Centro Universitario de Ciencias Exactas e Ingeniería

TRADUCTORES DE LENGUAJES ll

Practica 4

JULIO ESTEBAN VALDES LOPEZ

Juan Pablo Hernández Orozco

219294285

Objetivo:

Implementar un analizador sintáctico descendente predictivo en Python —con analizador léxico capaz de reconocer identificadores y números de más de un dígito—, usando la gramática LL(1) transformada de la página 360 del *“libro del dragón”*, e incluyendo operadores de multiplicación (*) y división (/). La salida debe ser una expresión en notación postfija o, en su defecto, un mensaje de error descriptivo.

Introducción:

En la construcción de compiladores, el análisis sintáctico es una etapa crítica para validar que la secuencia de tokens generada por el analizador léxico forma estructuras gramaticalmente correctas. Un método clásico es el **análisis descendente predictivo**, que utiliza una gramática libre de ambigüedades y factorizada para anticipar la producción aplicable basándose en el próximo token de entrada.

Para este proyecto se usa una gramática LL(1) que describe expresiones aritméticas compuestas por identificadores, números de más de un dígito, paréntesis, y operadores aritméticos básicos (+, -, *, /). Se integró un **analizador léxico** que utiliza expresiones regulares para separar adecuadamente los tokens.

El resultado del análisis es la conversión de expresiones aritméticas infijas a su forma **postfija** (o notación polaca inversa), facilitando su posterior evaluación o transformación.

Además, la gramática original fue extendida para incluir los operadores de **multiplicación** y **división**, siguiendo la estructura:

- **E → T E'**
- **E' → + T E' | - T E' | ε**
- **T → F T'**
- **T' → * F T' | / F T' | ε**
- **F → (E) | id | num**

Esta estructura garantiza el correcto orden de precedencia entre las operaciones de suma/resta y multiplicación/división.

Problemas: recursión por la izquierda (expr → expr + term) y ambigüedad en precedencia/asociatividad.

Gramática transformada (LL(1), p. 360):

E → T E′

E′ → + T E′ | - T E′ | ε

T → F T′

T′ → * F T′ | / F T′ | ε

F → (E) | id | num

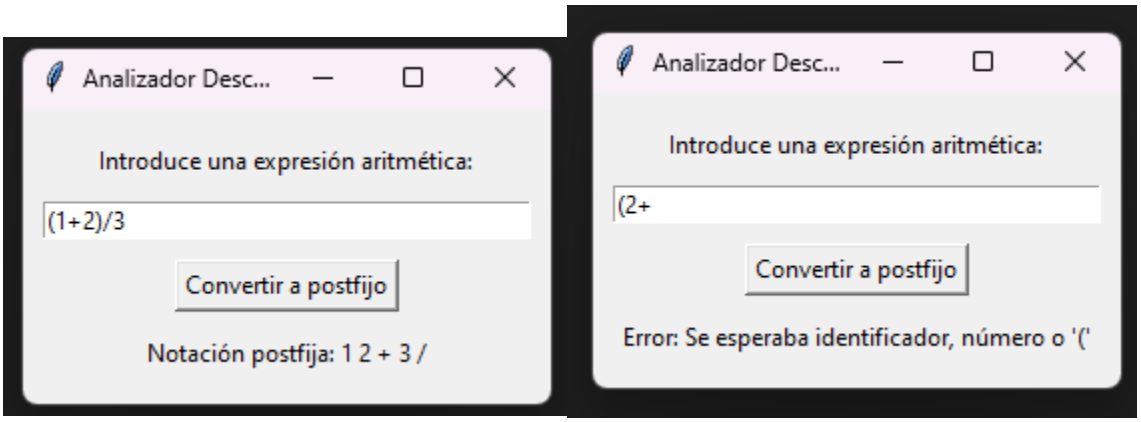
Con **FIRST** y **FOLLOW** calculados para garantizar predictividad.

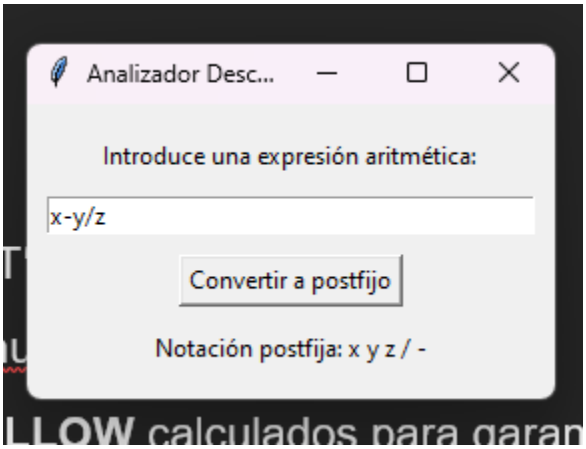
Analizador Léxico (Lexer)

Reconoce:

- NUMBER: \d+ (números multicífras)
- ID: [A-Za-z_]\w*
- Operadores literales (PLUS, MINUS, MULT, DIV) y paréntesis (LPAREN, RPAREN).
- Omite espacios/tabs (SKIP).
- Genera EOF explícito al terminar.

- En caso de coincidencia inesperada lanza Exception("Token inesperado: <char>").





Conclusión:

Se implementó exitosamente un **analizador sintáctico descendente predictivo** en Python, incluyendo un **analizador léxico** que reconoce **identificadores** y **números de más de un dígito**. Se utilizó la gramática transformada de la **página 360** del *Dragon Book*, y se extendió para manejar adecuadamente los operadores de **multiplicación** (*) y **división** (*/*). El programa traduce las expresiones a **notación postfija** o genera mensajes de **error sintáctico** claros, cumpliendo los objetivos establecidos.