MiniGo Compiler & Editor

Aaron González y Daniel Porras

ITCR

Compiladores e Intérpretes

Óscar Víquiez

2 de abril de 2024

MiniGo Compiler & Editor

Índice

Introducción

Este trabajo presenta un compilador desarrollado con ANTLR-4 para un lenguaje de programación inspirado en Go, pero con una sintaxis simplificada. Se describirá la gramática del lenguaje, la implementación del compilador y su funcionamiento.

Desarrollo

El desarrollo del compilador se basa en el uso de ANTLR-4, una herramienta ampliamente utilizada para generar analizadores léxicos y sintácticos a partir de gramáticas

formales. En este proyecto, se ha definido una gramática específica para el lenguaje simplificado inspirado en Go. Esta gramática describe la estructura sintáctica del lenguaje, incluyendo reglas para declaraciones, expresiones, estructuras de control y otros elementos fundamentales.

Una vez definida la gramática en ANTLR-4, se utiliza esta herramienta para generar el código fuente de un analizador léxico y sintáctico. Este código, escrito en Go, constituye el núcleo del compilador. El programa resultante recibe como entrada el nombre de un archivo que contiene el código fuente en el lenguaje simplificado. El analizador léxico y sintáctico se encarga de leer este archivo, tokenizar el código y verificar su estructura sintáctica según la gramática definida.

Durante el proceso de análisis, el compilador identifica posibles errores sintácticos, como declaraciones mal formadas o estructuras de control incorrectas. Si se encuentra algún error, el compilador muestra un mensaje indicando la naturaleza del problema y la ubicación aproximada en el código fuente. Por otro lado, si no se encuentran errores, el compilador devuelve un código de éxito, indicando que el código fuente es válido según la gramática definida.

Además del compilador en Go generado por ANTLR-4, se ha desarrollado un editor de código con una interfaz gráfica amigable. Este editor es responsable de ejecutar el compilador y mostrar la salida al usuario.

Cuando el usuario abre un archivo en el editor, este envía el código al compilador para su análisis. El compilador procesa el código y, si encuentra errores, devuelve mensajes detallados sobre la naturaleza de los problemas encontrados y su ubicación en el código fuente.

El editor recibe esta salida del compilador y la muestra de manera clara y organizada al usuario, resaltando los errores y facilitando su comprensión.

Análisis de la gramática

Root

La regla root define la estructura básica del programa, que consiste en un paquete seguido de un identificador y una lista de declaraciones principales (topDeclarationList).

Declaraciones principales (topDeclarationList)

Esta regla permite la presencia de múltiples declaraciones de variables, tipos y funciones en cualquier orden.

Declaración de variables (variableDecl)

Esta regla define la sintaxis para la declaración de variables. Puede ser una declaración de una sola variable con tipo y valor (VAR singleVarDecl SEMICOLON), una declaración de múltiples variables dentro de paréntesis (VAR LEFTPARENTHESIS innerVarDecls RIGHTPARENTHESIS SEMICOLON) o una declaración de variables vacía (VAR LEFTPARENTHESIS RIGHTPARENTHESIS SEMICOLON).

Declaración de tipos (typeDecl)

Esta regla define la sintaxis para la declaración de tipos. Puede ser una declaración de un solo tipo (TYPE singleTypeDecl SEMICOLON), una declaración de múltiples tipos dentro de paréntesis (TYPE LEFTPARENTHESIS innerTypeDecls RIGHTPARENTHESIS SEMICOLON) o una declaración de tipos vacía (TYPE LEFTPARENTHESIS RIGHTPARENTHESIS SEMICOLON).

Declaración de funciones (funcDecl)

Esta regla define la sintaxis para la declaración de funciones. Consiste en una declaración frontal de función (funcFrontDecl) seguida de un bloque de código (block) y un punto y coma.

Declaración frontal de función (funcFrontDecl)

Esta regla define la sintaxis para la declaración frontal de una función, que incluye el nombre de la función, los argumentos entre paréntesis (funcArgsDecls) y el tipo de retorno

opcional (declType).

Argumentos de función (funcArgsDecls)

Esta regla define la sintaxis para los argumentos de una función, que consiste en una lista de declaraciones de variables sin expresiones (singleVarDeclNoExps) separadas por comas.

Tipos de declaración (declType)

Esta regla define la sintaxis para los tipos de declaración, que puede ser un tipo entre paréntesis (LEFTPARENTHESIS declType RIGHTPARENTHESIS), un identificador (IDENTIFIER), un tipo de slice (sliceDeclType), un tipo de array (arrayDeclType) o un tipo de estructura (structDeclType).

Expresiones (expression)

Esta regla define la sintaxis para las expresiones en el lenguaje. Puede ser una expresión primaria (primaryExpression), una expresión binaria (expression op expression), una expresión unaria (op expression), o una llamada a función (identifierList arguments).

Expresiones primarias (primaryExpression)

Esta regla define la sintaxis para las expresiones primarias, que pueden ser un operando (operand), una expresión con selector (primaryExpression selector), una expresión con índice (primaryExpression index), una expresión con argumentos (primaryExpression arguments), una expresión de append (appendExpression), una expresión de length (lengthExpression) o una expresión de capacidad (capExpression).

Operandos (operand)

Esta regla define la sintaxis para los operandos de las expresiones, que pueden ser literales (literal), identificadores (IDENTIFIER) o expresiones entre paréntesis (LEFTPARENTHESIS expression RIGHTPARENTHESIS).

Literales (literal)

Esta regla define la sintaxis para los literales, que pueden ser literales enteros (INTLITERAL), literales flotantes (FLOATLITERAL), literales de caracteres (RUNELITERAL), literales de cadenas de caracteres en bruto (RAWSTRINGLITERAL) o literales de cadenas de caracteres interpretadas (INTERPRETEDSTRINGLITERAL).

Sentencias (statement)

Esta regla define la sintaxis para las diferentes sentencias en el lenguaje, como sentencias de impresión (PRINT), sentencias de retorno (RETURN), sentencias de ruptura (BREAK), sentencias de continuación (CONTINUE), sentencias simples (simpleStatement), bloques de código (block), sentencias de selección (switch), sentencias condicionales (ifStatement) y bucles (loop).