**Problem Statement**

Write a procedure – lexer(), that writes a procedure – lexer(), that returns a token when it is needed. The lexer() should return a record, one field for the token, and another field the actual "value" of the token (lexeme), i.e., the instance of a token. Construct a FSM and write the REs for Identifiers, Real and Integer, and also show the NFSM using Thompson.

**Usage**

Steps:

1. Open Command Prompt (Windows) with a working directory containing the app and the three test files
2. Enter LexerRat20f.exe <filename> (Windows) for the file to be read by the lexer
   1. LexerOutput.txt will be created
3. If desired, view or copy the contents of LexerOutput.txt before proceeding to step 4 because executing the program again will overwrite LexerOutput.txt
4. Repeat step 2 for all input files

**Program Design**

Our program consists of the main function that continuously calls the lexer function with the ifstream to the input as an argument. After calling the lexer and getting the record, it will print the output to the output file named "LexerOutput.txt" if the token held in the returned record does not equal "fileend". If the token does equal "filend", the do while will end, and the program will close the streams to the input and output files and terminate.

The lexer function begins by setting the string "state" to "start", the lexeme string to empty, the int "done" to 0, and the char c to empty. It then enters a while loop that will only end when done == 0. In the loop, the program reads one char from the input file and depending the state and the input, change the string "state" to another state or set done = 1 and unget from the input file to end the lexer call and set up for the next call. If done == 1, the program will create the record with the state as its token and the lexeme and return the record to the main function. To determine certain states, we used three standard C++ vectors to hold string for keywords and chars for operators and separators. We used the standard C++ find function to compare our lexeme and our char "c" to the elements in our vectors.

**Limitations**

None

**Shortcomings**

None

**Regular Expressions, NFSM and DFSM**

1. Int: d+
2. Real: d+.d+
3. Identifier: l ( l | d | \_ )\*

Real-Int FSM

Diagram

Description automatically generated

Identifier FSM

A picture containing diagram

Description automatically generated