Lexer Documentation

1. Problem Statement

This program is a lexical analyzer and its function is to scan source code, detect token boundaries, find valid and invalid tokens, and provide a list of all tokens for future processing in the compiler. Essentially, the main purpose of the lexical analyzer is to prepare the source code for the following stage of the compiler: syntactic analysis.

2. How to use your program

Our program uses CMake.To run the program :

In case of needing to compile use this:

```
g++ main.cpp classes/Lexer.cpp -o program
```

```
To run the program go to your terminal and run:
If windows: program <input.txt> (which is the file path/address
to use with the code)
```

File names to use:

```
smlrat25s.txt

medrat25s.txt

Largerat25s.txt

Input.txt
```

3. Program Design

In our design of the lexical analyzer, we used C++ and a finite state machine approach to process tokens. Our lexer operates as such:

1. *The entire source file is loaded into memory.*
2. *A finite state machine is used to transition between different states based on the input character from the string buffer.*
3. *The lexer analyzes two characters at a time to determine the boundaries of each token. Using std::stringview the lexer analyzes the current character and the next character in the buffer and recognizes identifiers, integers, keywords, operators, separators, comments, and unknown or invalid tokens.*
4. *Keywords were stored using std::unordered_map for quick lookup (average O(1) time complexity).*

The states operate as follows:

Start State: Consumes whitespace and identifies the start of a lexeme and determines transition to other states.

Identifier State: Reads until a non-alphanumeric character or "_" is encountered and once the end of the lexeme is found it checks for a keyword.

Integer State: Reads until a non-digit or decimal is encountered. If a decimal is found followed by a digit, it transitions to Real Number State.

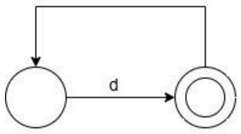
Real Number State: Reads until a non-numeric character is encountered and verifies floating point format.

Operator State: Recognizes operators.

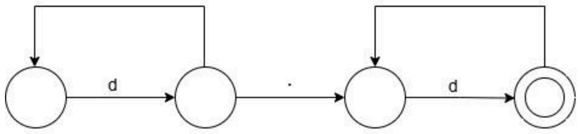
Comment State: Detects comments beginning with `[*` and continues until `*]` is encountered and handles EOF within comments as error.

Error State: Used for detecting unknown characters and returns an error token (`UNKW`).

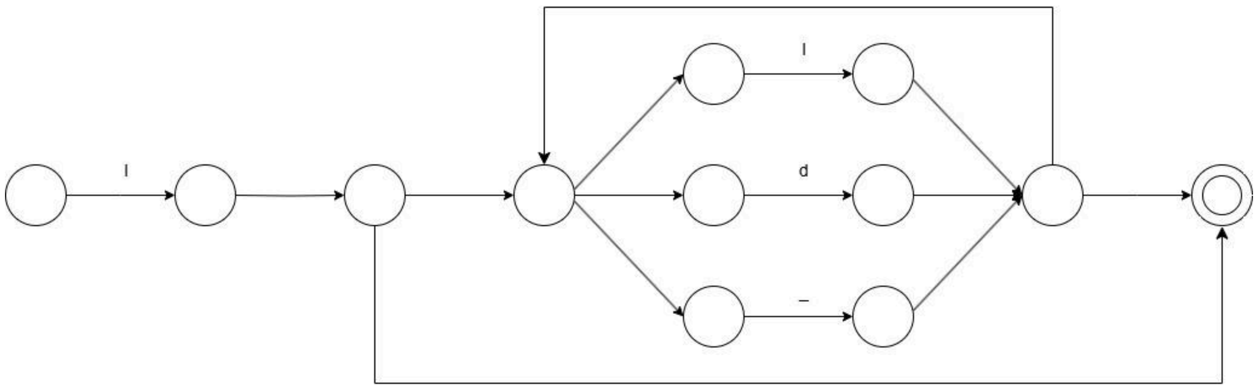
RE for Integer: ***d+***



RE for Real: ***d+.d+***



RE for Identifier: ***l (l | d | _)****



4. Limitations

None.

5. Any shortcomings

None.