TP2 - Técnicas de Compilación

En este segundo trabajo práctico se extiende el compilador de Cmini para incluir una **Tabla de Símbolos** y la detección de **errores semánticos**. El sistema ahora procesa un archivo fuente y genera tres resultados principales: 1. **Árbol sintáctico** (ANTLR). 2. **Tabla de símbolos** con contextos e historial. 3. **Reportes de errores**: - Léxicos/sintácticos en reports/syntax.txt. - Semánticos en reports/semantic.txt. Adicionalmente, se genera un volcado de la tabla de símbolos en reports/symbols.txt.

Clases principales desarrolladas:
- TablaDeSimbolos, Contexto, ID, Variable, Tipo → manejo de contextos y variables.
- MiListener → recorre el árbol, actualiza la TS y detecta errores semánticos. -
ErrorReporter → administra y guarda reportes de errores.
- BaseErrorListener → captura errores sintácticos.
- App.java → integra todo el flujo de análisis.

Errores semánticos detectados por el compilador: - Doble declaración del mismo identificador.
- Uso de identificador no declarado.
- Uso de identificador sin inicializar.
- Identificador declarado pero no usado.
- Tipos de datos incompatibles.

Paso a paso para ejecutar y probar:

- 1. Compilar y generar fuentes ANTLR: ejecutar `mvn clean generate-sources compile`.
2. Ejecutar el compilador con un archivo de prueba: `mvn exec:java
-Dexec.args="samples/ok_semantic.c"`.
- 3. Revisar la consola: se imprimirá el árbol sintáctico.
- 4. Revisar los reportes generados en la carpeta `reports/`:
- - `syntax.txt`: errores sintácticos (vacío si no hay).
- - `semantic.txt`: errores semánticos detectados.
- - `symbols.txt`: historial de contextos y variables.
- 5. Probar con `samples/bad_semantic.c` para verificar detección de errores semánticos.
6. Probar con `samples/bad_syntax.c` para verificar errores sintácticos en `syntax.txt`.

Conclusión:

El TP2 cumple con la consigna al integrar la tabla de símbolos y la verificación semántica sobre la gramática de Cmini. El compilador ahora puede detectar y reportar tanto errores sintácticos como semánticos, y mantiene un historial de los identificadores en distintos contextos. Esto sienta la base para el examen final, donde se incorporará la generación de código intermedio.