Link to github repo: <https://github.com/MarianC10/flcd>

I have created two different csv files containing finite automata, one for identifying integer constants and one for identifiers. I then loaded those files in memory, having created my own data structure for a finite automaton (which contains a bunch of sets and map for storing transitions), and I have replaced the regex expression matchers from my lexical analyzer with these finite automata. The matching algorithm is pretty simple, due to the fact that the automata are deterministic, so all I had to do was to iterate through the input string character by character and move to the corresponding next state (which can be found searching through the transitions).

EBNF structure of the csv files:

finiteautomaton = initialstate{initialstate} finalstate {finalstate} transition {transition}

transition = state “,” (alphanumerical | “\_”) “,” state

alphanumerical = letter | digit

finalstate = state “,” “final”

initialstate = state “,” “initial”

state = “q” digit {digit}

letter = “a” | “b” | “c” | … | “z” | “A” | “B” | … | “Z”

digit = "0" | "1" | ... | "9"