A Lexical Analyzer

Author(s): Hareena Chowdary Polavaram
Date: 02 Nov. 23

A Lexical Analyzer

Phase 1: INTRODUCTION

- In this report, the task is to implement a lexical analyzer for a self-designed programming language called Cminus.
- The lexical analyzer's purpose is to read source code written in Cminus and produce a list of tokens as output.

Phase 2: SPECIFICATION

To implement a lexical analyzer for a self-designed Cminus programming language (a tiny simplified subset of the C language) using Python.

Objectives that we should follow in order to write a code:

2.1: Language Specification:

- Cminus is a case-sensitive language using ASCII characters.
- It supports int and float data types.
- Supports arithmetic, logical, relational, and assignment operators.
- Control flow statements include if-else, while, and compound statements.
- Supports single-line and multi-line comments (/* ... */).
- Identifiers start with a letter and can contain alphanumeric characters.
- Literal strings are enclosed in single quotes.
- Keywords (reserved words): int, float, if, else, exit, while, read, write, return.

2.2: Lexical Analyzer Specification:

- Read source code line by line and store it in a buffer.
- Scan the buffer to identify tokens based on language specification.
- Create a token object with type and value for each token.
- Append the token object to a list of tokens.
- Continue scanning until the end of the buffer or an error is encountered.
- Return the list of tokens as output or display an error message for encountered errors.

Phase 3: DESIGN

The code defines a lexer using the PLY (Python Lex-Yacc) library, which is commonly used for parsing and tokenizing input text. The lexer's purpose is to break down input text into smaller units, called tokens. Analysis of the design and functionality of my code:

1. Importing PLY:

The code starts by importing the PLY library with the alias 'lex,' which is used for lexical analysis.

2. Token Definitions:

The code defines a list of token names (such as KEYWORD, IDENTIFIER, CONSTANT, ARITH_OP, LOGIC_OP, SEPARATOR, and COMMENT) that represent different types of tokens in the input code.

3. Regular Expressions for Token Matching:

- Regular expressions (regex) are used to define the patterns for each token type. For example, `t_LOGIC_OP` matches logical operators like '&', '|', '||', and '=='.
- `t_SEPARATOR` matches various separators like ';', '(', ')', and '{'.
- `t_ARITH_OP` matches arithmetic operators like '+', '-', '*', '/', '%', and '='.

- `t_IDENTIFIER` matches identifiers (variable and function names) and checks if they are also keywords. Keywords are a specific subset of identifiers.
- `t_CONSTANT` matches numeric constants, both integers and floating-point numbers.
- `t_ignore` specifies which characters to ignore (whitespace and tabs).
- `t_newline` handles newlines and updates the line number.

4. Token Handling Functions:

Custom functions are defined for handling different token types (`t_IDENTIFIER`, `t_CONSTANT`, `t_COMMENT`, and `t_error`). These functions are called when a token is matched, and they perform specific actions on the matched tokens.

5. `t_error` Function:

This function is called when an unrecognized character is encountered in the input. It prints an error message and skips the problematic character.

6. Lexer Building:

The lexer is constructed using `lex.lex()`.

7. Testing the Lexer:

The code includes a section for testing the lexer with example input. It reads the input from a file named 'Cminus.txt,' feeds it to the lexer, and prints the identified tokens along with their types.

8. Usage:

To use this lexer for your specific input code, you need to replace the 'Cminus.txt' file with your code or provide the input text in another way.

Phase 4: RISK ANALYSIS

The code is a Lexical Analyzer. No specific risks were identified in this project. However, potential issues might include:

1.Testing Coverage:

The code includes a basic test of the lexer, but it may lack comprehensive testing to cover various input scenarios and edge cases.

2.Portability:

The code is written for a specific use case and input file ('Cminus.txt'). It may not be easily portable to different projects without modification.

3.Dependency on PLY:

The code relies on the PLY library. If PLY's support or compatibility changes, it could affect the code's functionality.

4.Regular Expression Complexity:

The regular expressions used to define token patterns are relatively complex. There's a risk of regex errors or inefficiencies. While the provided regular expressions seem to be valid, complex regular expressions can be challenging to debug and maintain.

