Ethan Letrong

Juan Linares

CS323 Assignment 1 Documentation

1. Problem Statement.

Write a Lexical analyzer that will take a text of code as an input. It will then separate the entire text document and create a token and lexeme based off of each part of the text document. Essentially the assignment is to create a FSM or to create one based off of the example in the book.

1. How to use your program

User must have a text document containing code and stored in the same directory as the code. When the program initiates, it will prompt the user to type in the text document’s exact file name. If the text document is properly named and called, it will run through several functions based off the document. This will produce an output displaying the proper lexemes and tokens that the original problem needed.

1. Design of the program.

Program uses vectors to dynamically store the contents of the code. They are stored in vectors of type CHAR, STRING, and STRUCT. The STRUCT will store two data types: Lexeme and Token. The code will indiscriminately read in the data into a string array. Afterwards it will go through that array and convert it into a character array to be manipulated. After manipulation it will be stored in the final STRUCT array for manipulation.

In terms of Algorithm, the main algorithm involves finding the existence of a symbol without manipulating the vector used. Afterwards it calls onto another function which based off the previous algorithm will systematically take apart the vector and populate the final vector with tokens and lexemes.

1. Any Limitation

Limitations are based off of information provided by the instructor based off of their documentation. While inputs and examples are shown, only one document has a list of keywords, symbols, spacers, and identifiers. This becomes an issue as it isn’t a proper guideline for lexemes and tokens since they vary differently from the outputs. As a result, if other code documents are different from the ones given to the students, their FSM would not be able to properly categorize each token and lexeme. So this may result in multiple instances of mis-categorized outputs.

If the text document containing code is too large, there will be a memory issue and may slow down performance or not enough memory will be allocated for the function to work.

1. Any shortcomings

Code isn’t fully optimized, but solves the original problem.