Centro Universitario de Ciencias Exactas e Ingeniería

SEM. de Solución de Problemas de Traductores 2

Practica 2

LUIS FELIPE MUNOZ MENDOZA

Juan Pablo Hernández Orozco

219294285

# Objetivo:

Realizar un analizador sintáctico que reconozca sentencias de asignación u operaciones simples,  y valide su estructura con una gramática como la siguiente (ejemplo):  
<program>  -> <assignment> | <assignment> <program>  
<assignment> -> <identifier> = <expression> ;  
<identifier>   -> [a-zA-Z][a-zA-Z0-9\_]\*  
<expression> -> <term> | <term> + <expression> | <term>

* <expression>  
  <term>       -> <factor> | <factor> \* <term> | <factor> / <term>  
  <factor>     -> <identifier> | <number> | ( <expression> )  
  <number>     -> [0-9]+

Pueden usar arboles sintácticos, tablas de derivación y/o autómatas. Se pueden apoyar con herramientas y librerias como yacc, o bison

El objetivo de esta práctica es ***desarrollar un analizador sintáctico*** que reconozca sentencias de asignación y operaciones aritméticas simples (+, -, \*, /) para validar la estructura gramatical de una entrada. Se busca:

* Implementar un **analizador léxico** que identifique tokens como identificadores, números y operadores.
* Implementar un **analizador sintáctico** basado en la gramática propuesta.
* Probar interactivamente la correcta lectura de cadenas y la generación de un árbol sintáctico.

# Introducción:

En la teoría de lenguajes formales y compiladores, un analizador sintáctico (parser) se encarga de verificar que las secuencias de tokens (generadas por un analizador léxico) cumplan con las reglas gramaticales de un lenguaje. Para ello, se define una gramática libre de contexto que describe la estructura válida de las expresiones y sentencias.

En esta práctica, se utiliza la librería PLY (Python Lex-Yacc) para construir tanto el scanner (léxico) como el parser (sintáctico). Se ejemplifica con sentencias de asignación y expresiones aritméticas simples, como x = 3 + 5; o y = (x - 2) \* 4;.

# Desarrollo:

## Código

# Conclusión:

Este analizador sintáctico demuestra cómo, a partir de un conjunto de tokens y reglas gramaticales, se puede verificar la validez de sentencias de asignación y expresiones aritméticas. El árbol sintáctico obtenido facilita posteriores fases de un compilador o intérprete, como la generación de código intermedio o la evaluación directa de la expresión.

# Repositorio

<https://github.com/ELJuanP/Seminario_de_Traductores_de_lenguajes_ll/tree/main/Practica%201>