UNIVERSIDAD DE GUADALAJARA

Centro universitario de ciencias exactas e ingenierías

Avances en la Construcción del Traductor

SEMINARIO DE SOLUCION DE PROBLEMAS DE TRADUCTORES DE LENGUAJES II

Alumno: Carlos Eduardo Borja Soltero

Código: 218292319

Sección: D02

Fecha: 24/Marzo/25

Introducción.....1
Resumen.....1
Código.....2
1. Explicación de la práctica.....5
2.- Entrenamiento del Modelo.....6
3.- Procesamiento de los datos.....6
4.- Definición de las múltiples capas.....7
5. Código para probar el modelo ya entrenado y que reconozca las imágenes.....8
7. Resultados de la Ejecución del Programa.....8
7. Conclusión y resumen de los resultados.....9
8. Referencias Bibliográficas.....10

Introducción.

Desde una perspectiva teórica, el análisis léxico y sintáctico constituyen las dos primeras fases esenciales en la traducción de un lenguaje de programación. Durante el análisis léxico, el código fuente se fracciona en componentes elementales llamados “tokens” mediante reglas basadas en expresiones regulares, lo cual facilita la identificación de palabras reservadas, operadores y literales. A continuación, el análisis sintáctico valida la organización secuencial de estos tokens, contrastándolos con una gramática formal para asegurar la coherencia de la estructura del programa.

Resumen

Este documento expone la conceptualización de un analizador léxico y sintáctico, siguiendo un modelo recursivo descendente para verificar la conformidad del código con una serie de reglas gramaticales predefinidas. Se describe cómo la fase léxica segmenta cadenas en tokens y cómo el parser interpreta dichas secuencias para detectar posibles violaciones a la sintaxis. El enfoque teórico realza la importancia de estas etapas como base para la detección temprana de errores y la posterior generación de código o ejecución interpretada.

1. Explicación del Traductor

Este es el primer avance del traductor, en el cual mediante el analizador_lexico y analizador_sintactico, usamos tokens para validar lineas de código. El funcionamiento es el siguiente:

El archivo tokens_def.py define la clase Token y las reglas de expresión regular para reconocer cada tipo de token. Luego, analizador_lexico.py usa esas definiciones para leer el código y producir una lista de tokens. A continuación, analizador_sintactico.py toma esos tokens y, basándose en una gramática, verifica la estructura del programa. Por último, main.py integra ambos analizadores para realizar todo el proceso de traducción.

- Se dividió el proyecto en cuatro archivos para mantener el código organizado:
    - `tokens_def.py`: define la clase Token y las expresiones regulares.
    - `analizador_lexico.py`: implementa la clase Lexer que convierte el código fuente en tokens.
    - `analizador_sintactico.py`: contiene la clase Parser con la gramática en forma de métodos recursivos.
    - `main.py`: orquesta ambos analizadores y ejecuta el proceso completo con un ejemplo de prueba.

Decisiones Importantes

- **Expresiones Regulares**: Se usó una lista de tuplas (tipoToken, regex) para mapear cada tipo de token a una regla clara, generando una sola expresión regular final.
- **Lexer con Ignorados**: Se optó por ignorar espacios, tabulaciones y comentarios, dado que no aportan significado en la fase léxica. También se decidió incluir saltos de línea en la regla SKIP para no generar errores con líneas vacías.
- **Parser Recursivo Descendente**: Se eligió este enfoque por la sencillez para implementar y entender la gramática. Cada método en `analizador_sintactico.py` corresponde a una regla, lo que mantiene el código legible.
- **Operadores Relacionales**: Para soportar `<`, `>`, `<=`, `>=`, `==`, `!=`, se agregaron reglas específicas (`relational_expr`) que se integran con el resto de expresiones aritméticas.

Manejo de Errores Léxicos

- El lexer lanza una excepción al encontrar un carácter que no encaje en ninguna regla (token `MISMATCH`).
- Se proporciona el número de línea y columna para facilitar la depuración del error.

