CS 4013/CS 6413: Compiler Construction

Project 1: Lexical Analyzer
(Due: September 23, 2022)

For the overall class project, each student is required to (independently) develop the front end of a compiler for the Pascal language (see pp. 745-751 of the OLD Dragon Book).

The first phase of the compiler project involves the design and implementation of the lexical analyzer. The lexical analyzer may be written in a language of your choice (C/C++, Java, Python, etc.). Students interested in pursuing careers in compiler construction, code analysis and reverse engineering or cyber operations should program their projects in C.

The analyzer should be designed as a separate program that will be invoked by the parser (Project 2). It should read each line of the source program (with no more than 72 characters per line) into a 72 character buffer, generate tokens, produce a listing file with line numbers, and detect and print the appropriate error messages.

It is also necessary to design and implement a simple symbol table. Assume that all special words in the language are reserved words. They should be listed in a file which is read and loaded into a reserved word table. When an identifier is encountered, the reserved word table is scanned. It is only when the identifier is not in the reserved word table and is not already present in the symbol table that it is added to the symbol table.

Start working on the project right away. You will do much better in Exam 1 if you have the project done on time. Also, please remember to document your learnings and project-related activities in your journal.

Special Restrictions

You must observe the following restrictions when designing and implementing your lexical analyzer.

Buffer Size: 72 chars

Identifier length: 10 chars (Max)

Integers: 10 digits (Max)

Reals: xx.yyEzz xx: 5 digits (Max)
yy: 5 digits (Max)
zz: 2 digits (Max)

Documentation

The project report should be organized as follows:

Cover Page
Introduction
Methodology
Implementation
Discussion and Conclusions
References
Appendix I: Sample Inputs and Outputs
Appendix II: Program Listings

Input Files

1. Source Program: You are required to run and submit the output for two source programs. The first program should have NO lexical errors. The second program should contain EVERY lexical error that your scanner is capable of trapping.

2. Reserved Word File: This file is read in during the initialization process and its information is stored in the reserved word table. The file should contain the following information:

   "reserved word lexeme" TOKEN-TYPE ATTRIBUTE

   TOKEN-TYPE and ATTRIBUTE would most likely be integers.

Output Files

1. Listing File: The listing file should have line numbers. Lexical errors should be printed under the corresponding line of code. When multiple errors are present, each error should be printed on a new line. You are required to turn in two listing files corresponding to the two source programs.

   Sample Listing File:

```
 1         program test #@
 LEXERR:   Unrecognized Symbol:         #
 LEXERR:   Unrecognized Symbol:         @
 2
 3         12345678901  44.444444.
 LEXERR:   Extra Long Integer:          12345678901
 LEXERR:   Extra Long Fractional Part:  44.444444
```

2. Token File: This file has each (TOKEN-TYPE, ATTRIBUTE) ordered pair printed on a new line along with the Lexeme and the Line No.

in the source program. Remember to print all erroneous lexemes and their corresponding information. Note that the last token in the file must be the end of file (EOF) token.

Sample Token File:

Line No.	Lexeme	TOKEN-TYPE	ATTRIBUTE
1	program	10 (PROG)	0 (NULL)
1	test	20 (ID)	loc2 (ptr to sym tab)
1	#	99 (LEXERR)	1 (Unrecog Symbol)
1	@	99 (LEXERR)	1 (Unrecog Symbol)
3	12345678901	99 (LEXERR)	2 (Extra Long Int)
3	44.444444	99 (LEXERR)	4 (Extra Long Frac)
3	.	88 (PERIOD)	0 (NULL)
	EOF	40 (EOF)	0 (NULL)

3. Symbol Table: (Optional)