, expresiones, instrucciones de control, etc.) y CUP generó las clases que permiten construir el árbol de sintaxis a partir de los tokens proporcionados por el lexer.
- El parser construido por CUP se encarga de verificar que el código fuente esté estructuralmente correcto según las reglas gramaticales y genera un error si se encuentra una violación en la sintaxis.

Análisis de resultados

Objetivos alcanzados

En el desarrollo del proyecto, se lograron cumplir todos los objetivos establecidos al inicio del proyecto. Los siguientes resultados son los que evidencian el cumplimiento de los objetivos:

1. Desarrollo del Analizador Léxico: Se logró implementar exitosamente el analizador léxico usando JFlex. Este analizador fue capaz de identificar correctamente los tokens del código fuente según las reglas de la gramática definida.
2. Implementación del Analizador Sintáctico: Se cumplió con la creación del analizador sintáctico utilizando la librería CUP, que validó correctamente la estructura gramatical del código fuente. El árbol de sintaxis se generó de acuerdo con las reglas establecidas.
3. Integración de Lexer y Parser: Se alcanzó la integración de ambas herramientas (JFlex y CUP) de manera eficiente, logrando que el lexer y el parser trabajaran juntos para procesar el código fuente y verificar su validez.
4. Correcta Ejecución de Pruebas: Todos los casos de prueba definidos para el analizador léxico y sintáctico fueron superados con éxito, lo que confirma que los componentes del proyecto funcionan correctamente**.**

Objetivos no alcanzados

Todos los resultados se lograron dentro del marco de tiempo y los recursos disponibles. Los problemas o desafíos que surgieron durante el desarrollo fueron resueltos a tiempo, y no hubo impedimentos que impidieran alcanzar los objetivos planteados.

Bitácora

Link del repositorio: https://github.com/MITTuu/PP01-Compiladores_E_Interpretes.git