Universidad La Salle
Facultad de Ingenierías
Departamento de Ingenierías y Matemáticas
Carrera Profesional de Ingeniería de Software

Alumno: Vizarreta Checya Carlos

March 26, 2025

Informe del Analizador Léxico en PLY

Especificación Léxica del Lenguaje de Programación "NEW C+"

1 Introducción

El siguiente informe describe el funcionamiento del analizador léxico desarrollado en Python utilizando la librería PLY. Este analizador léxico tiene la tarea de leer una secuencia de caracteres de entrada y dividirla en **tokens**, los cuales representan las unidades léxicas del lenguaje.

2 Objetivo del Analizador Léxico

El objetivo del código es reconocer los siguientes elementos en una entrada de texto:

- Identificadores y palabras clave.
- Números enteros y decimales.
- Operadores aritméticos, relacionales y lógicos.
- Paréntesis, llaves y punto y coma.
- Comentarios de una línea.

3 Lista de Tokens

A continuación, se detallan los tokens que reconoce el analizador:

4 Funcionamiento del Código

El analizador léxico se basa en la librería PLY y sigue estos pasos:

Token	Expresión Regular	
PALABRA_CLAVE	\b(int|float|if|else|while|for|return|print|let|var|const)\b	
IDENTIFICADOR	[a-zA-Z_][a-zA-Z0-9_]*	Dete
NUM_ENTERO	\d+	
NUM_DECIMAL	\d+\.\d+	De
OP_ARITMETICO	[+*/%]	
OP_RELACIONAL	==|!=|<=|>=|<|>	
OP_LOGICO	&&|||	
PARENTESIS	[()]	
LLAVE	[{}]	
PUNTO_COMA	;	
COMENTARIO	//.*	

Table 1: Tokens reconocidos por el analizador léxico

4.1 1. Importación de PLY

Se importa `ply.lex` para definir las reglas de tokenización en Python.

4.2 2. Definición de tokens

Se declara una lista con los nombres de los tokens y sus expresiones regulares.

4.3 3. Reglas de reconocimiento de tokens

- Las palabras clave se detectan en la función `t_PALABRA_CLAVE`.
- `t_IDENTIFICADOR` reconoce nombres de variables y funciones.
- Se diferencian los números enteros (`t_NUM_ENTERO`) de los decimales (`t_NUM_DECIMAL`).
- Se incluyen reglas para reconocer operadores (`OP_ARITMETICO`, `OP_RELACIONAL`, `OP_LOGICO`).
- Se detectan los símbolos de puntuación: paréntesis, llaves y punto y coma.
- Se ignoran los espacios en blanco y los comentarios.

4.4 4. Manejo de errores

Si el analizador encuentra un carácter no reconocido, la función `t_error` lo detecta e imprime un mensaje.

4.5 5. Ejecución del analizador léxico

Se proporciona un código de prueba en "NEW C+" y se imprimen los tokens detectados.

5 Ejemplo de Ejecución

Código de prueba:

```
int x = 10;
float y = 3.14;
if (x > y) {
    print(x);
}
```

Salida esperada:

```
LexToken(PALABRA_CLAVE,'int',1,0)
LexToken(IDENTIFICADOR,'x',1,4)
LexToken(OP_ARITMETICO,'=',1,6)
LexToken(NUM_ENTERO,'10',1,8)
LexToken(PUNTO_COMA,';',1,10)
LexToken(PALABRA_CLAVE,'float',2,12)
LexToken(IDENTIFICADOR,'y',2,18)
LexToken(OP_ARITMETICO,'=',2,20)
LexToken(NUM_DECIMAL,'3.14',2,22)
LexToken(PUNTO_COMA,';',2,26)
LexToken(PALABRA_CLAVE,'if',3,28)
LexToken(PARENTESIS,'(',3,31)
LexToken(IDENTIFICADOR,'x',3,32)
LexToken(OP_RELACIONAL,'>',3,34)
LexToken(IDENTIFICADOR,'y',3,36)
LexToken(PARENTESIS,')',3,37)
LexToken(LLAVE,'{',3,39)
LexToken(PALABRA_CLAVE,'print',4,43)
LexToken(PARENTESIS,'(',4,48)
LexToken(IDENTIFICADOR,'x',4,49)
LexToken(PARENTESIS,')',4,50)
LexToken(PUNTO_COMA,';',4,51)
LexToken(LLAVE,'}',5,53)
```

6 Conclusión

Este analizador léxico desarrollado con PLY es capaz de reconocer correctamente los componentes básicos del lenguaje de programación “NEW C+”, permitiendo su futura integración en un compilador o intérprete. Su diseño modular permite la expansión para admitir más tokens y mejorar la detección de errores.