Mohammed Alqadda

Programming Languages

Fall 2024

Course Project

Implement a Scanner/Parser for a simple language for expressions involving complex numbers using FLEX (Fast Lexical Analyzer Generator) and Bison (parser generator). Your language should support all four basic arithmetic operations between complex numbers, as well as parentheses. A complex number should be specified as (a ± bi), where a and b are integers. Note that numbers that are missing either the real component or the imaginary component are considered valid.

Example: (5 – 2i) - (-2 + 3i)\*(-5i) + (7)

Installing FLEX and Bison:

For windows,

<http://gnuwin32.sourceforge.net/packages/flex.htm>

<http://gnuwin32.sourceforge.net/packages/bison.htm>

For mac,

brew install flex

brew install bison

**Defining Terms for Undertanding**

Scanning and parsing serve to recognize the structure of the program, without regard to its meaning.

The **scanner** reads characters (‘ i’, ‘n’, ‘t’, ‘ ’, ‘m’, ‘a’, ‘i’, ‘n’, ‘(’, ‘)’, etc.) and groups them into tokens, which are the smallest meaningful units of the program. Scanning is also known as lexical analysis. The principal purpose of the scanner is to simplify the task of the parser, by reducing the size of the input (there are many more characters than tokens) and by removing extraneous characters like white space. The scanner also typically removes comments and tags tokens with line and column numbers, to make it easier to generate good diagnostics in later phases. (from course book)

A **parser**: Parsing organizes tokens into a parse tree that represents higher-level constructs (statements, expressions, subroutines, and so on) in terms of their constituents. Each construct is a node in the tree; its constituents are its children. The root of the tree is simply “program”; the leaves, from left to right, are the tokens received from the scanner. Taken as a whole, the tree shows how the tokens fit together to make a valid program. (from course book)

**Complex numbers** are numbers that consist of two parts: a **real part** and an **imaginary part**. For example, 3+4i

**Real part**: The part of a complex number that does not involve the imaginary unit i. The real part is 3.

**Imaginary part**: The part of a complex number that is multiplied by the imaginary unit i. the imaginary part is 4i.

**Flex (Fast Lexical Analyzer Generator),** or simply Flex, is a tool for generating lexical analyzers scanners or lexers. (<https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/>)

**Bison** is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1) parser tables. **(<https://www.gnu.org/software/bison/#:~:text=Bison%20is%20a%20general%2Dpurpose,LR(1)%20parser%20tables>)**

**Project Requirements**

* It should support all four basic arithmetic operations between complex numbers, as well as parentheses
* A complex number should be specified as (a ± bi), where a and b are integers.
* Note that numbers that are missing either the real component or the imaginary component are considered valid.

**Terminal Commands**

FLEX files end with .l

After creating the file we do,

flex filename.l

to get,

lex.yy.c

then,

gcc lex.yy.c -o GIVE NAME -L/usr/local/opt/flex/lib -I/usr/local/opt/flex/include -lfl

compile the FLEX file and create an executable

now we can run it using,

echo "write what you want" | ./GIVENAME

or,

./GIVE NAME

Bison:

Filename.y

After creating,

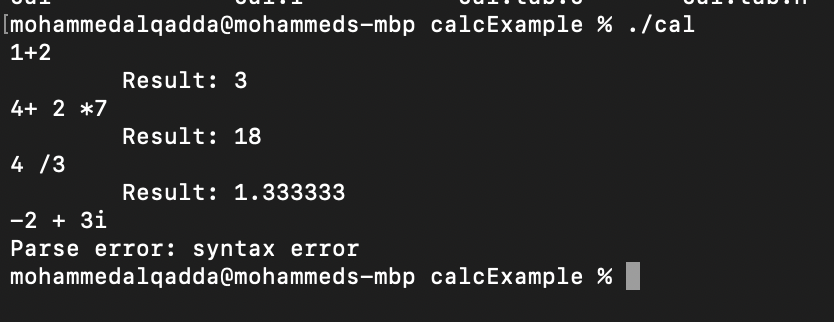
bison -d filename.y

we get,

filename.tab.c and filename.tab.h

**Example to Learn From,**

<https://github.com/meyerd/flex-bison-example/blob/master/README.md>

****