Informe Final - Trabajo de Compiladores

Introducción

El objetivo de este trabajo fue extender el compilador desarrollado en los trabajos prácticos, agregando verificación gramatical, generación de código intermedio y optimización básica, siguiendo las consignas de la materia.

Problemática

Se requiere analizar un archivo fuente en C, detectar errores gramaticales y semánticos, generar una tabla de símbolos para todos los contextos, y producir código intermedio en tres direcciones. Además, se debe reportar variables y funciones no usadas, y evitar la generación de código si hay errores.

Desarrollo

- **Gramática:** Se utilizó ANTLR para definir la gramática, soportando bloques, expresiones aritméticas y booleanas, declaraciones, asignaciones, funciones y control de flujo.
- **Parser y Listener:** Se implementó un listener (`Escucha.py`) para construir la tabla de símbolos, detectar errores y advertencias, y generar el reporte de compilación.
- **Código Intermedio:** Se implementó un visitor (`Walker.py`) que recorre el árbol sintáctico y genera código de tres direcciones, siguiendo el formato visto en clase.
- **Optimización:** Se incluyeron ejemplos y sugerencias para optimización de código intermedio (propagación de constantes, eliminación de redundancias).
- **Errores y advertencias:** Se reportan en un archivo separado y, si existen, se detiene la generación de código intermedio.

Resultados

El compilador genera los siguientes archivos de salida:
- **Compilacion.txt:** Tabla de símbolos y reporte de uso de variables y funciones.
- **CodigoIntermedio.txt:** Código intermedio en tres direcciones.
- **Errores&Warnings.txt:** Errores y advertencias detectados.

Conclusión

El proyecto cumple con las consignas del trabajo final, permitiendo analizar código fuente, detectar errores, generar código intermedio y reportar información relevante para el usuario. La estructura modular facilita futuras extensiones y mejoras.
