**Nombre: Monjaraz Briseño Luis Fernando**

**Materia: Compiladores**

**Actividad: Hands-on 2 Implementación de Analizadores Léxicos**

**Tema: autómatas**

**Fecha: 09 de marzo de 2025.**

# Índice

[Índice 2](#_Toc192539328)

[Tabla de imágenes 2](#_Toc192539329)

[Implementación de autómatas 4](#_Toc192539330)

[Guía Técnica para Crear Analizadores Léxicos 4](#_Toc192539331)

[Herramientas Necesarias 4](#_Toc192539332)

[Ejercicios Progresivos 5](#_Toc192539333)

[Ejercicio 1 (Básico): Reconocimiento de Palabras Clave, Identificadores y Números 5](#_Toc192539334)

[Ejercicio 2 (Intermedio): Reconocimiento de Comentarios y Cadenas de Texto 6](#_Toc192539335)

[Ejercicio 3 (Avanzado): Conteo de Tokens en un Archivo de Código Fuente 8](#_Toc192539336)

[1. Instalación y Configuración de Herramientas 9](#_Toc192539337)

[2. Diseño de un Archivo .l (Estructura, Reglas Léxicas y Acciones) 10](#_Toc192539338)

[3. Cómo compilar y ejecutar el analizador léxico en cada sistema operativo. 11](#_Toc192539339)

[4. Cómo integrar el analizador con programas en C, Java y Python. 11](#_Toc192539340)

[5. Ejemplos de Código y Salidas 12](#_Toc192539341)

[6. Recomendaciones Adicionales 13](#_Toc192539342)

[7. Buenas Prácticas para Diseñar Especificaciones Léxicas 13](#_Toc192539343)

[8. Recursos Adicionales 13](#_Toc192539344)

[Referencias bibliográficas 15](#_Toc192539345)

# Tabla de imágenes

[Imagen 1 Evidencia de la instalación del ply en Python 5](#_Toc192539346)

[Imagen 2 Reconocimiento de Palabras Clave, Identificadores y Números 5](#_Toc192539347)

[Imagen 3 Resultado. 6](#_Toc192539348)

[Imagen 4 Reconocimiento de Comentarios y Cadenas de Texto 7](#_Toc192539349)

[Imagen 5 Resultado 7](#_Toc192539350)

[Imagen 6 Conteo de Tokens en un Archivo de Código Fuente 8](#_Toc192539351)

[Imagen 7 Resultado 9](#_Toc192539352)

[Imagen 8 Ejemplo de archivo .l 10](#_Toc192539353)

[Imagen 9 Ejemplo en lex con Python 12](#_Toc192539354)

[Imagen 10 Salida 13](#_Toc192539355)

# Implementación de autómatas

## Guía Técnica para Crear Analizadores Léxicos

Esta guía está diseñada para estudiantes de ingeniería en informática y computación que deseen aprender a utilizar herramientas como Lex, Flex, ANTLR y PLY para crear analizadores léxicos. La guía incluye instrucciones detalladas para Windows y Linux, así como ejercicios progresivos para practicar.

## Herramientas Necesarias

Para Windows:

* WinFlex: Herramienta para generar analizadores léxicos en Windows.
* WSL (Windows Subsystem for Linux): Permite ejecutar herramientas de Linux en Windows.
* Cygwin: Entorno similar a Linux para Windows.

Para Linux:

* Flex: Generador de analizadores léxicos.
* Bison: Generador de analizadores sintácticos (útil para proyectos más avanzados).
  + Instalación: sudo apt-get install flex bison

Para Java:

* ANTLR: Herramienta para crear analizadores léxicos y sintácticos en Java.

Para Python:

* PLY (Python Lex-Yacc): Implementación de Lex y Yacc en Python.
  + Instalación: pip install ply

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 1 Evidencia de la instalación del ply en Python

## Ejercicios Progresivos

### Ejercicio 1 (Básico): Reconocimiento de Palabras Clave, Identificadores y Números

Descripción

Crear un analizador léxico que reconozca:

* Palabras clave: int, return.
* Identificadores (nombres de variables).
* Números enteros.

Código en Python (PLY):

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Imagen 2 Reconocimiento de Palabras Clave, Identificadores y Números

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 3 Resultado.

Compilación y ejecución

Linux:

* Generar el código C: flex ejercicio1.l
* Compilar: gcc lex.yy.c -o ejercicio1 -lfl
* Ejecutar: ./ejercicio1 < archivo.txt

Windows (usando WSL o Cygwin):

* Sigue los mismos pasos que en Linux.

### Ejercicio 2 (Intermedio): Reconocimiento de Comentarios y Cadenas de Texto

Descripción

Extender el analizador para reconocer:

* Comentarios de una línea (//) y múltiples líneas (/\* ... \*/).
* Cadenas de texto entre comillas dobles ("...").

Código en Python (PLY):

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Imagen 4 Reconocimiento de Comentarios y Cadenas de Texto

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 5 Resultado

Explicación:

* Cadenas de texto: Se reconocen entre comillas dobles.
* Comentarios de una línea: Empiezan con // y terminan al final de la línea.
* Comentarios de múltiples líneas: Empiezan con /\* y terminan con \*/.

