

Nombre: Monjaraz Briseño Luis Fernando

Materia: Compiladores

Actividad: Hands-on 2 Implementación de Analizadores Léxicos

Tema: autómatas

Fecha: 09 de marzo de 2025.

Índice

<i>Índice</i>	<i>2</i>
<i>Tabla de imágenes.....</i>	<i>2</i>
<i>Implementación de autómatas.....</i>	<i>4</i>
Guía Técnica para Crear Analizadores Léxicos	4
Herramientas Necesarias.....	4
Ejercicios Progresivos.....	5
Ejercicio 1 (Básico): Reconocimiento de Palabras Clave, Identificadores y Números	5
Ejercicio 2 (Intermedio): Reconocimiento de Comentarios y Cadenas de Texto...	6
Ejercicio 3 (Avanzado): Conteo de Tokens en un Archivo de Código Fuente.....	8
1. Instalación y Configuración de Herramientas	9
2. Diseño de un Archivo .l (Estructura, Reglas Léxicas y Acciones)	10
3. Cómo compilar y ejecutar el analizador léxico en cada sistema operativo.	11
4. Cómo integrar el analizador con programas en C, Java y Python.	11
5. Ejemplos de Código y Salidas	12
6. Recomendaciones Adicionales	13
7. Buenas Prácticas para Diseñar Especificaciones Léxicas	13
8. Recursos Adicionales	13
<i>Referencias bibliográficas</i>	<i>15</i>

Tabla de imágenes

Imagen 1 Evidencia de la instalación del ply en Python	5
Imagen 2 Reconocimiento de Palabras Clave, Identificadores y Números	5

Imagen 3 Resultado.....	6
Imagen 4 Reconocimiento de Comentarios y Cadenas de Texto	7
Imagen 5 Resultado.....	7
Imagen 6 Conteo de Tokens en un Archivo de Código Fuente	8
Imagen 7 Resultado.....	9
Imagen 8 Ejemplo de archivo .l	10
Imagen 9 Ejemplo en lex con Python	12
Imagen 10 Salida.....	13

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`

```
PS D:\betos\Descargas\Tareas\8Vo\Compiladores> pip install ply
Collecting ply
  Obtaining dependency information for ply from https://files.pythonhosted.org/packages/a3/58/35da89ee790598a0700ea49b2a66594140f44dec458c07e8e3d4979137fc/ply-3.11-py2.py3-none-any.whl.metadata
  Downloading ply-3.11-py2.py3-none-any.whl (844 bytes)
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
    49.6/49.6 kB 125.6 kB/s eta 0:00:00
Installing collected packages: ply
Successfully installed ply-3.11

[notice] A new release of pip is available: 23.2.1 -> 25.0.1
[notice] To update, run: C:\Users\betos\AppData\Local\Programs\Python\Python311\python.exe -m pip install --upgrade pip
PS D:\betos\Descargas\Tareas\8Vo\Compiladores>
```

