PROYECTO FINAL

TÉCNICAS DE COMPILACIÓN

Profesor: Maximiliano Eschoyez

Alumnos: Bautista Madruga / Tomas Alvarez

Introducción:

El objetivo de este trabajo final es poder desarrollar los diferentes elementos que conforman un compilador utilizando la herramienta ANTLR4. El mismo esta conformado por 3 elementos bases los cuales serán explicados en un futuro, estos son: Listener, Visitor, Reglas.

Conceptos:

Reglas:

Las reglas son un conjunto de expresiones regulares las cuales permiten buscar patrones de texto en nuestro código de entrada.

Listener:

El Listener en ANTLR4 es un mecanismo que permite reaccionar a eventos mientras se recorre el árbol de análisis sintáctico. Se usa comúnmente para realizar verificaciones semánticas, como la validación de variables y funciones utilizando una Tabla de Símbolos (creada manualmente dentro del Listener). Recorre el árbol en **preorden**, es decir, primero visita el nodo padre (`enterRule`), luego sus hijos y finalmente ejecuta (`exitRule`) al salir del nodo.

Visitor:

El Visitor en ANTLR4 se encarga de recorrer el árbol sintáctico en **postorden**, procesando primero los hijos y luego el nodo actual. Se puede utilizar para generar **código intermedio (TAC)**, recorriendo los nodos del árbol y generando instrucciones a medida que se resuelven expresiones o declaraciones. Cada nodo del árbol puede producir una o más instrucciones de TAC.

Implementación:

Para el desarrollo del trabajo final implementamos las siguientes clases para poder cumplir con la consigna estipulada:

ID.java:

Esta clase abstracta se encarga de guardar la siguiente información de una variable/función:

- Indicar el tipo de dato.
- Indicar el nombre del dato.
- Indicar si la variable fue usada.
- Indicar si fue instanciada.
- Indicar si es una función.

Variable.java:

Esta clase hereda de ID la cual se encarga de encapsular la información y mostrar los valores y tipos de datos mediante una función toString.

Contexto.java:

Esta clase se encarga de contener el listado de variables dentro de un contexto determinado, esto nos permite poder declarar e identificar variables con las mismas propiedades en los diferentes contextos, y que cada contexto pueda manipularla sin errores sintácticos y semánticos.

TablaDeSimbolos.java:

Esta clase contiene la lógica de la Tabla de Símbolos, la cual se encargará de guardar las diferentes variables/funciones del código. Esta misma contiene dos listas diferentes:

- Una de estas nos permite almacenar las variables por Contexto, pudiendo indicar lo mencionado anteriormente en la clase *ID.java*, guardando el tipo de dato y la variable en la Tabla, además de realizar validaciones adicionales dentro del contexto al momento de utilizar una variable, que aseguran que la validación semántica y sintáctica sea correcta.
- La segunda guarda el histórico de todas las variables/funciones declaradas en el código.
- Se encarga de contar la cantidad de contextos totales empezando desde el valor 0(base).

TACLevel.java:

Utilizamos dicha clase para poder ordenar sintácticamente las operaciones aritméticas lógicas utilizando un listado de factores y símbolos.

TACHelper.java:

Nos permite escribir y generar el archivo de salida del código TAC, indicando las variables temporales y los labels correspondientes.

Conclusión:

En el desarrollo de este compilador nos permitió entender de mejor manera las diferentes etapas en el momento de compilación de un código, permitiéndo nos ver el análisis, validación, procesado y generación del código fuente.

Repositorio:

Github (https://github.com/TomasAlvarez78/TCProyecto2022)