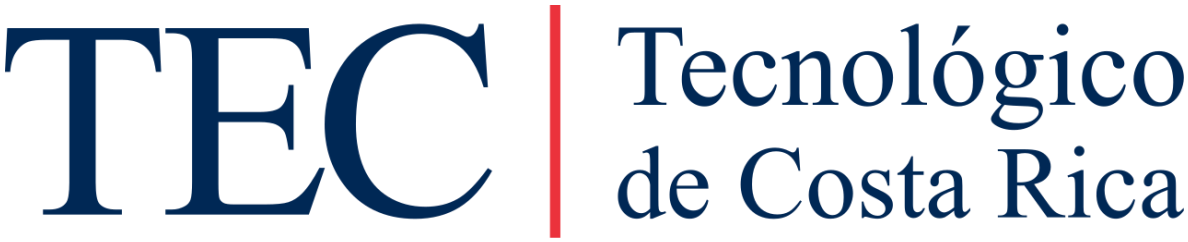

INSTITUTO TECNOLÓGICO DE COSTA RICA

UNIDAD DE INGENIERÍA EN COMPUTACIÓN

COMPILADORES E INTÉRPRETES IC-5701

I Tarea Programada

Análisis Sintáctico Mini Go

Profesor

Oscar Mario Víquez Acuña

Autores

Juan Bautista Núñez Parrales - 2021106189

Jennifer Gonzáles Solís – 2022078253

Sede San Carlos

Primer semestre

3 abril de 2024

Índice

Contenido

Introducción

Este proyecto nos sumerge en la implementación de un compilador para un subconjunto del lenguaje Go, denominado "Mini GO". A través del análisis sintáctico, utilizando herramientas como ANTLR4 y Go, buscamos comprender la estructura y procesamiento del código fuente, desde la identificación de tokens hasta la construcción de un Árbol de Sintaxis Abstracta (AST).

A lo largo del documento, abordaremos los requerimientos del proyecto, desde la implementación del scanner hasta la creación de un editor de texto para visualizar y analizar pruebas sintácticas. La evaluación se centrará en la detección de errores en el código de prueba, que jugará un papel fundamental en la evaluación del trabajo.

Este proyecto representa una oportunidad para explorar la compilación de lenguajes de programación, fortaleciendo nuestras habilidades técnicas y analíticas mientras navegamos por las complejidades de Mini GO.

Análisis del Lenguaje

La gramática definida en el archivo `goParser.g4` define las reglas sintácticas del lenguaje Mini GO, un subconjunto simplificado del lenguaje de programación Go. Este es un análisis detallado de la gramática y las capacidades del lenguaje Mini GO:

1. Declaraciones de variables y tipos:
   - Mini GO admite la declaración de variables utilizando la palabra clave `var`, seguida de uno o más identificadores y, opcionalmente, su tipo y valor inicial.
   - También permite la declaración de tipos personalizados mediante la palabra clave `type`, seguida de un identificador y un tipo base.
   - Las estructuras de datos, como los slices y los arrays, así como las estructuras de tipo `struct`, también están soportadas en Mini GO.
2. Declaraciones y llamadas de funciones:
   - La definición de funciones se realiza con la palabra clave `func`, seguida de un identificador y los parámetros de la función, seguidos opcionalmente por su tipo de retorno.
   - Las funciones pueden tener parámetros con nombres y tipos específicos, y también pueden retornar múltiples valores.
3. Expresiones y operadores:
   - Mini GO soporta una amplia variedad de expresiones y operadores, incluyendo operadores aritméticos (`+`, `-`, `*`, `/`, `%`), operadores de comparación (`==`, `!=`, `<`, `>`, `<=`, `>=`), operadores lógicos (`&&`, `||`, `!`), entre otros.
   - Las expresiones pueden ser construidas mediante la combinación de operadores y operandos, como identificadores, literales numéricos y cadenas, y llamadas a funciones.
4. Estructuras de control de flujo:
   - Mini GO soporta estructuras de control de flujo como `if`, `for`, `switch`, `break` y `continue`, que permiten controlar la ejecución del programa en base a condiciones lógicas y bucles.
5. Impresión y manejo de errores:
   - El lenguaje Mini GO incluye funciones predefinidas para la impresión de datos en consola, como `print` y `println`.

- Además, Mini GO permite el retorno de valores desde las funciones, lo que facilita el manejo de errores y la lógica de control del programa.

Comparación con GoLang:

La gramática de Mini GO presenta similitudes significativas con GoLang, lo que facilita la comprensión para aquellos familiarizados con el lenguaje original. Ambos lenguajes comparten la sintaxis básica de declaración de variables, funciones, estructuras de control de flujo y expresiones. Sin embargo, Mini GO omite algunas características avanzadas de GoLang, como los punteros, las interfaces y las goroutines, con el fin de simplificar la implementación del compilador y el entendimiento del lenguaje por parte de los estudiantes.