Imagen 1 Evidencia de la instalación del ply en Python

Ejercicios Progresivos

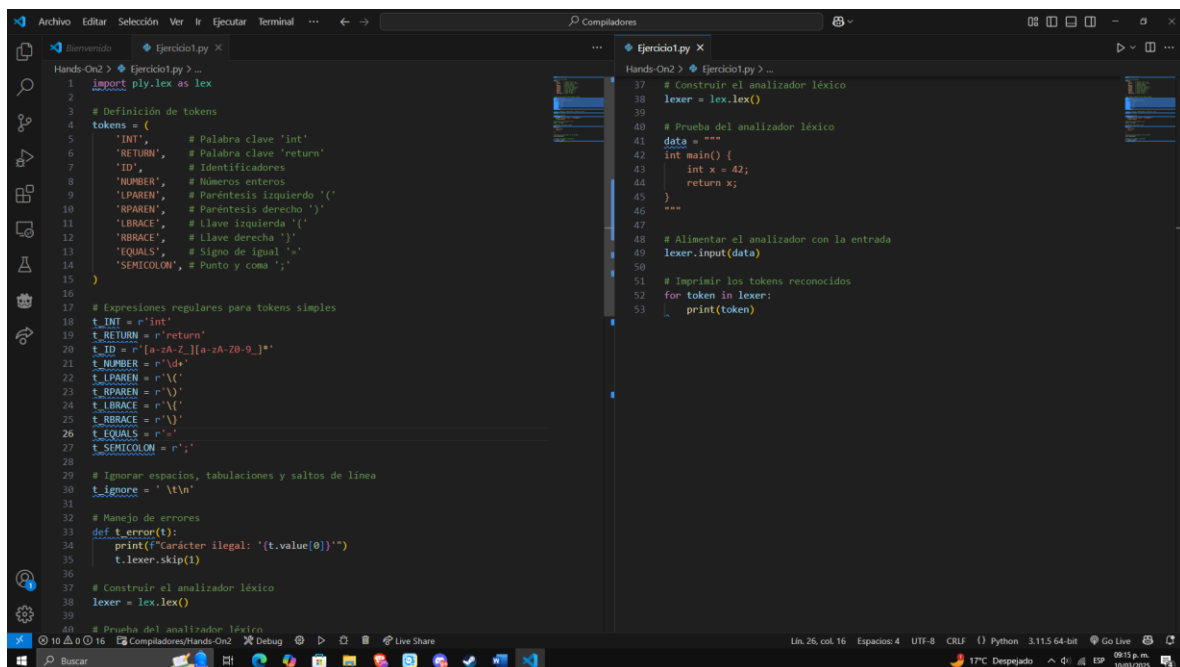
Ejercicio 1 (Básico): Reconocimiento de Palabras Clave, Identificadores y Números

Descripción

Crear un analizador léxico que reconozca:

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

Código en Python (PLY):



```
1 import ply.lex as lex
2
3 # Definición de tokens
4 tokens = (
5     'INT',          # Palabra clave 'int'
6     'RETURN',       # Palabra clave 'return'
7     'ID',           # Identificadores
8     'NUMBER',       # Números enteros
9     'LPAREN',       # Paréntesis izquierdo '('
10    'RPAREN',        # Paréntesis derecho ')'
11    'LBRACE',        # Llave izquierda '{'
12    'RBRACE',        # Llave derecha '}'
13    'EQUALS',        # Signo de igual '='
14    'SEMICOLON',     # Punto y coma ';'
15)
16
17 # Expresiones regulares para tokens simples
18 t_INT = r'int'
19 t_RETURN = r'return'
20 t_ID = r'[a-zA-Z_][a-zA-Z0-9_]*'
21 t_NUMBER = r'[0-9]+'
22 t_LPAREN = r'\('
23 t_RPAREN = r'\)'
24 t_LBRACE = r'\{'
25 t_RBRACE = r'\}'
26 t_EQUALS = r'='
27 t_SEMICOLON = r';'
28
29 # Ignorar espacios, tabulaciones y saltos de línea
30 t_ignore = ' \t\n'
31
32 # Manejo de errores
33 def t_error(t):
34     print(f"Carácter ilegal: '{t.value[0]}'")
35     t.lexer.skip(1)
36
37 # Construye el analizador léxico
38 lexer = lex.lex()
39
40 # Prueba del analizador léxico
41 data = """
42 int main() {
43     int x = 42;
44     return x;
45 }
46 """
47
48 # Alimentar el analizador con la entrada
49 lexer.input(data)
50
51 # Imprimir los tokens reconocidos
52 for token in lexer:
53     print(token)
```

Imagen 2 Reconocimiento de Palabras Clave, Identificadores y Números

```
PS D:\beto5\Descargas\Tareas\8Vo\Compiladores> & C:/
LexToken(ID, 'int', 1, 1)
LexToken(ID, 'main', 1, 5)
LexToken(LPAREN, '(', 1, 9)
LexToken(RPAREN, ')', 1, 10)
LexToken(LBRACE, '{', 1, 12)
LexToken(ID, 'int', 1, 18)
LexToken(ID, 'x', 1, 22)
LexToken(EQUALS, '=', 1, 24)
LexToken(NUMBER, '42', 1, 26)
LexToken(SEMICOLON, ';', 1, 28)
LexToken(ID, 'return', 1, 34)
LexToken(ID, 'x', 1, 41)
LexToken(SEMICOLON, ';', 1, 42)
LexToken(RBRACE, '}', 1, 44)
PS D:\beto5\Descargas\Tareas\8Vo\Compiladores> █
```

