Informe Escrito

Técnicas de Compilación — Trabajo Práctico Nro. 1

Maximiliano A. Eschoyez
Fecha de Entrega: Martes 28 de Mayo de 2024

Integrantes

- Godoy Cabrera, Lucas Matias
- Robledo, Francisco Lautaro

---

1. Introducción

En este trabajo práctico se abordó el análisis léxico y sintáctico mediante el uso de ANTLR. El objetivo principal fue desarrollar un programa en Java que, dado un archivo de código fuente en una versión reducida del lenguaje C, generara el árbol sintáctico correcto. Este informe describe la problemática abordada, el desarrollo de la solución propuesta y una conclusión sobre los resultados obtenidos.

2. Problemática

La problemática central consistió en la construcción de un parser capaz de reconocer y verificar diversos componentes del lenguaje C, incluyendo:

- **Bloques de código**: Controlando el balance de llaves.
- **Declaraciones y asignaciones**.
- **Operaciones aritméticas y lógicas**.
- **Declaración y llamada a funciones**.
- **Estructuras de control**: if, for y while.

3. Desarrollo de la Solución

3.1. Análisis Léxico

Para el análisis léxico, se definieron una serie de tokens que representan los elementos básicos del lenguaje C reducido. Entre estos tokens se incluyen paréntesis, llaves, corchetes, operadores aritméticos y lógicos, palabras reservadas, y tipos de datos. A continuación, se muestra un extracto de las definiciones de los tokens:

```
PARIZQ : '(';
PARDER : ')';
fragment DIGITO : [0-9];
WS : [ \t\n\r]+ -> skip;
COMILLA : '"';
COMSIMPLE : '\'';
fragment LETRA : [a-zA-Z];
PUNTOCOMA : ';';
COMA : ',';
LLAVEIZQ : '{';
LLAVEDER : '}';
CORCHETEIZQ : '[';
CORCHETEDER : ']';
```

3.2. Análisis Sintáctico

En el análisis sintáctico, se definió la gramática para el parser que incluye reglas para la estructura del programa y sus componentes. Estas reglas permiten reconocer declaraciones, asignaciones, ciclos, estructuras condicionales, funciones, y más. A continuación se muestra un extracto de las reglas sintácticas:

```
programa : instrucciones EOF;

instrucciones : instruccion*;

instruccion : declaracion
            | asignacion
            | ciclo
            | if
            | for
            | funcion
            | llamadaFuncion
            | comentario
            | retorno;

declaracion : TDATO ID PUNTOCOMA?
            | TDATO asignacion
            | TDATO ID COMA;

asignacion : ID IGUAL expresion
            | ID INCREMENTO PUNTOCOMA?
            | ID DECREMENTO PUNTOCOMA?;
```

3.3. Implementación

La implementación se realizó en Java utilizando ANTLR para la generación del lexer y parser. El proyecto fue gestionado con Maven y el control de versiones se realizó con Git. A continuación, se presenta una visión general de las etapas de implementación:

1. **Definición de la gramática en ANTLR**.
2. **Generación del lexer y parser** utilizando ANTLR.
3. **Desarrollo de un programa en Java** que utilizara el parser generado para analizar el código fuente.
4. **Pruebas y depuración** para asegurar la correcta generación del árbol sintáctico.

4. Conclusión

El desarrollo de este trabajo práctico permitió aplicar los conceptos de análisis léxico y sintáctico utilizando ANTLR. La solución propuesta logró cumplir con los requisitos especificados en la consigna, permitiendo reconocer y verificar bloques de código, declaraciones, asignaciones, operaciones aritméticas y lógicas, funciones y estructuras de control en una versión reducida del lenguaje C. Este ejercicio no solo reforzó los conocimientos teóricos, sino que también proporcionó una valiosa experiencia práctica en el uso de herramientas modernas de compilación.

Enlace al repositorio en GitHub: GitHub Repository

El código fuente y el informe escrito se encuentran disponibles en el repositorio mencionado, cumpliendo con las pautas establecidas para la entrega del trabajo práctico.

Este informe proporciona una descripción clara y concisa del trabajo realizado, explicando tanto la problemática abordada como la solución implementada, permitiendo una fácil comprensión del desarrollo y resultados obtenidos.