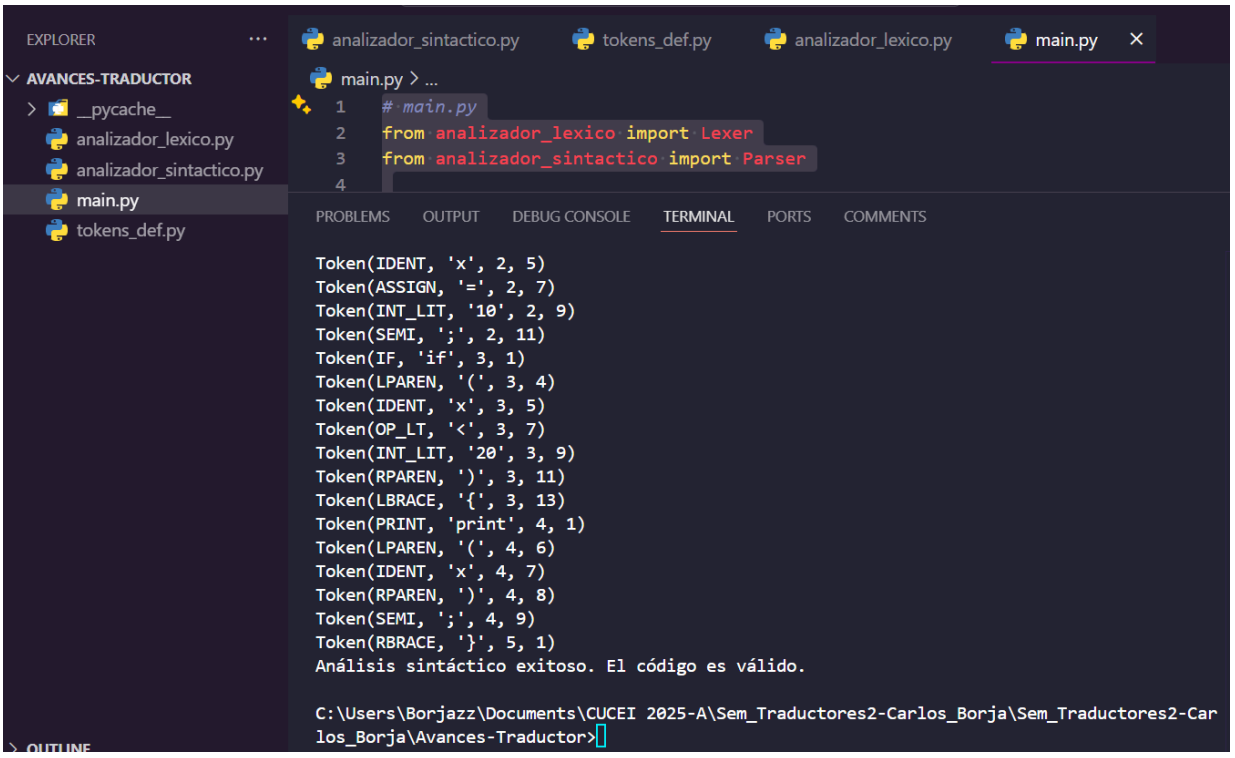
Manejo de Errores Sintácticos

- El parser confirma el tipo de token esperado en cada paso y, si no coincide, arroja una excepción detallada con la posición (línea, columna) y el token problemático.

- Esto detiene el proceso, indicando rápidamente al usuario qué constructo falla en la gramática.

8. Código.

El archivo tokens_def.py define la clase Token y las reglas de expresión regular para reconocer cada tipo de token. Luego, analizador_lexico.py usa esas definiciones para leer el código y producir una lista de tokens. A continuación, analizador_sintactico.py toma esos tokens y, basándose en una gramática, verifica la estructura del programa. Por último, main.py integra ambos analizadores para realizar todo el proceso de traducción.



El resto del codigo se encuentra en:
https://github.com/xBorjazz/Sem_Traductores2-Carlos_Borja/tree/avances-traductor