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):

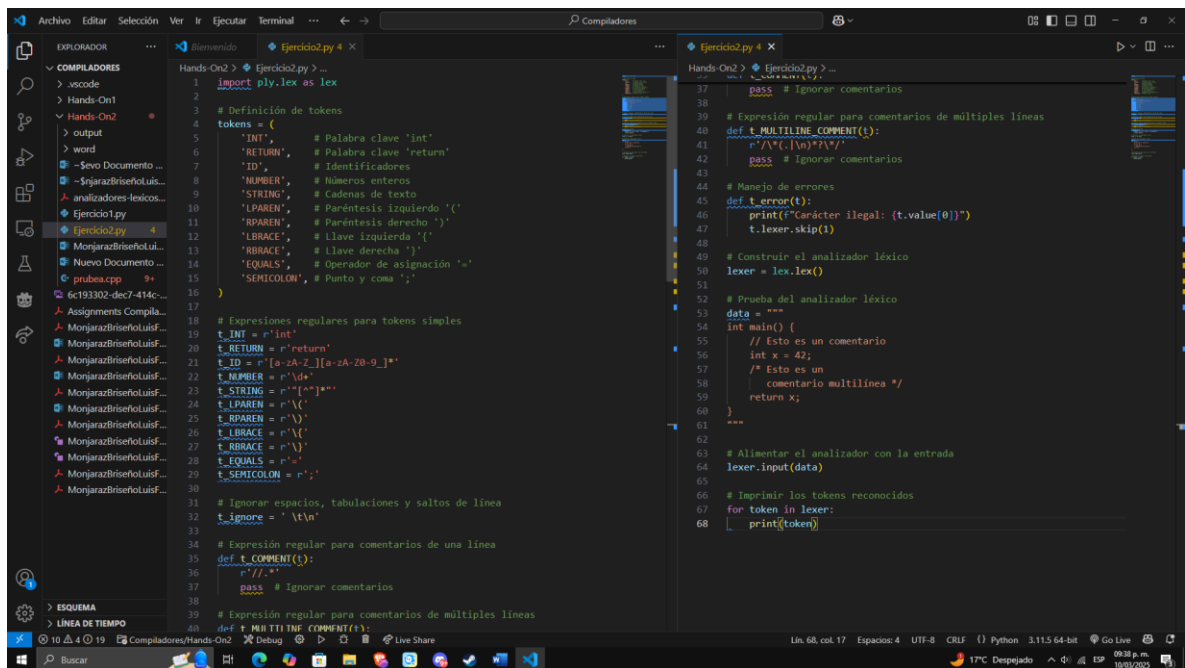


Imagen 4 Reconocimiento de Comentarios y Cadenas de Texto

```

PS D:\beto5\Descargas\Tareas\8Vo\
py
LexToken(ID, 'int',1,1)
LexToken(ID, 'main',1,5)
LexToken(LPAREN, '(',1,9)
LexToken(RPAREN, ')',1,10)
LexToken(LBRACE, '{',1,12)
LexToken(ID, 'int',1,47)
LexToken(ID, 'x',1,51)
LexToken(EQUALS, '=',1,53)
LexToken(NUMBER, '42',1,55)
LexToken(SEMICOLON, ';',1,57)
LexToken(ID, 'return',1,113)
LexToken(ID, 'x',1,120)
LexToken(SEMICOLON, ';',1,121)
LexToken(RBRACE, '}',1,123)

```

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.

