Especificación Léxica del Lenguaje SKRN

Hugo Alonso Youzzueff Diaz Chavez
Juan José Huamaní Vásquez
Melvin Jarred Yabar Carazas
Gabriel Frank Krisna Zela Flores

20 de Marzo de 2025

Universidad La Salle - Escuela Profesional de Ingeniería de Software
Curso: Compiladores
Docente: Vicente Enrique Machaca Arceda

1 Introducción

El lenguaje **SKRN** es un lenguaje de programación diseñado para facilitar el aprendizaje de programación utilizando palabras clave en español escritas al revés. En este informe, se detalla la implementación del analizador léxico para este lenguaje.

2 Marco Teórico

El analizador léxico es la primera fase de un compilador. Su tarea principal es transformar el código fuente en una secuencia de tokens reconocibles por el analizador sintáctico. Para este proyecto, se utiliza la librería **PLY (Python Lex-Yacc)**.

3 Definición de Tokens

La siguiente tabla describe los tokens reconocidos por el lenguaje SKRN:

Token	**Expresión Regular**	**Descripción**
ID	[a-zA-Z][a-zA-Z0-9]*	Identificadores
INTEGER	[0-9]+	Números enteros
FLOATING	[0-9]+.[0-9]+([eE][+-]?[0-9]+)?	Números flotantes
ASSIGN	=	Operador de asignación
PLUS	+	Operador de suma
MINUS	-	Operador de resta
TIMES	*	Operador de multiplicación
DIVIDE	/	Operador de división
LPAREN	(	Paréntesis izquierdo
RPAREN	)	Paréntesis derecho
LBRACE	{	Llave izquierda
RBRACE	}	Llave derecha
IF	fi	Condicional if
ELSE	esle	Condicional else
RETURN	nruter	Retorno de función

4 Implementación del Analizador Léxico

A continuación, se presenta el código en Python para la implementación del analizador léxico utilizando PLY:

```
import ply.lex as lex

# Lista de nombres de los tokens (seg n tu tabla de tokens)
tokens = [
    'ID', 'INTEGER', 'FLOATING',
    'ASSIGN', 'PLUS', 'MINUS', 'TIMES', 'DIVIDE', 'MODULO',
    'INCREMENT', 'DECREMENT', 'EQUAL', 'NOT_EQUAL',
    'LESS', 'GREATER', 'LESS_EQUAL', 'GREATER_EQUAL',
    'LOGICAL_AND', 'LOGICAL_OR', 'LOGICAL_NOT',
    'PLUS_ASSIGN', 'MINUS_ASSIGN', 'TIMES_ASSIGN', '
       DIV_ASSIGN',
    'TERNARY_Q', 'TERNARY_C', 'MEMBER_ACCESS', '
       POINTER_ACCESS',
    'LPAREN', 'RPAREN', 'LBRACE', 'RBRACE', 'LBRACKET', '
       RBRACKET',
    'SEMICOLON', 'COMMA', 'STRING', 'CHARACTER'
]

# Palabras reservadas (reversed)
reserved = {
    'tni': 'type_int',
    'taolf': 'type_float',
    'fi': 'IF',
    'esle': 'ELSE',
    'nruter': 'RETURN'
}

tokens += list(reserved.values())

# Expresiones regulares de operadores y delimitadores
t_ASSIGN = r'='
t_PLUS = r'\+'
t_MINUS = r'-'
t_TIMES = r'\*'
t_DIVIDE = r'/'
t_MODULO = r'%'
t_EQUAL = r'=='
t_NOT_EQUAL = r'!='
t_LPAREN = r'\('
t_RPAREN = r'\)'
t_LBRACE = r'\{'
t_RBRACE = r'\}'

lexer = lex.lex()

data = """
tni niam() {
    tuoc << "Hola, Mundo!"
```

```
    nruter
}
"""
lexer.input(data)

while True:
    tok = lexer.token()
    if not tok:
        break
    print(tok)
```

5 Tokens Generados

Ejecutando el código anterior, se obtiene la siguiente salida de tokens:

```
Token(type='type_int', lexeme='tni', line=1, column=1)
Token(type='ID', lexeme='niam', line=1, column=5)
Token(type='LPAREN', lexeme='(', line=1, column=9)
Token(type='RPAREN', lexeme=')', line=1, column=10)
Token(type='LBRACE', lexeme='{', line=2, column=1)
Token(type='ID', lexeme='tuoc', line=3, column=5)
Token(type='STRING', lexeme='"Hola, Mundo!"', line=3, column=10)
Token(type='RETURN', lexeme='nruter', line=4, column=5)
Token(type='RBRACE', lexeme='}', line=5, column=1)
```

6 Conclusión

La implementación del analizador léxico para el lenguaje SKRN demuestra la viabilidad de utilizar palabras clave en español invertidas para el aprendizaje de programación. La combinación de PLY y expresiones regulares permite reconocer tokens de manera eficiente.