Ingeniería en Computación

Lenguajes de Programación

Mini-Go Compiler

Profesor:

Oscar Mario Víquez Acuña

Grupo 50, 51

Estudiantes:

Kevin Vinicio Varela Rojas

Anthony Andrés Jiménez Zamora

Fabián Rojas Ugalde

Sede San Carlos

03/04/2024

Análisis del Lenguaje:

El análisis del lenguaje definido por la gramática proporcionada revela un enfoque conciso y funcional, diseñado específicamente para facilitar el desarrollo de un compilador en Go. Inspirado en el lenguaje original Go, esta versión simplificada sigue de cerca sus estructuras y sintaxis fundamentales, permitiendo la definición de paquetes, variables, tipos y funciones.

Una de las características destacadas es la capacidad de definir tipos personalizados mediante la palabra clave `type`. Esto refleja la flexibilidad del lenguaje original Go para permitir al programador crear abstracciones y estructuras de datos complejas. Esta capacidad es crucial en el desarrollo de compiladores, donde la manipulación y representación de datos puede ser variada y sofisticada.

El lenguaje de compilador también presenta una manera simplificada de manejar declaraciones y asignaciones, lo que puede resultar en un código más compacto y fácil de entender para el compilador mismo. Sin embargo, esta simplicidad puede venir a costa de la expresividad y la capacidad de manejar casos más complejos presentes en el lenguaje Go original.

La inclusión de estructuras de control como condicionales, bucles y sentencias de switch demuestra la intención de proporcionar al desarrollador las herramientas necesarias para escribir programas estructurados y funcionales. Esto refleja el enfoque del lenguaje Go en la legibilidad y la eficiencia, incluso en un contexto de desarrollo de compiladores.

Además, la gramática ofrece operaciones aritméticas y lógicas sobre expresiones, lo que permite al compilador realizar cálculos y evaluaciones complejas. Las funciones integradas como `len`, `cap` y `append` agregan funcionalidades útiles para manipular estructuras de datos dinámicas, como slices y arrays.

Además de la implementación de las reglas gramaticales y la lógica del compilador, es crucial abordar la gestión de errores de manera efectiva. Esto implica desarrollar un código que identifique y maneje adecuadamente una amplia gama de

posibles errores léxicos, sintácticos y semánticos que pueden surgir durante el proceso de compilación. La implementación de una sólida infraestructura de manejo de errores es fundamental para la usabilidad y la confiabilidad del compilador, contribuyendo así a una experiencia de desarrollo más fluida y productiva para los usuarios.

Aunque simplifica algunas características del lenguaje original Go, el lenguaje definido por esta gramática mantiene su esencia y funcionalidades principales. Esto lo convierte en una herramienta poderosa para el desarrollo de compiladores en Go, al tiempo que proporciona una base sólida para la comprensión y manipulación del código fuente en el contexto de la construcción de herramientas de compilación.

Soluciones e implementación:

Para la implementación del proyecto, se adoptó una estrategia integral que abordó tanto la creación de una interfaz web para la introducción del código a compilar como el desarrollo de un parser y un scanner que satisfacen todas las necesidades de la gramática proporcionada.

En primer lugar, se creó un servidor en Go para alojar la interfaz web. Esta interfaz permite a los usuarios ingresar el código fuente que desean compilar utilizando el lenguaje definido por la gramática. El servidor actúa como un punto central para recibir las solicitudes de compilación, procesar el código introducido y devolver los resultados al usuario. La utilización de Go para este propósito proporciona una solución eficiente y escalable, aprovechando las capacidades del lenguaje para manejar solicitudes web de manera rápida y confiable.

Por otro lado, se desarrolló un scanner que cumple con los requisitos de la gramática proporcionada. En el scanner, se declararon todos los tokens necesarios del lenguaje, estableciendo así la base para el análisis léxico del código fuente. Este componente es fundamental para identificar y clasificar las diferentes unidades léxicas presentes en el código, como palabras clave, identificadores, literales y operadores.

En cuanto al parser, se implementaron las reglas de la gramática para definir la estructura sintáctica del lenguaje. El parser se encarga de analizar el flujo de tokens generados por el scanner y construir un árbol de sintaxis abstracta que representa la estructura del código fuente. Este árbol sintáctico sirve como base para realizar análisis semántico y generar el código objeto correspondiente.

