Name: Shardul Vikram Dharmadhikari
Roll no: 8016

ASSIGNMENT_5

Title:

Write a program using Lex specifications to implement lexical analysis phase of compiler to count no. of words, lines and characters of given input file.

Objectives:

- Understand the implementation of the Lexical analyzer
- To understand LEX & YACC concepts
- To implement LEX & YACC programs
- To study about lex and yacc specifications

THEORY :

Structure of a Lex Specification:-

A lex program consists of three parts: the definition section, the rules section, and the user subroutines.

...definition section ...

%%

... rules section ...

%%

... user subroutines ...

The parts are separated by lines consisting of two percent signs. The first two parts are required, although a part may be empty. The third part and the

preceding %%line may be omitted. (This structure is the same as that used by yacc , from which it was copied.)

Definition Section:-

The definition section can include the literal block, definitions, internal table declarations, start conditions, and translations. (There is a section on each in this reference.) Lines that start with whitespace are copied verbatim to the C file.

Typically this is used to include comments enclosed in "/*" and "*/", preceded by white space.

Rules Section:-

The rules section contains pattern lines and C code. A line that starts with white space, or material enclosed in "%{" and "%}" is C code. A line that starts with anything else is a pattern line.

Variables in lex Program:-

1)Yytext:- whenever the scanner matches a token, the text of the token is stored in thenull terminated string yytext.

2) yyleng:- The length of the string yytext.

3) yylex():- The scanner created by the Lex has the entry point yylex()

When you call yylex() to start or resume scanning. if lex action does a return to pass a value to the calling program, the next call to yylex() will continue from the point where it left off

The compiled lexical analyzer performs the following functions:

a) Reads an input stream of characters.

b) Copies the input stream to an output stream.

c) Breaks the input stream into smaller strings that match the extended regular expressions in the lex specification file.

d) Executes an action for each extended regular expression that it recognizes. These actions are C language program fragments in the lex specification file. Each action fragment can call actions or subroutines outside of itself.

Compiling the lexical analyzer:-

To compile a lex program, do the following:

1. Use the lex program to change the specification file into a C language program. The resulting program is in the lex.yy.c file.
2. Use the cc command with the -ll flag to compile and link the program with a library of lex subroutines. The resulting executable program is in the a.out file.
3. For example, if the lex specification file is called lextest, enter the following commands:

Lex lextest

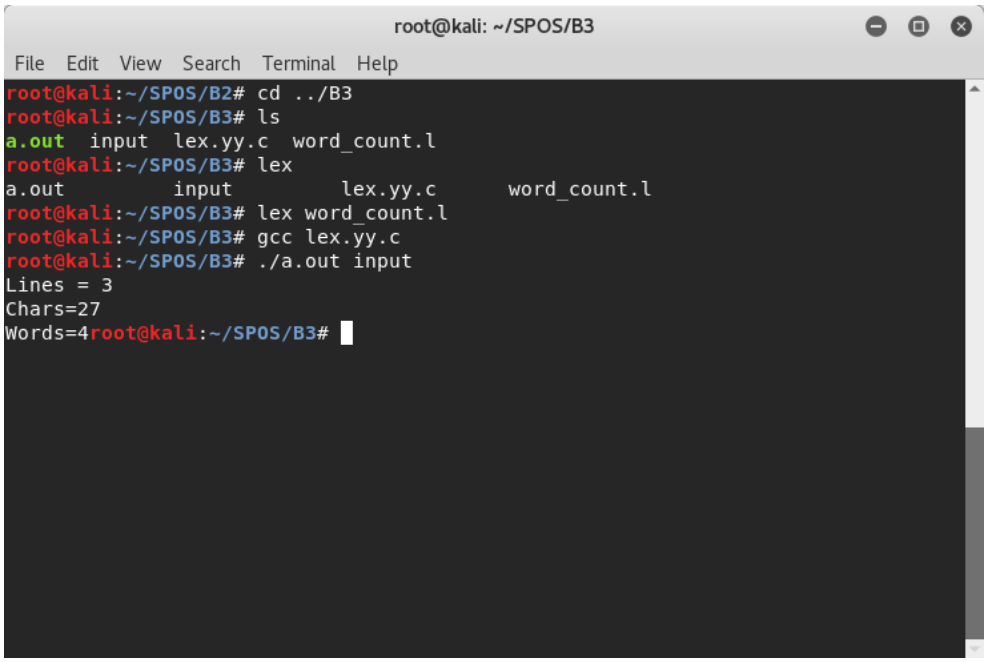
cc lex.yy.c –ll

CODE:

```
/*definition or declaration*/
%{
	#include<stdio.h>
	FILE *fp;
	int
line_cnt=0,space_cnt=0,words_cnt=0,char_cnt=0,num_cnt=0,digit_count=0,special_char_cnt=0
;
```

```
%}



/*Tokenization*/
new_line [\n]
words [a-zA-Z]+
space [\t|' ']
digit [0-9]+
special_char ['+'|'-'|'*'|'&'|'^'|'%'|'/']
/*Rules*/
%%
{new_line} {line_cnt++;char_cnt+=yyleng;}
{words} {words_cnt++;char_cnt+=yyleng;}
{space} {space_cnt++;char_cnt+=yyleng;}
{digit} {num_cnt++;char_cnt+=yyleng;digit_count+=yyleng;}
{special_char} {special_char_cnt++;char_cnt+=yyleng;}
%%


/*Main Function*/
int main(int argc,char *argv[])
{
	fp=fopen(argv[1],"r");
	yyin=fp;
	yylex();

	printf("\nTotal number of lines : %d",line_cnt);
	printf("\nTotal number of words : %d",words_cnt);
	printf("\nTotal number of space : %d",space_cnt);
	printf("\nTotal numbers in the file : %d",num_cnt);
	printf("\nTotal number of digits : %d",digit_count);
	printf("\nTotal number of characters : %d",char_cnt);
	printf("\nTotal number of special characters : %d\n",special_char_cnt);
	return 0;
}
```

OUTPUT:



CONCLUSION:

Thus I have studied lexical analyzer, syntax analysis and implemented lex and yacc application for syntax analyzer to validate the given infix expression