### Ejercicio 3 (Avanzado): Conteo de Tokens en un Archivo de Código Fuente

Descripción

Integrar el analizador léxico con un programa en Python que cuente:

* Número de palabras clave.
* Número de identificadores.
* Número de números.
* Número de operadores y delimitadores.

Código en Python (PLY):

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Imagen 6 Conteo de Tokens en un Archivo de Código Fuente

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 7 Resultado

## 1. Instalación y Configuración de Herramientas

Para Windows

Python:

* Descarga e instala Python desde python.org.
* Asegúrate de marcar la opción "Add Python to PATH" durante la instalación.

PLY (Python Lex-Yacc):

* Instala PLY usando pip: pip install ply

Visual Studio Code:

* Descarga e instala VS Code desde code.visualstudio.com.
* Instala la extensión oficial de Python en VS Code.

Para Linux

Python:

* Python generalmente viene preinstalado en Linux. Si no, instálalo con: sudo apt-get install python3

PLY (Python Lex-Yacc):

* Instala PLY usando pip: pip install ply

Visual Studio Code (opcional):

* Descarga e instala VS Code desde code.visualstudio.com.
* Instala la extensión oficial de Python en VS Code.

## 2. Diseño de un Archivo .l (Estructura, Reglas Léxicas y Acciones)

Un archivo .l en Flex o Lex tiene la siguiente estructura básica:

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 8 Ejemplo de archivo .l

Explicación:

Sección de Declaraciones:

* Aquí se incluyen definiciones de tokens y código en C (opcional).

Reglas Léxicas:

* Cada regla tiene una expresión regular y una acción asociada (por ejemplo, imprimir el token reconocido).

Código en C:

* Puedes incluir funciones adicionales, como main(), para ejecutar el analizador.

## 3. Cómo compilar y ejecutar el analizador léxico en cada sistema operativo.

En Windows (con PLY y Python):

* Guarda el código del analizador léxico en un archivo .py (por ejemplo, lexer.py).
* Ejecuta el archivo con Python: python lexer.py

En Linux (con Flex y Bison):

* Guarda el código del analizador léxico en un archivo .l (por ejemplo, lexer.l).
* Compila el archivo con Flex: flex lexer.l, gcc lex.yy.c -o leer
* Ejecuta el analizador: ./leer

## 4. Cómo integrar el analizador con programas en C, Java y Python.

En Python (PLY):

* El código del analizador léxico ya está integrado en Python. Simplemente ejecuta el archivo .py como se muestra arriba.

En C (Flex):

* Genera el archivo lex.yy.c con Flex.
* Compila el archivo con GCC: gcc lex.yy.c -o leer
* Ejecuta el analizador: ./lexer

En Java (ANTLR):

* Crea un archivo de gramática .g4 para ANTLR.
* Genera el código Java con ANTLR: antlr4 Lexer.g4, javac Lexer\*.java
* Ejecuta el analizador: java Lexer

## 5. Ejemplos de Código y Salidas

Ejemplo 1: Analizador Léxico Básico en Python (PLY)

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 9 Ejemplo en lex con Python

Salida Esperada:

Texto

El contenido generado por IA puede ser incorrecto.

Imagen 10 Salida

## 6. Recomendaciones Adicionales

Cómo Depurar Errores Comunes en el Análisis Léxico

Caracteres No Reconocidos:

* Asegúrate de que todas las expresiones regulares estén correctamente definidas.
* Usa t\_error para manejar caracteres no reconocidos.

Problemas con Espacios y Saltos de Línea:

* Ignora espacios, tabulaciones y saltos de línea con t\_ignore.

Tokens No Definidos:

* Verifica que todos los tokens estén definidos en la lista tokens.

## 7. Buenas Prácticas para Diseñar Especificaciones Léxicas

Mantén las Expresiones Regulares Simples:

* Evita expresiones regulares demasiado complejas.
* Documenta cada token para facilitar el mantenimiento.

Prueba con Diferentes Entradas:

* Prueba el analizador con una variedad de entradas para asegurarte de que funcione correctamente.

## 8. Recursos Adicionales

Documentación Oficial de PLY:

* <http://www.dabeaz.com/ply/>

Tutoriales de Python en VS Code:

* <https://code.visualstudio.com/docs/python/python-tutorial>

Ejemplos de PLY en GitHub:

* <https://github.com/search?q=ply+lexer> Busca repositorios con ejemplos de PLY).

Tutorial de PLY en Real Python:

* <https://realpython.com/ply/>

Documentación Oficial de Python:

* <https://docs.python.org/3/>

Tutorial de Expresiones Regulares en Python:

* <https://docs.python.org/3/howto/regex.html>

Curso de Python en W3Schools:

* <https://www.w3schools.com/python/>

# Referencias bibliográficas

3.13.2 documentation. (n.d.). Retrieved from <https://docs.python.org/3/>

Build software better, together. (n.d.). Retrieved from <https://github.com/search?q=ply+lexer&type=repositories>

Getting Started with Python in VS Code. (2021, November 3). Retrieved from <https://code.visualstudio.com/docs/python/python-tutorial>

Regular Expression HOWTO. (n.d.). Retrieved from <https://docs.python.org/3/howto/regex.html>

W3Schools.com. (n.d.). Retrieved from <https://www.w3schools.com/python/>