**CS323 Documentation**

1. **Problem Statement**

This program is a simple lexer that reads a file and identifies the different tokens in the file. It uses several functions to perform different tasks such as opening and closing files, initializing and finalizing the output file, checking if a value in an array matches the value passed to the function, and defining finite state machines (FSMs) for different token types.

1. **How to use Our program**

Step1) Click on lexer.exe

Step2) The input file (test1.rat23s)

Step3) The output file (output1.txt)

Or

Step 1) go to command line

Step 2) type lexer.exe < test1.rat23s > < output1.txt >

Step 3) Hit the CR

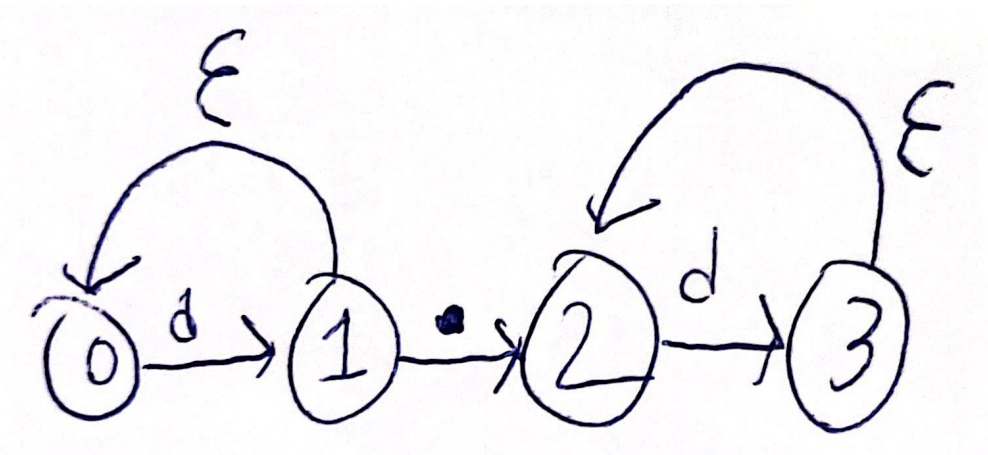
1. **Design of Our program**

**Repository:** <https://github.com/MichaelR13/cpsc323.Project>

**Regular Expressions:**

* **Real: [d]+ [.] [d]+**
* **Integers: [d]**

**Thompson NFSM:**

****

**Our NFSM consisted of 4 states:**

* **0: Starting state -> state 1 on [d]**
* **1: Integer state -> state 2 on [.]**
* **1: Integer state -> state 1 on [d]**
* **2: Real state -> state 3 on [d]**
* **3: Error state -> state 3 on [d]**

**Functions:**

* function openFiles() to prompt the user for the input and output file names, and then open the input and output files. If there is an error opening the files, the program exits.
* function initPrint() to print a header for the output file.
* function checkArr() to check if a given value is in a given array.
* function endPrint() to print a footer for the output file.
* function intRealFSM() to implement a finite state machine for recognizing integers and real numbers.
* function isKeyword() to check if a given identifier is a keyword.
* function lexer() to tokenize the input file and output the tokens to the console. The function uses a series of if statements to identify and handle different types of tokens, including identifiers, keywords, integers, real numbers, operators, and separators.

The primary data structure used in this program is a two-dimensional array that represents a finite state machine (FSM) used to identify whether a number is an integer or a real number. Additionally, two one-dimensional character arrays, ops and seps, are used to store operators and separators.

**Algorithm:**

The algorithm used in this program is a finite state machine. For example, intRealFSM uses a finite state machine with three states to determine whether a given number is an integer or a real number. The function isKeyword checks if the identifier is a keyword by comparing it to a list of predefined keywords.

1. **Any Limitation**

Being able to align each token/lexeme pair when printing/writing.

1. **Any shortcomings**

Implementation of a potential use of 2 finite state machines; We had previously attempted to utilize 2 finite state machines and use their determined states to tokenize. This approach provided too many issues, so we reworked the lexer to use 1 finite state machine for integers and reals.

**Samples:**

Here is an example input and the corresponding output that can be produced using the functions in the provided code:

lexeme\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_token

while keyword

( seperator

true identifier

) seperator

{ seperator

cout identifier

< operator

< operator

" seperator

Hello identifier

, seperator

world identifier

! operator

" seperator

< operator

< operator

endl identifier

; seperator

string identifier

name identifier

; seperator

int identifier

age identifier

, seperator

birthYear1 identifier

, seperator

birthYear2 identifier

; seperator

cout identifier

< operator

< operator

" seperator

What identifier

is identifier

your identifier

name identifier

? seperator

" seperator

; seperator

cin identifier

> operator

> operator

name identifier

; seperator

cout identifier

< operator

< operator

" seperator

How identifier

old identifier

are identifier

you identifier

? seperator

" seperator

; seperator

cin identifier

> operator

> operator

age identifier

; seperator

birthYear1 identifier

= operator

2023 integer

- seperator

age identifier

; seperator

birthYear2 identifier

= operator

2022 integer

- seperator

age identifier

; seperator

cout identifier

< operator

< operator

" seperator

You identifier

were identifier

born identifier

in identifier

" seperator

< operator

< operator

birthYear1 identifier

< operator

< operator

" seperator

or identifier

" seperator

< operator

< operator

birthYear2 identifier

< operator

< operator

endl identifier

; seperator

cout identifier

< operator

< operator

" seperator

Do identifier

you identifier

want identifier

to identifier

continue keyword

? seperator

( seperator

y identifier

/ seperator

n identifier

) seperator

" seperator

; seperator

char identifier

answer identifier

; seperator

cin identifier

> operator

> operator

answer identifier

; seperator

if keyword

( seperator

answer identifier

== operator

' seperator

n identifier

' seperator

) seperator

{ seperator

break keyword

; seperator

} seperator

} seperator

return identifier

0 integer

; seperator