Phase 5: VERIFICATION

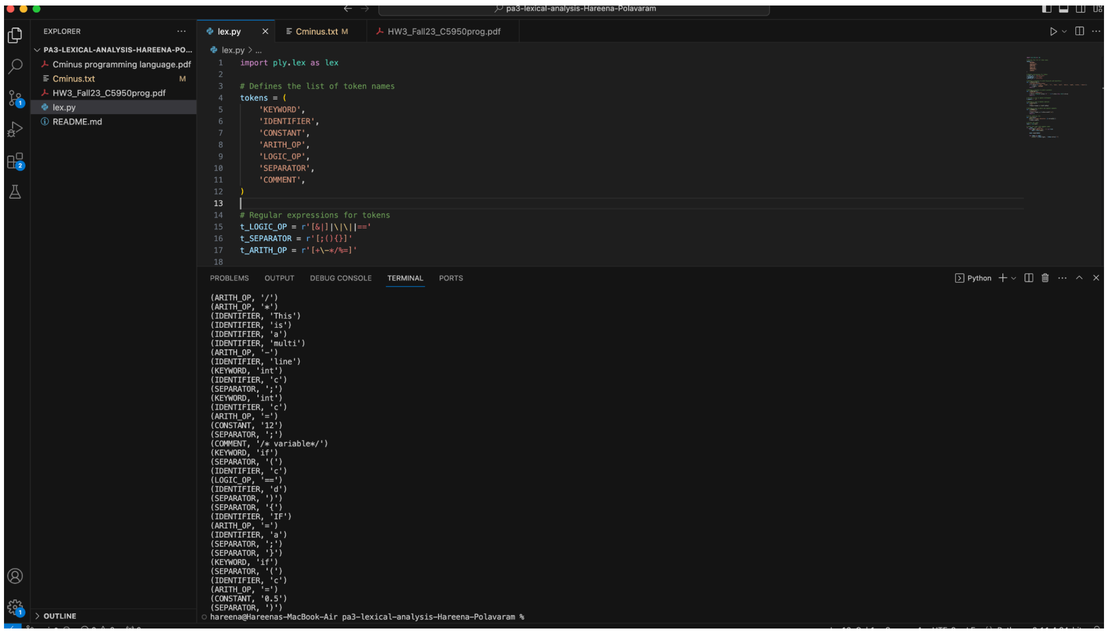
All the functions in the code are tested with multiple expressions and satisfied the required output.

Phase 6: CODING

The code of this program is included in the zip file.

