

**Ex No: 2**

**Date:**

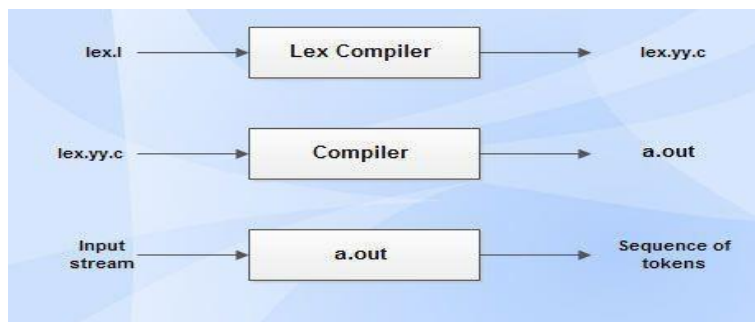
## **IMPLEMENT A LEXICAL ANALYZER TO COUNT THE NUMBER OF WORDS USING LEX TOOL**

### **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 a 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.
- yyval 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:

- Define tokens `let` and `dig` using `%token` directive and lexical rules in `yylex()` to recognize them.
- Define grammar rules in BNF form for `sad` and `recl` in the Bison specification.
- Implement semantic actions to print "accepted" for valid inputs and "rejected" for errors.
- In the `main()` function, call `yyparse()` to initiate parsing and prompt user input with "Enter a variable : ".
- During execution, the program scans input, applies grammar rules, and executes semantic actions.
- Handle errors by triggering the `error` rule and calling `yyerror()` to print "rejected" and exit.

### PROGRAM:

```
% {
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
% }

%token let dig

%%

sad : let recl {printf("accepted\n"); exit(0);}
    | let 'n' {printf("accepted\n"); exit(0);}
    |
    |error {yyerror("rejected\n");exit(0);}
;

recl : let recl
    | dig recl
```

```
| let
| dig
;
%%
yylex(){
char ch;
while((ch=getchar())!=' ');
if(isalpha(ch))
return let;
if(isdigit(ch))
return dig;
return ch;
}
yyerror(char *s){
printf("%s\n",s);
exit(0);
}
main(){
printf("Enter a variable : ");
yyparse();
}
```

**OUTPUT:**

```
[root@localhost student]# vi 281t.l
[root@localhost student]# lex 281t.l
[root@localhost student]# cc lex.yy.c
[root@localhost student]# ./a.out
hi this is tamanna from Cse-E
No. of words in a string : 7
```

**RESULT:**