CSE 304-January 2017

Assignment 4

Department of CSE, MIST — March 14, 2017

1 Introduction

In this assignment we are going to construct a lexical analyzer. Lexical analysis is the process of scanning the source program as a sequence of characters and converting them into sequence of tokens. A program that performs this task is called a lexical analyzer or a lexer or a scanner. For example if a portion of source program contains *int x=5;*, the scanner would convert in a sequence of tokens like <INT><ID,x><ASSIGNOP,=><COST_NUM,5><SEMICOLON>.

After successfully(!) completing construction of a very simple symbol table, we will construct a scanner for a subset of **C** language. The task will be performed using a tool named *flex*(*Fast Lexical Analyzer Generator*) which is a popular tool for generating scanners.

2 Tasks

You have to complete the following tasks in this assignment.

2.1 Identifying Tokens

2.1.1 Keywords

You have to identify lexeme of the keywords given in Table 1 and print token <Keyword_name> in the output file. For example, you will print <IF> in case you find the keyword "if" in source program. Keywords will not be inserted in the symbol table.

2.1.2 Constants

For each constant you have to print a token in <CONST_type,Symbol> in the output file and insert the symbol in symbol table.

- **Integer Literals:** One or more consecutive digits form an integer literal. Token name will be CONST_INT.
- **Floating Point Literals:** Numbers like 3.14159, 3.14159E-10 and 314159E10 will be considered as floating point constants. In this case, token name will be CONST_FLOAT.

Keyword	Token	Keyword	Token
if	<IF>	else	<ELSE>
for	<FOR>	while	<WHILE>
do	<DO>	break	<BREAK>
int	<INT>	char	<CHAR>
float	<FLOAT>	double	<DOUBLE>
void	<VOID>	return	<RETURN>
switch	<SWITCH>	case	<CASE>
default	<DEFAULT>	continue	<CONTINUE>

Table 1: Keyword List

- **Character Literals:** Character literals are enclosed in single quotes. There will be a single character within the single quotes with the exception of ‘\n’, ‘\t’, ‘\\’, ‘\“’, ‘\a’, ‘\f’, ‘\r’, ‘\b’, ‘\v’ and ‘\0’. For character literals token name will be CONST_CHAR.

2.1.3 Operators and Punctuators

The operator list for the subset of C program we are dealing with is given in Table 2. A token in the form of <TOKEN_NAME,Symbol> should be printed in the output file and the operator should be inserted in the symbol table.

Operators	Token_Name
+, -	ADDOP
*, /, %	MULOP
++, –	INCOP
<, <=, >, >=, ==, !=	RELOP
=	ASSIGNOP
&&, ||, !	LOGICOP
(,)	PAREN
{, }	CURL
[,]	BRACKET
,	COMMA
;	SEMICOLON

Table 2: Operator and Punctuators List

2.1.4 Identifiers

Identifiers are names given to C entities, such as variables, functions, structures etc. An identifier can only have alphanumeric characters(a-z , A-Z , 0-9) and underscore(_). The first character of an identifier can only contain alphabet(a-z , A-Z) or underscore (_). For any identifier encountered in the input file you have to print token <ID,Symbol> and also insert it in the symbol table.

2.1.5 Strings

String literals or constants are enclosed in double quotes “". String can be single line or multi line. A multi line string is ended with a \character in each line except the last line. You have to print a token like <STRING, “abc"> if you find a string “abc" in the input file. String will not be inserted in the symbol table.

2.1.6 Comments

Comments can be single lined or multiple lined. A single line comment usually statr with // symbols. A multiline comment starts with /* and terminate with the characters */. The multiline version can also serve as a single line comment. If there is any comment in input file you have to recognize it but not generate any token in output file.

2.1.7 White Space

You have to ignore all the white spaces in the input file.

2.2 Line Count

You should count the number of lines in the source program.

2.3 Lexical Error

You should detect lexical errors in the source program and report it along with line number. You have to detect following type of errors.

- Too many decimal point error for character sequence like 1.2.345
- Ill formed number such that 1E10.7
- Invalid Suffix on numeric constant or invalid prefix on identifier for character sequence like 12abcd;
- Multi character constant error for character sequence like ‘ab’
- Unfinished string
- Unfinished comment
- Unrecognized character

Also count the number of errors.

3 Input

The input will be a text file containing a C source program. File name will be given from command line.

4 Output

In this assignment, there will be two output file. One is a file containing tokens. This file should be named as <YourStudentID>_token (For example 201514091_token.txt). You will output all the tokens in this file.

The other file is a log file named as <YourStudentID>_log.txt (For example 201514091_log.txt). In this file you will output all the actions performed in your program. For example, after detecting any lexeme except one representing white spaces you will print a line containing " Line no <line_count>: Token <TokenName> Lexeme <Lexeme> found". For example if you find a comment "//abcd" at line no 5 in your source code you will print " Line no 5: Token <Comment> Lexeme <abcd> found". Note that, although you will not print any token in corresponding token.txt file for comment, you will print it in log file. For any insertion into symbol table, you will print the symbol table in the output file (only print the non empty buckets). If symbol already exists print appropriate message. For any detected error print "Line no 5: Corresponding error message". Print the line count and no of errors at the end of log file.

For more clarification about input output please refer to the sample input output file supplied here with. You are highly encouraged to produce output exactly like the sample one.

5 Rules

- Any type of plagiarism is strongly forbidden and may result in severe punishments.
- Prepare for an online evaluation.

Acknowledgement

The author wants to thank Tanvir Ahmed Khan, Lecturer, CSE, BUET.