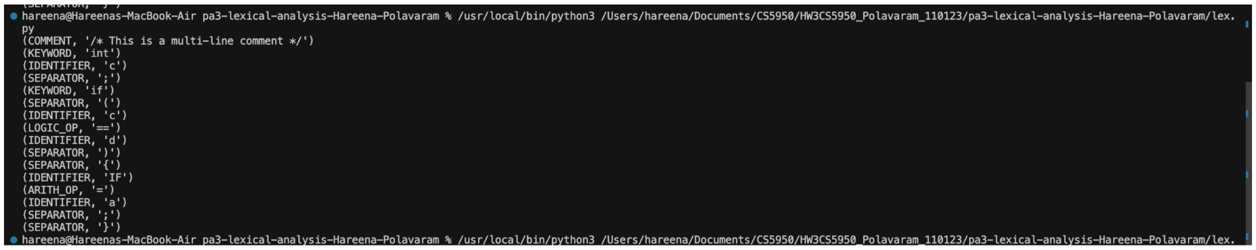
Phase 7: TESTING

7.1 Input & Output Specifications:

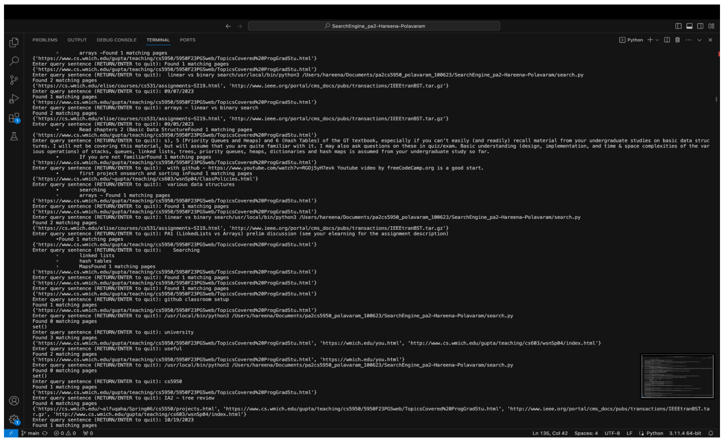


```python
import ply.lex as lex

# Defines the list of token names
tokens = (
    'KEYWORD',
    'IDENTIFIER',
    'CONSTANT',
    'ARITH_OP',
    'LOGIC_OP',
    'SEPARATOR',
    'COMMENT',
)

# Regular expressions for tokens
t_LOGIC_OP = r'[&|]|\|\||=='
t_SEPARATOR = r'[;(){}]'
t_ARITH_OP = r'[+\-*/%=]'
```

```
(ARITH_OP, '/')
(ARITH_OP, '*')
(IDENTIFIER, 'This')
(IDENTIFIER, 'is')
(IDENTIFIER, 'a')
(IDENTIFIER, 'multi')
(ARITH_OP, '-')
(IDENTIFIER, 'line')
(KEYWORD, 'int')
(IDENTIFIER, 'c')
(SEPARATOR, ';')
(KEYWORD, 'int')
(IDENTIFIER, 'c')
(ARITH_OP, '=')
(CONSTANT, '12')
(SEPARATOR, ';')
(COMMENT, '/* variable*/')
(KEYWORD, 'if')
(SEPARATOR, '(')
(IDENTIFIER, 'c')
(LOGIC_OP, '==')
(IDENTIFIER, 'd')
(SEPARATOR, ')')
(SEPARATOR, '{')
(IDENTIFIER, 'IF')
(ARITH_OP, '=')
(IDENTIFIER, 'a')
(SEPARATOR, ';')
(SEPARATOR, '}')
(KEYWORD, 'if')
(SEPARATOR, '(')
(IDENTIFIER, 'c')
(ARITH_OP, '=')
(CONSTANT, '0.5')
(SEPARATOR, ')')
hareena@Hareenas-MacBook-Air pa3-lexical-analysis-Hareena-Polavaram %
```



```
hareena@Hareenas-MacBook-Air pa3-lexical-analysis-Hareena-Polavaram % /usr/local/bin/python3 /Users/hareena/Documents/CS5950/HW3CS5950_Polavaram_110123/pa3-lexical-analysis-Hareena-Polavaram/lex.
py
(COMMENT, '/* This is a multi-line comment */')
(KEYWORD, 'int')
(IDENTIFIER, 'c')
(SEPARATOR, ';')
(KEYWORD, 'if')
(SEPARATOR, '(')
(IDENTIFIER, 'c')
(LOGIC_OP, '==')
(IDENTIFIER, 'd')
(SEPARATOR, ')')
(SEPARATOR, '{')
(IDENTIFIER, 'IF')
(ARITH_OP, '=')
(IDENTIFIER, 'a')
(SEPARATOR, ';')
(SEPARATOR, '}')
hareena@Hareenas-MacBook-Air pa3-lexical-analysis-Hareena-Polavaram % /usr/local/bin/python3 /Users/hareena/Documents/CS5950/HW3CS5950_Polavaram_110123/pa3-lexical-analysis-Hareena-Polavaram/lex.
```



```
        ◦       arrays -Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):  linear vs binary search/usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): 09/07/2023
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): arrays - linear vs binary search
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): 09/05/2023
        •       Read chapters 2 (Basic Data StructureFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): s), 5 (Priority Queues and Heaps) and 6 (Hash Tables) of the GT textbook, especially if you can't easily (and readily) recall material from your undergraduate studies on basic data struc
tures. I will not be covering this material, but will assume that you are quite familiar with it. I may also ask questions on these in quiz/exam. Basic understanding (design, implementation, and time & space complexities of the var
ious operations) of stacks, queues, linked lists, trees, priority queues, heaps, dictionaries and hash maps is assumed from your undergraduate study so far.
        •       If you are not familiarFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):  with github - https://www.youtube.com/watch?v=RGOj5yH7evk Youtube video by freeCodeCamp.org is a good start.
        •       first project onsearch and sorting inFound 1 matching pages
{'https://www.cs.wmich.edu/~gupta/teaching/cs603/wsnSp04/ClassPolicies.html'}
Enter query sentence (RETURN/ENTER to quit):  various data structures
        •       searching
        ◦       arrays - Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): linear vs binary search/usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 2 matching pages
{'https://cs.wmich.edu/elise/courses/cs531/assignments-SI19.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.tar.gz'}
Enter query sentence (RETURN/ENTER to quit): PA1 (LinkedLists vs Arrays) prelim discussion (see your elearning for the assignment description)
        •Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit):    Searching
        ◦       linked lists
        ◦       hash tables
        ◦       MapsFound 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): github classroom setup
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): university
Found 3 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): useful
Found 2 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'https://wmich.edu/you.html'}
Enter query sentence (RETURN/ENTER to quit): /usr/local/bin/python3 /Users/hareena/Documents/pa2cs5950_polavaram_100623/SearchEngine_pa2-Hareena-Polavaram/search.py
Found 0 matching pages
set()
Enter query sentence (RETURN/ENTER to quit): cs5950
Found 1 matching pages
{'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html'}
Enter query sentence (RETURN/ENTER to quit): IA2 - tree review
Found 4 matching pages
{'https://cs.wmich.edu/~alfuqaha/Spring06/cs5550/projects.html', 'https://www.cs.wmich.edu/gupta/teaching/cs5950/5950F23PGSweb/TopicsCovered%20ProgGradStu.html', 'http://www.ieee.org/portal/cms_docs/pubs/transactions/IEEEtranBST.ta
r.gz', 'http://www.cs.wmich.edu/gupta/teaching/cs603/wsnSp04/index.html'}
Enter query sentence (RETURN/ENTER to quit): 10/19/2023
Found 1 matching pages
```

Phase 8: REFINING THE PROGRAM

We can make few improvements for better code structure, efficiency, and usability. Here are some refinements:

1.Error Handling:

The error handling mechanism in the code is limited to printing an error message and skipping the character. This is not very robust. More sophisticated error handling, such as logging errors and providing better feedback to the user, would be preferable.

2.Performance and Efficiency:

Complex regular expressions, especially in large input files, can impact performance. Depending on the use case, performance may need to be optimized.

Phase 9: MAINTAINANCE

This application can be futher improved.