UBP UNIVERSIDAD Blas Pascal

Técnicas de Complicación

2023

Trabajo Práctico Nro 2

Profesor: Eschoyez Maximiliano

Alumnos:

- Mora Colodrero Jonathan
- Gigli Constanza

Introducción

El objetivo de este Trabajo Práctico 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.

Consigna

Dado un archivo de entrada en lenguaje C, se debe generar como salida el Árbol Sintáctico (ANTLR) correcto. Para lograr esto 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 el balance de llaves.
- Verificación de:
    - Declaraciones y asignaciones.
    - Operaciones aritmeticológicas.
    - 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

El **primer paso** para empezar el desarrollo del programa fue definir los TOKENS necesarios al principio del código. 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 (==, >=, <=, >, <, !)
- 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.

Otro problema que se encontró fue que el analizador gramatical. Luego de un análisis, se llegó a la conclusión de que se debía al orden de declaración de TOKENS, por lo tanto, se tenía que hacer respetando la regla de generalidad, primero los TOKENS más específicos y luego los más generales. De lo contrario, los más generales “sobrescriben” a los más específicos que se encuentran contenidos en el general.

Además, la estructura con respecto a las reglas gramaticales se realizó ´ de manera encadenada y recursiva para poder tener en cuenta las distintas expresiones posibles. Se definió una expresión general de la cual se desprendieron sub-expresiones.

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.

Github

Jonathan Mora Colodrero: https://github.com/JoniMora/TC

Constanza Gigli: https://github.com/Mariaubp/Tc