Compilador de C

Universidad de la Defensa Nacional

Centro Regional Universitario Córdoba IUA

Facultad de Ingeniería

Carrera: Ingeniería en Informática

Asignatura: Desarrollo de Herramientas de Software

Profesor: Ing. Maximiliano Eschoyez

Año: 2022

Equipo de Trabajo

Camargo Tenaglia, Gaston Matias

García Bidart, Aurelio

Índice

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 aritméticas lógicas
- declaración / llamada a función
- estructuras de control (if, for y while)

Además, al finalizar el reconocimiento de la entrada se deberá mostrar mediante un archivo de texto plano contenido de la Tabla de Símbolos para cada contexto.
Desarrollo

Marco teórico

ANTLR

ANTLR es una herramienta creada principalmente por Terence Parr, que opera sobre lenguajes, proporcionando un marco para construir reconocedores (parsers), intérpretes, compiladores y traductores de lenguajes a partir de las descripciones gramaticales de los mismos (conteniendo acciones semánticas a realizarse en varios lenguajes de programación).

Gramática

La gramática es el estudio de las reglas y principios que gobiernan el uso de las lenguas y la organización de las palabras dentro de unas oraciones y otro tipo de constituyentes sintácticos. También se denomina así al conjunto de reglas y principios que gobiernan el uso de una lengua concreta; así, cada lengua tiene su propia gramática.

Analizador léxico

Un analizador léxico o analizador lexicográfico es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa (secuencia de caracteres) y produce una salida compuesta de tokens (componentes léxicos) o símbolos. Estos tokens sirven para una posterior etapa del proceso de traducción, siendo la entrada para el analizador sintáctico (en inglés parser).
La especificación de un lenguaje de programación a menudo incluye un conjunto de reglas que definen el léxico. Estas reglas consisten comúnmente en expresiones regulares que indican el conjunto de posibles secuencias de caracteres que definen un token o lexema. En algunos lenguajes de programación es necesario establecer patrones para caracteres especiales (como el espacio en blanco) que la gramática pueda reconocer sin que constituya un token en sí.

Analizador sintáctico

El Analizador sintáctico analiza una cadena de símbolos según las reglas de una gramática formal. Convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada.

Analizador semántico

El analizador semántico utiliza el árbol sintáctico y la información en la tabla de símbolos para comprobar la consistencia semántica del programa fuente con la definición del lenguaje. También recopila información sobre el tipo y la guarda, ya sea en el árbol sintáctico o en la tabla de símbolos, para usarla más tarde durante la generación de código intermedio.

Procedimiento

Para poder desarrollar la consigna se declaran las reglas gramaticales, se implementa un analizador léxico, un analizador sintáctico y un analizador semántico.
Para las reglas gramaticales se decide utilizar etiquetas intuitivas para la fácil comprensión.
Para el analizador léxico se crea el "Listener" correspondiente, en este caso *MyListener*, encargado de cargar la *Tabla de Símbolos* a través de los eventos generados.
La tabla de símbolos es una clase que contiene una lista de diccionarios en la cual se cargan y eliminan los contextos a medida que se va leyendo el código C.En cada uno de estos contextos se almacenan las variables correspondientes, las cuales serán seguidas a lo largo de la lectura para verificar que hayan sido inicializadas y utilizadas.

Resultado

Al finalizar el desarrollo del proyecto, se pudieron observar que los objetivos planteados se cumplieron y también se logró hacer un mayor entendimiento de cómo es que trabajan los compiladores que utilizamos en el dia a dia. Cuando programamos hay muchas cosas que damos por hecho, pasamos por alto o simplemente las entendemos como "las cosas son así porque sí", pero en realidad detrás del código hay analizadores que realizan distintas tareas, de formas lógicas que uno al programar no se plantea. Pero luego de hacer este trabajo práctico, luego de incorporar el conocimiento sobre los compiladores podemos entender que hay reglas que se cumplen por simple metodología o costumbre, y otras que hasta se puede aprovechar su potencial para lograr hacer cosas más óptimas, simples y efectivas.

Repositorio

https://github.com/sharkymid/ANTLR4cCompiler