TRABAJO PRACTICO 1

Reglas sintácticas

Materia: Técnicas de Compilación

Profesor: Maximiliano Andrés Eschoyez

Integrantes:

- Albarracin, Juan
- Asmuzi Gali

Problema para resolver

El problema consiste en realizar una serie de reglas sintácticas para que nuestro parser tenga la implementación de:

- 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 aritmeticológicas,
    - declaración/llamada a función.
- Verificación de las estructuras de control if, for y while.

Herramientas utilizadas

ANTLR4: Herramienta que se usó para construir parsers.

Java 17: Lenguaje el cual contiene el proyecto.

Maven: Controlador y gestor del proyecto.

Visual Studio Code: editor de código elegido junto al plugin de ANTLR4 y Maven.

Solución Planteada

El programa cuenta con la serie de Tokens declarados al principio, los mismos contienen varios de los tokens que mas se suelen usar en la programación, como el ;, (), etc. Cada uno efectivamente se usa para el mismo propósito que el lenguaje C. Para tener en cuenta en el token COMP, se asignaron distintos tipos de comparaciones COMP : '==' | '>' | '<' | '=>' | '=<' | '||' | '&&' | '!=' ; Se lo realizo de esta forma para resolver el problema de cuando por ejemplo realizamos una comparación en un if, while, etc.

El token TIPO, se uso para que en una función se puedan reconocer distintas formas que puede tener una función TIPO : 'void' | 'char' | 'short' | 'int' | 'long' | 'float' | 'double' | 'signed' | 'unsigned' | 'string' ;

Por último tenemos los tokens que detectaran cada instrucción especifica (if, else, for, while).

Las principales reglas, como la de asignación, declaración de variables, detección de suma, resta, multiplicación y division fueron brindadas por el profesor.

Podemos observar la regla sintáctica instrucción, la misma contiene todos los tipos de sentencias de comparación, la declaración de funciones entre otros, esta regla será llamada para cada vez que se encuentre alguna de estas instrucciones.

Cada regla sintáctica contiene una estructura básica de sus respectivas sentencias para lograr su correcta implementación, por ejemplo, para un if tenemos un nodo if, un paréntesis (, la sentencia de comparación, el paréntesis que cierra), y un bloque de instrucciones a realizar si se cumple la condición:

fi : IF_TOKEN PA comparacion PC instruccion ELSE_TOKEN instruccion ;

Problemas encontrados para plantear la solución

Los principales problemas se presentaron a la hora de realizar el bucle for o la declaración de una función. A las cuales se le tuvieron que agregar un trabajo extra y necesitaron otras reglas sintácticas mas aparte de las usadas previamente.

Cabe aclarar que todas las reglas sintácticas requieren de otras reglas, pero la mayoría son las que nos brindó el profesor previamente.

Por ejemplo, para el bucle for se debió crear una regla para detectar su sentencia final antes de llamar al bloque de código. La misma se llama bucleFor y lo que busca es cubrir todo lo que podría llegar a contener esa sentencia final de for (incrementos, decrementos, otros cálculos más específicos, etc).

Para la función tuvimos problemas para su reconocimiento, ya que no queríamos declararlo específicamente en el bloque de instrucción. Finalmente optamos por esa solución que era la mas aceptable. También se le creo una regla sintáctica llamada parametrosfunc, que básicamente es una regla sintáctica para asignar parámetros. Íbamos a usar la que ya está originalmente en el programa (declaración), pero optamos por esta solución ya que con la regla sintáctica declaración no podíamos optar por crear mas variables y con distintos tipos. Tampoco queríamos editar la declaración original por miedo a editar algo que pueda romper con todas las demás reglas donde se encuentra esta, cosa que ya previamente nos había pasado.

Conclusión

Como opinión podemos decir que ANTLR4 fue una herramienta complicada de trabajar al principio, pero una vez que entiendes sus componentes y como trabaja, resulta muy practico para por lo menos lo que hicimos nosotros (reconocedores de parsers).

Podemos concluir que es una interesante herramienta la cual nos ayuda a entender un poco mas como funcionan los compiladores en un lenguaje de programación y la forma en la que se estructuran, crean y realizan las acciones para lo que fue creado.

Links importantes

Repositorio de GitHub donde se alojó el proyecto ; tener en cuenta que el trabajo practico se encuentra en Branch TP1-TC.

Notas realizadas por nosotros a lo largo del desarrollo, para almacenar o documentar los problemas y soluciones planteadas

Link directo al archivo .g4 en GitHub

Documentación de ANTLR4

Página oficial de ANTLR4

Como ultima información, recomendamos utilizar alguna distribución Linux para realizar este ejercicio, ya que en Windows debemos cargar previamente los Virtual Enviroments de Java y Maven para que pueda funcionar en VS code.

Por último también recomendamos usar Java 17, no por algo en especial, sino que esa versión usamos nosotros y no tuvimos problemas al trabajar.