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).

Una de las primeras problemáticas que se encontró fue la falta de conocimiento concreto sobre la gramática del lenguaje C, es decir, que tipo de expresiones podían estar presentes dentro de otras expresiones. Para poder solucionar esto, se buscó ayuda online sobre distintas estructuras[1].

Otro 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. En este caso, se había definido el TOKEN ID antes que la palabra reservada *if* por lo que esta palabra era analizada como un ID.

El problema más importante y el que llevo a un cambio de toda la estructura del programa, fue poder tener en cuenta todas las posibilidades de distintos tipos de expresiones dentro de otras expresiones. Con la estructura que se utilizó en un principio, esto se hacía imposible ya que se generaba recursión por izquierda que ANTLR era incapaz de resolver. La primera estructura no tomaba en cuenta el encadenamiento de distintos tipos de expresiones, sino que las reglas gramaticales para estos distintos tipos eran definidas individualmente, y aunque por separado funcionaban correctamente, al tratar de concatenarlas se generaba la recursión por izquierda nombrada.
Para solucionar este problema, se procedió a cambiar la estructura del programa recurriendo a la recursividad y el encadenamiento de las distintas expresiones dentro de la misma regla de cada expresión.

Para realizar las pruebas correspondientes con cada regla gramatical, se procedió a generar archivos *.txt* conteniendo casos de prueba con ejemplos de la estructura a controlar y luego correr en *TestRig* con la regla correspondiente. Al finalizar el control de todas las reglas individualmente, se procedió a generar 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.

[1] https://www.tutorialspoint.com/cprogramming/c_overview.htm

Conclusión

La descripción de la gramática de un lenguaje de programación es un proceso complejo que requiere tener en cuenta todas las posibilidades que el lenguaje debe permitir y todas las restricciones que 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.