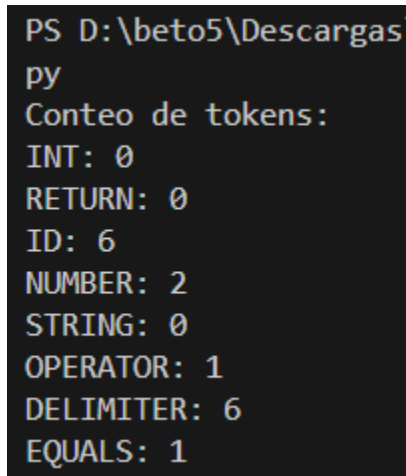
Código en Python (PLY):

```

1  import ply.lex as lex
2
3  # Definición de tokens
4  tokens = (
5      'INT',          # Palabra clave 'int'
6      'RETURN',      # Palabra clave 'return'
7      'ID',          # Identificadores
8      'NUMBER',      # Números enteros
9      'STRING',      # Cadenas de texto
10     'OPERATOR',     # Operadores (+, -, *, /)
11     'DELIMITER',    # Delimitadores (,, (, )
12     'EQUALS',       # Operador de asignación '='
13 )
14
15 # Expresiones regulares para tokens simples
16 t_INT = r'int'
17 t_RETURN = r'return'
18 t_ID = r'[a-zA-Z][a-zA-Z0-9_]*'
19 t_NUMBER = r'\d+'
20 t_STRING = r'"([^"]|\\")*"'
21 t_OPERATOR = r'[+,-,*,/]' # Corregido: guión escapado correctamente
22 t_DELIMITER = r'[(),,]' # Corregido: caracteres escapados correctamente
23 t_EQUALS = r'='
24
25 # Ignorar espacios, tabulaciones y saltos de línea
26 t_ignore = ' \t\n'
27
28 # Expresión regular para comentarios de una línea
29 def t_COMMENT(t):
30     r'//.*'
31     pass # Ignorar comentarios
32
33 # Expresión regular para comentarios de múltiples líneas
34 def t_MULTILINE_COMMENT(t):
35     r'/*.**/'
36     pass # Ignorar comentarios
37
38 # Manejo de errores
39 def t_error(t):
40     print(f"Carácter ilegal: {t.value[0]}")
41     return None
42
43 # Contadores
44 counters = {
45     'INT': 0,
46     'RETURN': 0,
47     'ID': 0,
48     'NUMBER': 0,
49     'STRING': 0,
50     'OPERATOR': 0,
51     'DELIMITER': 0,
52     'EQUALS': 0,
53 }
54
55 # Prueba del analizador léxico
56 data = """
57 int main() {
58     int x = 42;
59     return x + 1;
60 }
61 """
62
63 # Alimentar el analizador con la entrada
64 lexer.lex(data)
65
66 # Contar tokens
67 for token in lexer:
68     if token.type in counters:
69         counters[token.type] += 1
70
71 # Imprimir resultados
72 print("Conteo de tokens:")
73 for token_type, count in counters.items():
74     print(f"{token_type}: {count}")

```

Imagen 6 Conteo de Tokens en un Archivo de Código Fuente



```
PS D:\beto5\Descargas
py
Conteo de tokens:
INT: 0
RETURN: 0
ID: 6
NUMBER: 2
STRING: 0
OPERATOR: 1
DELIMITER: 6
EQUALS: 1
```

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:

```
%{  
  // Código en C (opcional)  
%}  
  
// Definición de tokens  
%%  
  
// Reglas léxicas y acciones  
"int"      { printf("Palabra clave: %s\n", yytext); }  
[0-9]+     { printf("Número: %s\n", yytext); }  
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identificador: %s\n", yytext); }  
  
%%  
  
// Código en C (opcional)  
int main(int argc, char **argv) {  
    yylex();  
    return 0;  
}
```

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)

```
import ply.lex as lex

tokens = ('INT', 'RETURN', 'ID', 'NUMBER')

t_INT = r'int'
t_RETURN = r'return'
t_ID = r'[a-zA-Z_][a-zA-Z0-9_]*'
t_NUMBER = r'\d+'

t_ignore = ' \t\n'

def t_error(t):
    print(f"Carácter ilegal: '{t.value[0]}'")
    t.lexer.skip(1)

lexer = lex.lex()

data = """
int main() {
    int x = 42;
    return x;
}
"""

lexer.input(data)

for token in lexer:
    print(token)
```

Imagen 9 Ejemplo en lex con Python

Salida Esperada:

```
LexToken(INT, 'int', 2, 1)
LexToken(ID, 'main', 2, 5)
LexToken(ID, 'x', 3, 9)
LexToken(NUMBER, '42', 3, 13)
LexToken(RETURN, 'return', 4, 5)
LexToken(ID, 'x', 4, 12)
```

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/>