Soluciones e Implementaciones

La implementación del proyecto se basa en la utilización de ANTLR4 para generar el analizador léxico y sintáctico a partir de las gramáticas definidas en los archivos `goParser.g4` y `goScanner.g4`. Algunas de las soluciones y decisiones técnicas tomadas durante la implementación del proyecto:

1. Uso de ANTLR4:

ANTLR4 es una herramienta ampliamente utilizada para la generación de analizadores léxicos y sintácticos a partir de gramáticas formales. Se eligió ANTLR4 por su capacidad para generar código en múltiples lenguajes de programación, incluyendo Go, y su facilidad de uso.

2. Separación de gramáticas:

Se optó por dividir la gramática del lenguaje Mini GO en dos archivos separados: `goParser.g4` y `goScanner.g4`. Esto permite una mejor organización y mantenimiento del código, al separar las reglas sintácticas de las reglas léxicas.

3. Integración con Go:

El código generado por ANTLR4 se integró directamente en el proyecto Go, lo que facilita la interacción entre el analizador sintáctico y el resto de la aplicación.

Se utilizó el paquete `github.com/antlr4-go/antlr/v4` para interactuar con los componentes generados por ANTLR4 desde el código Go.

4. Interfaz gráfica con Fyne:

Se implementó una interfaz gráfica de usuario utilizando la biblioteca Fyne, que proporciona herramientas para la creación de aplicaciones de escritorio en Go.

La interfaz de usuario permite al usuario escribir código Mini GO en un área de texto y compilarlo con un simple clic en un botón.

5. Manejo de errores:

Se implementó un mecanismo para capturar y mostrar errores de compilación en una ventana separada. Para ello, se utilizó un `ErrorListener` personalizado que se adjuntó al analizador sintáctico generado por ANTLR4.

Los errores se presentan de manera clara y legible, proporcionando información sobre la línea y columna donde ocurrió el error y una descripción del problema.

6. Pruebas y depuración:

Se realizaron pruebas exhaustivas utilizando archivos de prueba que cubren todas las capacidades sintácticas del lenguaje Mini GO. Esto permitió identificar y corregir errores en la implementación del analizador sintáctico.

Se utilizó el entorno de desarrollo integrado de Go (IDE) para depurar y solucionar problemas de manera efectiva.

Resultados Obtenidos

Requerimiento	Estado	Observaciones
Reconocer por medio del scanner todos los tokens válidos para Mini GO con sus respectivos formatos según el lenguaje original GoLang.	Completo 100%	
Permitirse el manejo de comentarios en el código fuente siguiendo el mismo formato.	Completo 100%	
Escribir archivos de prueba que ejemplifiquen todas las capacidades sintácticas del lenguaje.	Completo 100%	
Reconocer si dichas pruebas son sintácticamente correctas a partir de la implementación del parser para Mini Go.	Completo 100%	
Implementar un Editor de texto que permita mostrar y parsear las pruebas elaboradas.	Completo 100%	
El editor debe mostrar además los posibles errores sintácticos, el lugar en donde estos eventualmente aparezcan y debe permitir identificar fácilmente por línea y columna estos errores en el editor.	Completo 100%	

Conclusiones

Con este proyecto, hemos logrado desarrollar un programa capaz de entender y analizar el lenguaje de programación Mini GO, lo que nos ha permitido adentrarnos en el fascinante mundo de la compilación. A través de este proceso, hemos aprendido a transformar el código escrito por humanos en instrucciones comprensibles para una computadora, lo que nos ha proporcionado una comprensión más profunda de la esencia de la programación.

Utilizando herramientas especializadas como ANTLR4 y Go, hemos construido con éxito el analizador de Mini GO, enfrentando y superando diversos desafíos técnicos en el camino. Al explorar las capacidades del lenguaje Mini GO, hemos descubierto sus similitudes con Go, pero también reconocimos su simplicidad relativa, lo que facilitó enormemente el proceso de desarrollo del compilador.

El programa resultante es capaz de leer y analizar el código escrito en Mini GO, identificando y reportando cualquier error sintáctico que pueda contener. Además, hemos construido una interfaz de usuario intuitiva que simplifica el proceso de compilación para el usuario, brindando una experiencia más fluida y accesible.

A lo largo de este proyecto, hemos enfrentado desafíos significativos y hemos experimentado un profundo aprendizaje sobre el funcionamiento de la compilación, el uso de herramientas poderosas para construir software y la resolución de problemas en el desarrollo de aplicaciones. Esta experiencia ha sido invaluable, permitiéndonos mejorar nuestras habilidades técnicas y ampliar nuestra comprensión de cómo funcionan los programas que utilizamos en nuestra vida diaria.