Experiment II

**Aim**: Implementation of lexical analyzer using lex tool.

Algorithm

1. Start
2. Read the file name of the program: `filename`.
3. Open the file `filename` in read mode.
4. If the file does not exist or the user does not have the permission to open it, goto step 6.
5. Compare each pattern `p` in the program with the defined regular expression.
   1. If `p` is a preprocessor directive or a single line comment or a double line comment, skip the pattern.
   2. If `p` is a keyword, replace it with ‘kwd’.
   3. If `p` is an identifier, replace it with ‘id’.
   4. If `p` is a string or integer or floating point literal, print it as it is.
   5. If `p` is an operator, replace it with the corresponding token.
   6. If `p` is ‘,’ or ‘;’, replace it with the same character.
   7. If `p` does not match any regular expressions, ignore and move on to read the next line.
6. Close the file.
7. Stop

Input file

program.c

// Program to calculate the sum of two numbers

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void main() {

/\* This is a

multiline comment \*/

int num1, num2, sum;

num1 = 10;

num2 = 20;

sum = num1 + num2;

}

Output

Enter the file name of the program: program.c

kwd id, id, id;

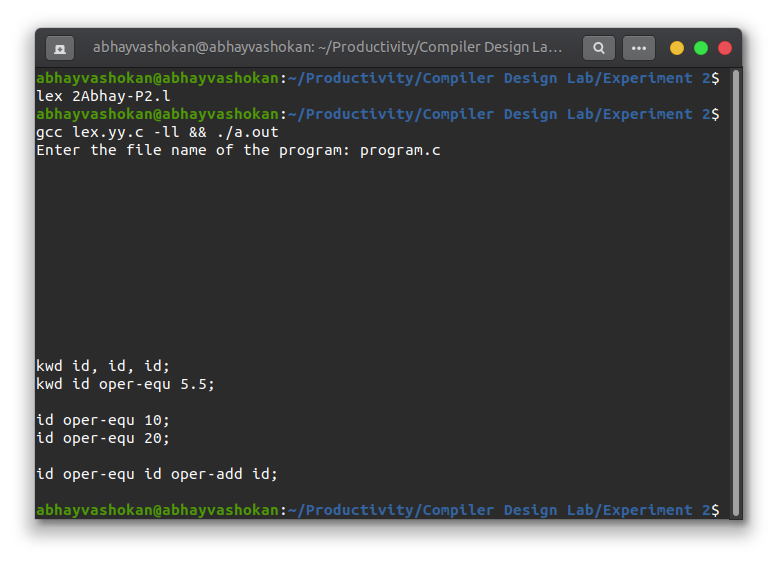
kwd id oper-equ 5.5;

id oper-equ 10;

id oper-equ 20;

id oper-equ id oper-add id;

Screenshot



Readme

1. Compile the lex program using the command

**lex 2Abhay-P2.l**

2. Now compile and run the **lex.yy.c** file generated using the command

**gcc lex.yy.c -ll && ./a.out**

3. Input the correct path of the file to be analysed.

3. The stream of tokens obtained is displayed in the terminal.

**Result**: Successfully implemented lexical analyzer using lex tool.