LEXICAL ANALYZER IN PERL Introduction to language processing assingnments 1

AUTHORS


```
SUDHIR SHARMA   {12041500} cse 2020
MANISH MARUTI SAHLUNKHE {12040840}  cse 2020
```


Instructions

- Input have been taken through files.
- We have tested our lexical analyser using testcases stored in file.
- If you wanted to change the input file (test file)
    - Go to line number `354` in the lexer.l file.
    - `yyin = fopen("test4.pl","r"); //add test case files here` change the file name.
- Output will be in Hashed Table format, will be displayed in your terminal.
- Comments have been shown as `Line comments`.
- KINDLY NOTE THAT

```
Present Line you are reading in the terminal = current line no. in the test case file
- no of "Line comments" you see above .
```

Using Flex code:

```
flex lexer.l
gcc lex.yy.c
./a.out
```

Lexical Analysis

- We begin the study of lexical-analyzer generators by introducing regular expressions.
- We show how this notation can be transformed,
    - first into nondeterministic automata and then into deterministic automata.
- The latter two notations can be used as input to a "driver",
    - that is, code which simulates these automata and uses them as a guide to determining the next token.
- This driver and the specification of the automaton form the nucleus of the lexical analyzer.

The Role of the Lexical Analyzer

lexical analyzer may perform certain other tasks besides identification of lexemes.

- stripping out comments and *whitespace*
- correlating error messages generated by the compiler with the source program.

In some compilers, the lexical analyzer makes a copy of the source program with the error messages inserted at the appropriate positions. If the source program uses a macro-preprocessor, the expansion of macros may also be performed by the lexical analyzer.

Sometimes, lexical analyzers are divided into a cascade of two processes:

- a) *Scanning* consists of the simple processes that do not require tokenization of the input,
  - such as deletion of comments and compaction of consecutive whitespace characters into one.
- b) *Lexical analysis* proper is the more complex portion, where the scanner produces the sequence of tokens as output.

What We Have Done

- Our lexical analyser contain lexer.l file in which we have written the lex code, 5-6 test cases file for testing.
- As we have good knowledge of C/C++. So we have use the techineque of mapping and Hash .We just define all the Identifiers ,Tokens,Keywords and mappped them with natural number, also we have define String Constant ,Char constant , digits,strings, and many more using Regular expression .
- We have taken the input as a test file using `yyin = fopen("test4.pl","r")` .
- Also used `can, slcline=0, mlc=0, mlcline=0, dq=0, dqline=0;` as a pointer to iterate over the input file .
- If the Keywords, tokens etc get matched, we just returned that identifiers and the line no. using `yytext, lineno` .
- The output will be in tabular format.
- There are much more we can do we haven't defined whole keywords we have tried to dfined as much as keywords ,predefined function

Quick Overview of Perl

- Perl Identifiers
  - A Perl variable name starts with either $, @ or % followed by zero or more letters, underscores, and digits (0 to 9).
- Three basic data type
  - `Scalar` Scalars are simple variables. They are preceded by a dollar sign ($). A scalar is either a number, a string, or a reference.
  - `Arrays` Arrays are ordered lists of scalars that you access with a numeric index, which starts with 0. They are preceded by an "at" sign (@).
  - `Hashes` Hashes are unordered sets of key/value pairs that you access using the keys as subscripts. They are preceded by a percent sign (%).

```
We have defined these three data types seperately in the lexer.l file.
```

References :

-Kumar Shivendhu

- Geeksforgeeks
- `info flex`
- YouTube
- notes