Instituto Tecnológico y de Estudios Superiores de Monterrey

TC3002B Desarrollo de Aplicaciones avanzadas de Ciencias Computacionales Gpo 502
Módulo 3 – Compiladores: Tarea 2

José Eduardo de Valle Lara A01734957

Profesora:

Elda Guadalupe Quiroga González

Generación automática de Analizadores de Léxico y Sintaxis (Scanners and Parsers)

Deberás documentar al menos 3 herramientas distintas, donde una de ellas será Lex&Yacc ó Flex&Bison (que son las más clásicas) y las otras 2 serán a tu elección y dependerá (principalmente) del lenguaje en el que estés planeando desarrollar nuestro mini-proyecto.

Flex&Bison (Free Software Foundation, 2021), (Princeton University 1995)

Son herramientas de software diseñadas para la creación de analizadores léxicos, sintácticos y compiladores. Operan juntas para procesar cadenas estructuradas de texto.

- **Flex**: Fast Lexical Analyzer Generator, es un generador de analizadores léxicos, procesa una secuencia de caracteres de entrada para producir tokens mediante expresiones regulares.
- **Bison**: generador de analizadores sintácticos con una gramática formal especificada por el usuario. Bison determina si la secuencia de tokens proporcionada posee una estructura válida.

Ambas herramientas son estándar en entornos de desarrollo GNU/Linux y sistemas operativos tipo Unix, pero pueden ser compiladas para plataformas como Windows y macOS, son herramientas de software libre y gratis, Flex se distribuye bajo la licencia BSD y Bison bajo la licencia GNU (GPL). El lenguaje de programación para Flex y Bison es C, los archivos de código fuente se deben compilar para producir un ejecutable final. La interacción con Flex y Bison se realiza a través de la terminal de comandos

Las especificaciones se escriben en archivos de texto plano **.l** para Flex y **.y** para Bison. El archivo de especificación de Flex contiene definiciones de alias para expresiones regulares con pares de pares de patrón-acción donde el patrón es una expresión regular y la acción es un bloque de código de C. El archivo de especificación de Bison contiene la declaración de tokens, asociatividad y precedencia de operadores con tipos de datos para símbolos gramaticales, también se define la gramática del lenguaje en una notación similar a la Backus-Naur Form.

Flex&Bison ofrecen un control total sobre el procesamiento ya que el usuario define las acciones semánticas en C permitiendo la construcción de Abstract Syntax Trees, evaluación de expresiones y generación de código intermedio.

Herramientas para el proyecto

El proyecto se realizará en el lenguaje Go usando la librería alecthomas/participle

alecthomas / participle (Thomas A., 2025)

Es un generador de parsers open source distribuida con la Licencia MIT, la gramática se define en el lenguaje Go. Esta es una librería integrada al lenguaje Go, no es una herramienta externa con su propia sintaxis, las espresiones se declaran usando structs nativos y se les asigna tags de **participle** para entender cada regla gramatical:

- @: espera un token terminal definifo por el lexer para asignarlo al campo, un emplo es **field string \`** `@ident`

La entrada principal es una cadena de texto que representa la expresión a analizar mediante un lexer que la divide en tokens. Participle usa un lexer basado en expresiones regulares para definir los tokens como **Int**, **Float**, **Ident**.

El tipo de ejecución es estático y se describe con el siguiente proceso:

1. Definición: se define la gramática con tags de participle en Go structs
2. Construcción: el parser se construye en tiempo de ejecución
3. Ejecución: se llama al parser para ejecutar el análisis sintáctico sobre la cadema de entrada

Es facil integrar lógica personalizada, la librería espera que el usuario defina las reglas gramaticales en Go structs como ya se definió previamente, pero también ofrece la posibilidad de implementar un lexer personalizado y proporcionarlo al constructor de participle.

goccmack / gocc (Ackerman M., 2024)

gocc es un compiler kit de código abierto para Go bajo la licencia MIT, es una herramienta de analizadores sintácticos. Su función es tomar especificaciones formales de un lenguaje (definiendo sus tokens mediante expresiones regulares y sus reglas sintácticas) y, a partir de ellas, generar automáticamente todo el código de Go necesario para leer, validar y estructurar cadenas de texto en ese lenguaje. Es una librería que se accede mediante la API de Go desde la terminal que genera el código en Go, el programa llama las funciones del parser generado para procesar la entrada en runtime.

La herramienta no trabaja directamente con cadenas de texto, trabaja con archivos de gramática en un formato específico parecifo a Extended Backus Naur Form:

- Especificación Léxica: un archivo que describe las expresiones regulares para cada token, el léxico se basa en expresiones regulares para definir los tokens.
- Especificación sintáctica: un archivo que define las reglas de gramática implementando un analizador sintáctico LALR (Look-Ahead Left-to-right, 1 lookahead token)

Implementar código propio es sencillo ya que el usuario puede definir sus propias reglas gramaticales en el archivo de especificación sintáctica construyendo el Abstract Syntax Tree.

Ejercicio en alecthomas / participle

El ejercicio se obtuvo de la documentación oficial en GitHub, es un script que lee expresiones matemáticas desde la terminal, construye un Abstract Syntax Tree a partir de esa expresión y posteriormente calcula el valor numérico de la operación matemática.

Inicializar el proyecto

- go mod init

Instalar las librerías de GitHub

- go get github.com/alecthomas/participle/v2
- go get github.com/alecthomas/kong

Correr el proyecto: un script de go se ejecuta con go run main.go, pero este script espera recibir parámetros desde la terminal al momento de iniciarlo, algunos ejemplos son:

- go run main.go "10 / 2 + 5"
- go run main.go -s a=10 -s b=5 "a * b"
- go run main.go --ast "1 + 2 ^ 3"

Uso de LLMs (ChatGPT)

Se usó ChatGPT como apoyo auxiliar para complementar la información restante que solicita la rúbrica y que no fue encontrada directamente en la documentación, en cosas muy puntuales como los fundamentos teóricos de cada herramienta. Cabe destacar que se usó mayormente para la configuración del entorno del ejemplo tomado directamente de la documentación oficial de alecthomas/participle.

Bibliografía

- Ackerman, M. (2025). gocc: A compiler kit written in Go. Recuperado de https://github.com/goccmack/gocc
- Free Software Foundation (2021). The Bison Parser Generator. GNU Operating System. Recuperado de https://www.gnu.org/software/bison/manual/bison.html
- Princeton University (1995) - Flex, version 2.5. Recuperado de https://www.cs.princeton.edu/~appel/modern/c/software/flex/flex.html
- Thomas, A. (2025). participle: A dead simple parser package for Go. Recuperado de https://github.com/alecthomas/participle