TÉCNICAS DE COMPILACIÓN

TRABAJO PRÁCTICO Nº1

ALUMNO: Johana Testa

Introducción

El objetivo de este Trabajo Practico es aplicar, utilizando ANTRL, los temas sobre Análisis Léxico y Sintáctico desarrollados en clase. El programa por desarrollar tiene como objetivo generar como salida el Árbol Sintáctico (ANTLR) correcto, dado un archivo de entrada en lenguaje C.

Se debe construir un parser que tenga como mínimo la implementación de los siguientes puntos:

- Reconocimiento de un bloque de código, que puede estar en cualquier parte del código fuente, controlando balance de llaves.
- Verificación de:
    - declaraciones y asignaciones
    - operaciones aritmético/lógicas
    - declaración/llamada a función.

- Verificación de las estructuras de control if, for y while.

Ante el primer error léxico o sintáctico el programa deberá terminar.

Desarrollo

Para el desarrollo del programa, se decidió definir los nombres de TOKENS y reglas gramaticales en inglés.

El primer paso para empezar el desarrollo del programa fue definir los TOKENS necesarios. Se decidió que debíamos declarar como TOKENS a:

- Signos de puntuación (la coma, el punto, el punto y coma, las llaves, los corchetes y los paréntesis)
- Signos de operadores aritméticos (+, -, /, *, %)
- Operadores unarios de incremento y decremento (++ y --)
- Operadores relacionales (==, >=, <=, >, <, !)
- Símbolos de asignación (=, +=, *=, /=, %=)
- Operadores lógicos (and: &&, or: ||, not: !)
- Valores (números enteros, números flotantes, char)
- IDs
- Palabras reservadas del lenguaje (ej: for, while, if, else, true, false, etc).

Un problema que se encontró fue que el analizador gramatical tomaba estructuras de selección (*if*) como funciones ya que identificaba la regla *ID '(' argumentos ')'* en vez de la regla gramatical de if. Luego de un análisis, se encontró que esto se debía al orden en el que estaban definidos los TOKENS, siendo los primeros definidos más generales que los subsiguientes.

Para realizar las pruebas correspondientes con cada regla gramatical, cree archivos *.txt* conteniendo casos de prueba con ejemplos de la estructura a controlar y luego correr con la regla correspondiente. Al finalizar el control de todas las reglas individualmente, hice un caso de prueba conteniendo todos los otros casos testeados anteriormente, y con la estructura de un programa C (con *int main()*, declaración de funciones y variables, inicialización de variables, etc) y probar ese caso de prueba con la regla inicial *prog* que engloba al resto de reglas.

Conclusión

La gramática de un lenguaje de programación ya que se tiene que tener en cuenta los procesos y procedimientos que el mismo debe cumplir. Además, es muy importante tener en cuenta el orden de los TOKENS ya que esto puede hacer que la interpretación cambie y sea errónea, así como también hay que tener en cuenta la dependencia de las reglas gramaticales para evitar recursiones indeseadas.