Implementación del Analizador Léxico

Universidad La Salle
Escuela Profesional de Ingeniería de Software

2025-03-26

Introducción

Este documento describe la implementación de un analizador léxico utilizando la biblioteca `Ply` de Python. El analizador está diseñado para reconocer tokens específicos de un lenguaje propuesto.

El análisis léxico es una fase fundamental en la construcción de compiladores, encargándose de transformar la entrada de texto en una secuencia de tokens. Estos tokens representan unidades léxicas reconocibles del lenguaje, como palabras clave, operadores y símbolos especiales.

Especificación Léxica

Los tokens que reconoce el analizador son:

- Variables, constantes y tipos de datos como `DT_INT`, `DT_FLOAT`, `DT_BOOL`, entre otros.
- Estructuras de control como `IF`, `ELSE`, `SWITCH`, `FOR`, `WHILE`, etc.
- Operadores aritméticos como `+`, `-`, `*`, `/` y `%`.
- Operadores de flujo de entrada/salida como `COUT`, `CIN`, `O_IN` y `O_EX`.
- Operadores relacionales como `==`, `!=`, `>`, `<`, `>=`, `<=`.
- Símbolos especiales como paréntesis, llaves y punto y coma.

Código Fuente

A continuación se presenta la implementación del analizador léxico en Python:

```
import ply.lex as lex

tokens = (
    'DT_STRING',
    'VAR_KEYWORD', 'VAR', 'CONST', 'DT_INT', 'DT_FLOAT', 'DT_CHAR', 'DT_BO
    'IF', 'ELSE', 'SWITCH', 'FOR', 'WHILE', 'DO_WHILE',
    'RETURN', 'BREAK', 'CONTINUE',
    'COUT', 'CIN', 'O_IN', 'O_EX', 'ENDLINE',
    'O_SUMA', 'O_REST', 'O_MULT', 'O_DIV', 'O_MOD',
    'O_IGUALDAD', 'O_DIFERENTE', 'O_MAYOR', 'O_MENOR', 'O_MAYORIGUAL', 'O_
    'SYM_PAR_IZQ', 'SYM_PAR_DER', 'SYM_LLAVE_IZQ', 'SYM_LLAVE_DER',
    'FUNCTION', 'E_INS', 'COM_SIMPLE', 'COM_MULTI'
)

t_O_SUMA = r'\\+'
t_O_REST = r'-'
t_O_MULT = r'\\*'
t_O_DIV = r'/'
t_O_MOD = r'%'
t_O_IGUALDAD = r'=='
t_O_DIFERENTE = r'!='
t_O_MAYOR = r'>'
t_O_MENOR = r'<'
t_O_MAYORIGUAL = r'>='
t_O_ASIG = r'='
t_SYM_PAR_IZQ = r'\\('
t_SYM_PAR_DER = r'\\)'
t_SYM_LLAVE_IZQ = r'\\{'
t_SYM_LLAVE_DER = r'\\}'
t_O_IN = r'<<'
t_O_EX = r'>>'
t_ENDLINE = r'ENDLINE'
t_E_INS = r';'

def t_VAR_KEYWORD(t):
    r'VAR(?![a-zA-Z0-9-])'
    return t

def t_VAR(t):
    r'V-[a-zA-Z]+-[0-9]+'
    return t

def t_DT_INT(t):
    r'(DT-INT|[0-9]+)'
    if t.value.isdigit():
        t.value = int(t.value)
    return t

def t_DT_STRING(t):
    r'\".+\"'
    return t

lexer = lex.lex()
```

Pruebas y Resultados

Para verificar la funcionalidad del analizador léxico, se realizaron pruebas con distintos fragmentos de código de entrada. Se analizaron los siguientes casos:

1. Un código simple de "Hola Mundo".
2. Un código que usa estructuras de control como condicionales y bucles.
3. Un código que define y utiliza funciones.

Cada uno de estos casos produjo una lista de tokens correctamente reconocidos por el

analizador. Los resultados obtenidos muestran que el analizador identifica correctamente los tokens definidos y distingue entre palabras clave, identificadores y operadores.

Conclusión

El uso de `Ply` facilita la implementación de un analizador léxico eficiente y modular. Se reconoce correctamente la sintaxis del lenguaje propuesto, permitiendo la identificación de distintos tipos de tokens.

Esta implementación sienta las bases para futuras mejoras en la creación de compiladores más avanzados, integrando la fase de análisis sintáctico y semántico.