Programming Assignment 3
A Lexical Analyzer[1]

Assignment spec may change depending upon discussion in the class, this is a working doc

Release Time	Due Date
October 12, 2023	**November 2, 2023**

Objectives

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

Problem Specification

Language Specification: The self-designed language is called **Cminus** and it has the following features:

- It is a case-sensitive language that uses ASCII characters.
- It supports only two data types: basic type int and a standard data type float.
- It supports arithmetic, logical, relational, and assignment operators.
- It supports if-else, while, and compound statements for control flow.
- It supports single-line and multi-line comments that start with /* and end with */
- It supports identifiers that start with a letter and can contain alphanumeric characters.
- It supports literals that are enclosed in single quotes for strings, and supports only decimal notation for floating point numbers.
- It supports keywords that are reserved for the language and cannot be used as identifiers. The keywords are: int, float, if, else, exit, while, read, write, and return.

Lexical Analyzer Specification: The lexical analyzer is a program that takes an input source code file written in Cminus and produces a list of tokens as output. A token is a pair of a token type and a token value. The token types are:

- KEYWORD: A reserved word in the language.
- IDENTIFIER: A user-defined name for a variable or a function.
- CONSTANT: A constant value of a data type.
- ARITH-OP: A symbol that performs an arithmetic operation on operands.
- LOGIC-OP: A symbol that performs a logical operation on operands.
- SEPARATOR: A symbol that separates tokens or groups them together.
- COMMENT: A text that is ignored by the compiler and used for documentation purposes.

The lexical analyzer should follow these steps:

[1] Assignment idea courtesy Steve Carr @ WMU-CS

1. Read the input source code file line by line and store it in a buffer.
2. Scan the buffer from left to right and identify the tokens based on the language specification.
3. For each token, create a token object with the token type and the token value as attributes.
4. Append the token object to a list of tokens.
5. Repeat steps 2 to 4 until the end of the buffer is reached or an error is encountered.
6. Return the list of tokens as output or display an error message if an error is encountered.

Example Input:

An example input source code snippet written in Cminus could look like this:

```
/* This is a multi-line

           comment */

int c;

if (c == d) { IF = a; }
```

Example Output:

An example output list of tokens produced by the lexical analyzer could look like this:

```
[(COMMENT,' ….'), (KEYWORD, int), (IDENTIFIER, c), (SEPERATOR, ;),
(KEYWORD, if), (SEPARATOR, (), (IDENTIFIER, c), (LOGIC-OP, ==),
(IDENTIFIER, d), (SEPERATOR, )), (SEPERATOR, {), (IDENTIFIER, IF),
(ARITH-OP, =), (IDENTIFIER, a), (SEPERATOR, ;), (SEPERATOR, })]
```

Congratulations! You're well on your way to becoming the next internet search pioneer! 🔍

Notes

References

- ChatGPT-like tools
- GeeksforGeeks - https://www.geeksforgeeks.org/introduction-of-lexical-analysis/#
- Lex - https://en.wikipedia.org/wiki/Lex_(software)
- Wiki - https://en.wikipedia.org/wiki/Lexical_analysis
- flex, bison, jflex, ply

If you have creative ideas for extensions or making it more interesting, run them by the course staff, and we'd be happy to give you guidance!

Design Requirements

Code Documentation

For this assignment, you must properly document your code and use good software development practices.

Github

Use github to store your repository. Use good revision-cotrol-system practices as you develop various pieces of the search engine.

Testing

Make sure you test your application with several different values capturing different cases, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
  - Source code files
  - any input or output files
- Don’t forget to follow the naming convention specified for submitting assignments
- You will also show execution of your application to grader / instructor. They may give you a test case or two on the spot.