Project #1 — Lexical Analyzer

Jared Dyreson, Chris Nutter

California State University, Fullerton

Contents

1 Introduction **2**

2 How to Use **2**

3 Design **2**
3.1 Regular Expression 2
3.2 Finite State Machines 3

4 Shortcomings **5**

1 Introduction

This project was intended to be a lexical analyzer for our compiler and was aptly named "Lexi". The goal of Lexi is to parse out the contents of a source document and generate meaningful lexemes. These lexemes were to adhere to a specific set of regular expressions to define a token.

2 How to Use

0. **Use a Unix-based OS.**
1. Extract ***lexi.zip***
2. Enter *lexical_analysis* folder
3. Run ***make*** in Terminal
4. Output can be run again with ***./lexi [input]***

3 Design

3.1 Regular Expression

The way *Lexi* parses each line and determines the identifier type is through the use of *regular expressions*. Being able to determine the identifier is crucial in defining the token's contents. *Lexi* after processing the file and creating a vector of strings that parses line by line which is then fed through a function that reads each character and determines one of the each lexeme types.

1. ***Comments*** determines any line that has *!* and ends with a trailing *!*. Multiple comments in a line are supported.

   ```
   (!.*!)
   ```

2. ***Keywords*** finds any word that is considered reserved for the structure of the language including data types, control-flow operators, and other key-defining words for the language.

   ```
   (int|float|bool|true|false|(end)?if|else|then|while(end)?
   |do(end)?|for(end)?|(in|out)put|and|or|not)
   ```

3. ***Number*** is any integer, float, double, size_t, (etc.) value for identifying amount.

   ```
   (?:\b)([-+]?\d*.?\d+)?(?=\b)
   ```

4. ***Identifier*** grabs any word that is not within a *comment* or *keyword* field.

   ```
   ([a-zA-Z]+(\d*)?)
   ```

5. ***Separators*** finds any symbol that helps keep the contents contained.

   ```
   ((|)|{|}|[|]|"|'|,)
   ```

6. ***Operators*** obtains symbols that the language uses for operation.

   ```
   (+|-|*|/|=|>|<|>=|<=|&+||+|%|^!$|^)
   ```

7. ***Terminators*** are symbols signalling end of a line.

   ```
   (;|$)
   ```

3.2 Finite State Machines

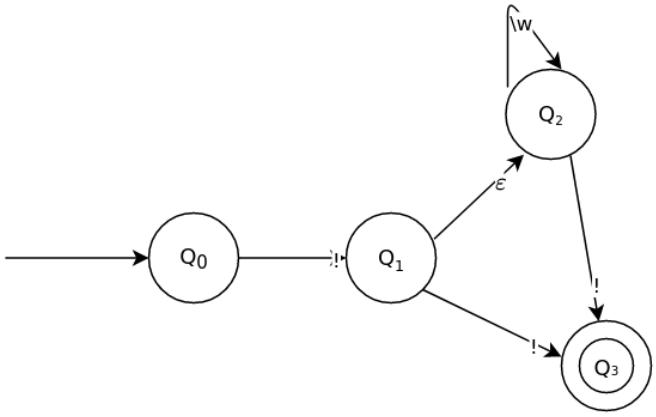

Figure 1: Comment

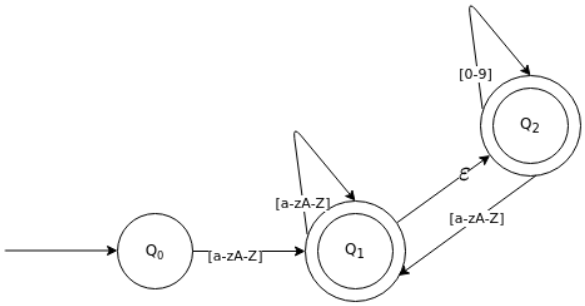

Figure 2: Identifier

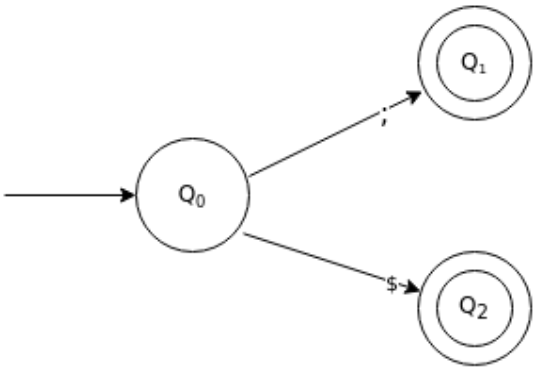

Figure 3: Terminator

4 Shortcomings

We had problems generating FSMs for large regular expressions that would be a headache to be written out. Creating formatting for line numbers with comments was a headache due to comments' lengths being fairly large and not being inline.