MiniGo: Compilador y Editor: Parte 2

Aaron González y Daniel Porras

ITCR

Compiladores e Intérpretes

Óscar Víquez

1 de mayo de 2024

MiniGo: Compilador y Editor: Parte 2

Índice

Introducción

Este trabajo presenta un compilador desarrollado con ANTLR-4 para un lenguaje de programación inspirado en Go, pero con una sintaxis simplificada. En esta etapa se desarrolló la fase de análisis sintáctico.

Solución

Para abordar la fase de análisis sintáctico en nuestro compilador, implementamos un typechecker que valida el código fuente en busca de errores de tipos y estructura. Utilizamos una tabla de símbolos para gestionar las variables declaradas, sus tipos y alcances, lo que facilitó las verificaciones de alcance, existencia y consistencia de tipos.

Implementación

El typechecker es un componente esencial que verifica las operaciones y asignaciones del programa. Implementado como un visitor, recorre el AST para analizar cada nodo en busca de operaciones y declaraciones que requieran verificación de tipos. Durante este proceso, mantenemos la tabla de símbolos actualizada con información sobre las variables declaradas y sus tipos, lo que nos permite realizar verificaciones de alcance y consistencia de tipos.

Durante el recorrido del AST, verificamos que todas las operaciones y asignaciones sean consistentes con los tipos de datos esperados. Si encontramos una operación o asignación con tipos incompatibles, generamos un mensaje de error indicando la ubicación del error y una descripción del problema. También verificamos que los tipos de las variables coincidan con los tipos esperados en cada contexto.

Además, utilizamos la tabla de símbolos para verificar que las variables se utilicen dentro de su alcance adecuado. Si encontramos una referencia a una variable fuera de su alcance, generamos un error.

Cuadro comparativo

Descripción	Estado
Redeclaración de identificadores en el mismo nivel.	Completo
Declaración de variables y usos de dichas variables con su tipo según el nivel de su alcance.	Completo
Declaración y uso de funciones, incluyendo parámetros en cantidad y tipo.	Completo
Checkeos de tipo de retornos.	Completo
Uso de arreglos y slices con indice para variables declaradas como arreglos o slices.	Completo
Declaración y uso de tipos `struct`.	Completo
Uso de métodos predefinidos.	Completo
Uso de identificadores definidos como tipos.	Completo
Chequeo de expresiones con todos los tipos de operaciones posibles.	Completo

Conclusiones

La implementación del typechecker en nuestro compilador ha sido fundamental para garantizar la corrección y robustez del análisis sintáctico de los programas. Esta etapa ha permitido detectar y corregir errores de tipos en el código fuente, evitando posibles fallos durante la ejecución.

Gracias a la tabla de símbolos utilizada para gestionar las variables declaradas, hemos podido verificar que las variables se utilicen dentro de su alcance adecuado, evitando errores de referencia a variables inexistentes o fuera de alcance.

La implementación del typechecker ha fomentado una escritura más cuidadosa y precisa por parte de los programadores, ya que los errores de tipos y alcance se detectan de manera temprana en el proceso de desarrollo. Además, la estructura modular del typechecker facilita la incorporación de nuevas reglas de verificación y la corrección de errores, lo que hace que el compilador sea más fácil de mantener y actualizar en el futuro.