Además de la creación del servidor web, el parser y el scanner, es fundamental abordar la implementación de un sistema robusto para el manejo de errores. Para lograr esto, se debe diseñar un mecanismo eficiente que permita al compilador identificar la causa y la ubicación precisa de los errores, así como proporcionar mensajes de error claros y descriptivos al usuario. Esto no solo facilitará la depuración y corrección de problemas en el código fuente, sino que también mejorará la experiencia general del usuario al interactuar con el compilador. La implementación

de un sólido sistema de manejo de errores garantizará la confiabilidad y la usabilidad del compilador, contribuyendo así a su efectividad y utilidad en el proceso de desarrollo de software.

La implementación del proyecto combinó el desarrollo de una interfaz web en Go con la creación de un scanner y un parser que cumplen con los requisitos de la gramática proporcionada. Esta aproximación integral permitió construir una herramienta completa y funcional para compilar código en el lenguaje definido, proporcionando una solución robusta y eficiente para el desarrollo de compiladores en Go.

Resultados obtenidos:

Aspecto	Realizado	No Realizado
Interfaz web	Desarrollo completo de la interfaz en Go	-Se realiza al 100% la interfaz web. Se logra ingresar texto y este muestra si hay errores o está correcto.
Scanner	Implementación de un scanner que cumple con la gramática proporcionada	-El scanner funciona al 100% con sus debidos tokens definidos.
Parser	Desarrollo de un parser que sigue las reglas de la gramática para construir la sintaxis	-El parser se logra realizar de forma exitosa. Con su estructura y etiquetas.
Integración	Integración exitosa de la interfaz web con el scanner y parser para compilar código Go	-Se logró de forma exitosa la integración entre la interfaz y el compilador. Funciona al 100%
Manejo de Errores	Se implementa código para administrar los errores del compilador	-Se logra identificar los errores que presenta el test.txt.
Ampliación y Mejora	Potencial para ampliar y mejorar el compilador en el futuro según necesidades y requisitos	-Se puede ampliar y mejorar las condiciones, gramática y las condiciones que se deseen.
Documentación	Creación de documentación del proyecto y del funcionamiento del compilador	-Se logró al 100% la documentación del proyecto. Se explica todo de forma clara y específica.
Pruebas	Realización de pruebas para verificar la correcta funcionalidad del compilador	-Se realizan las pruebas necesarias para comprobar el funcionamiento del scanner y parser.

Conclusiones:

Con base en el análisis de la gramática proporcionada y la descripción de la implementación del proyecto, se pueden extraer varias conclusiones importantes:

1. ***Viabilidad de la implementación en Go:*** La elección de Go como lenguaje de implementación para el compilador resulta ser adecuada dada su capacidad para manejar de manera eficiente tanto la construcción de servidores web como el desarrollo de herramientas de análisis sintáctico y léxico.
2. ***Adaptabilidad de la gramática:*** La gramática proporcionada ofrece un subconjunto funcional del lenguaje Go, lo que permite implementar un compilador capaz de manejar una amplia variedad de programas escritos en Go. Aunque simplifica algunas características, mantiene la esencia y la funcionalidad del lenguaje original.
3. ***Integración de tecnologías:*** La combinación de una interfaz web en Go con un scanner y parser desarrollados en el mismo lenguaje demuestra la capacidad de integrar diferentes tecnologías para crear una solución completa y funcional. Esta integración facilita la creación de un flujo de trabajo fluido y eficiente para los usuarios del compilador.
4. ***Flexibilidad y escalabilidad:*** La arquitectura modular del compilador permite una fácil expansión y modificación en el futuro. Se pueden agregar nuevas características o realizar mejoras en el análisis léxico y sintáctico según sea necesario, lo que garantiza la adaptabilidad del compilador a medida que evolucionan los requisitos del proyecto.
5. ***Potencial de uso y desarrollo futuro:*** El compilador desarrollado tiene el potencial de ser una herramienta útil para estudiantes, desarrolladores y entusiastas de la programación en Go. Además, puede servir como base para proyectos más avanzados en el campo de la compilación y el análisis de lenguajes de programación.

El proyecto demuestra la viabilidad y utilidad de desarrollar un compilador en Go utilizando una gramática simplificada del lenguaje original. La implementación

exitosa del compilador y su integración con una interfaz web proporcionan una solución completa y funcional para compilar código en el lenguaje definido, con un amplio potencial para futuros desarrollos y aplicaciones.

Bibliografía:

ChatGPT. (n.d.). *ChatGPT*. Ayuda en minigo. Retrieved marzo, 2024, from https://chat.openai.com/