

Ex No: 2

Date:

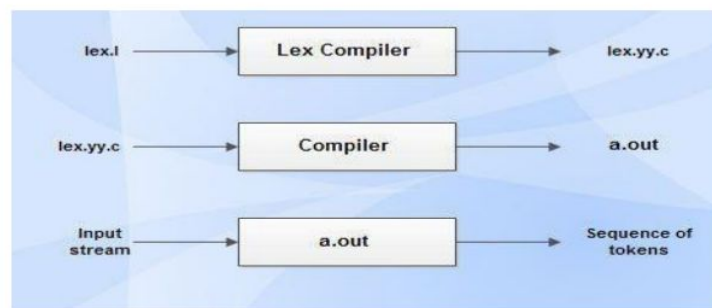
IMPLEMENT A LEXICAL ANALYZER TO COUNT THE NUMBER OF WORDS USING LEX TOOL

AIM:

To implement the program to count the number of words in a string using LEX tool.

STUDY:

Lex is a tool in lexical analysis phase to recognize tokens using regular expression. Lex tool itself is a lex compiler.



- lex.l is an input file written in a language which describes the generation of lexical analyzer. The lex compiler transforms lex.l to a C program known as lex.yy.c.
- lex.yy.c is compiled by the C compiler to a file called a.out.
- The output of C compiler is the working lexical analyzer which takes stream of input characters and produces a stream of tokens.
- yylval is a global variable which is shared by lexical analyzer and parser to return the name and an attribute value of token.
- The attribute value can be numeric code, pointer to symbol table or nothing.
- Another tool for lexical analyzer generation is Flex.

STRUCTURE OF LEX PROGRAMS:

Lex program will be in following form

declarations

%%

translation rules

%%

auxiliary functions

ALGORITHM

1. Declare necessary header files and variables in the beginning.
2. Define rules in the form of regular expressions to identify words and newline characters.
3. Increment a counter each time a word is matched.
4. Reset the counter when encountering a newline character and print the count.
5. Implement the main function to initiate lexical analysis and return 0.

PROGRAM

```
% {
#include<stdio.h>
#include<string.h>
int i = 0;
% }

/* Rules Section*/
%%
([a-zA-Z0-9])* {i++;} /* Rule for counting
                    number of words*/

"\n" {printf("%d\n", i); i = 0;}
%%

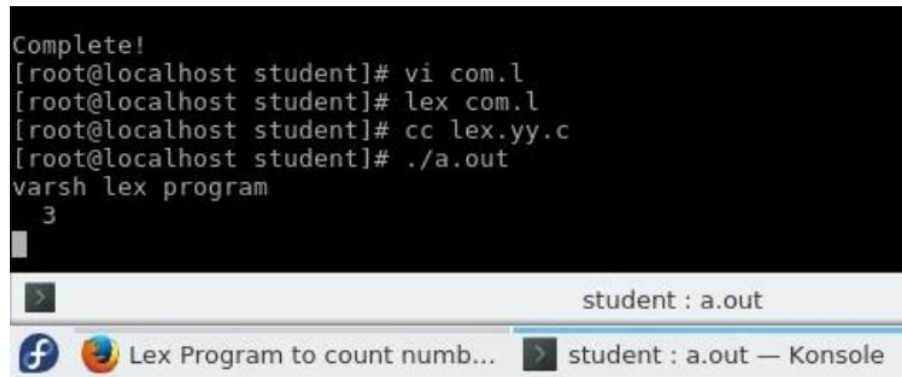
int yywrap(void){ }

int main()
{
    // The function that starts the analysis
    yylex();

    return 0;
```

OUTPUT:

```
Complete!
[root@localhost student]# vi com.l
[root@localhost student]# lex com.l
[root@localhost student]# cc lex.yy.c
[root@localhost student]# ./a.out
varsh lex program
3
```

**RESULT**