Diseño del compilador

Yampaul Chaux

Juan Marulanda

Kevin Niño

Faculta de Ingeniería, Corporación Universitaria Remington

Sede, Tuluá

Compiladores

Ing. Jorge Leonardo Gamarra

22 de abril del 2025

Arquitectura General

El compilador está dividido en módulos independientes para cada fase del análisis:

- Análisis léxico (lexer.py): Extrae tokens del código fuente.

- Análisis sintáctico (parser.py): Aplica una gramática para verificar la estructura del programa.

- Análisis semántico (integrado en parser.py): Verifica tipos, declaraciones y errores de uso.

- Interfaz gráfica (gui.py): Permite al usuario ingresar código y ver resultados.

Lexer (Análisis Léxico)

- Implementado con `ply.lex`.

- Identifica: palabras reservadas (`int`, `float`, `print`), identificadores, números, operadores y puntuación.

- Ignora espacios y saltos de línea innecesarios.

Parser (Análisis Sintáctico y Semántico)

- Implementado con `ply.yacc`.

- Define una gramática simple para:

- Declaraciones con asignación.

- Expresiones aritméticas.

- Sentencias `print()`.

Implementa el análisis semántico:

- Tabla de símbolos.

- Validación de tipos.

- Detección de errores de uso.

Interfaz Gráfica

- Construida con `Tkinter`.

Muestra:

- Área de entrada para el código fuente.

- Botón para ejecutar el compilador.

- Panel de salida con tokens y resultados del análisis.

Tabla de Símbolos

Se usa un diccionario Python para registrar:

- Identificadores declarados.

- Tipos asociados (`int`, `float`).

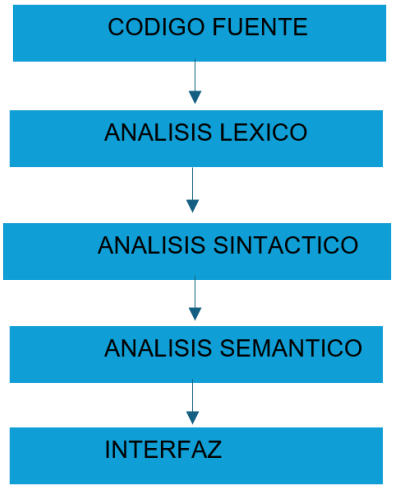
- Permite detectar:

- Reutilización de nombres.

- Uso de variables no declaradas.

- Incompatibilidad de tipos.

Diagrama de Flujo



Lenguaje de Entrada

El lenguaje soportado permite:

- Tipos: `int`, `float`

- Operadores: `+`, `-`, `=`

- Instrucciones: declaraciones, operaciones, `print(variable);`

Manejo de Errores

Tipos de errores reportados:

- Sintácticos: Estructura incorrecta.

- Semánticos: Tipos incompatibles, variables sin declarar, re-declaraciones.

Mensajes claros con íconos visuales (`✓`, `✗`).

Conclusión

Este diseño modular facilita el aprendizaje del proceso de compilación y su visualización, y permite una fácil extensión del compilador para nuevas funcionalidades en el futuro.