# Lab 2

## Lab 2 Output Description

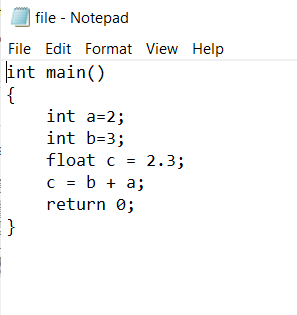
The scanner converts the high-level input program into a sequence of tokens. The output is a sequence of tokens that is sent to the parser for syntax analysis. In Flex:

1. An input file describes the lexical analyzer to be generated named lex.l is written in lex language. The lex compiler transforms lex.l to C program, in a file that is always named lex.yy.c.
2. The C compiler compiles lex.yy.c file into an executable file called a.exe
3. The output file a.exe take a stream of input characters and produce a stream of tokens.

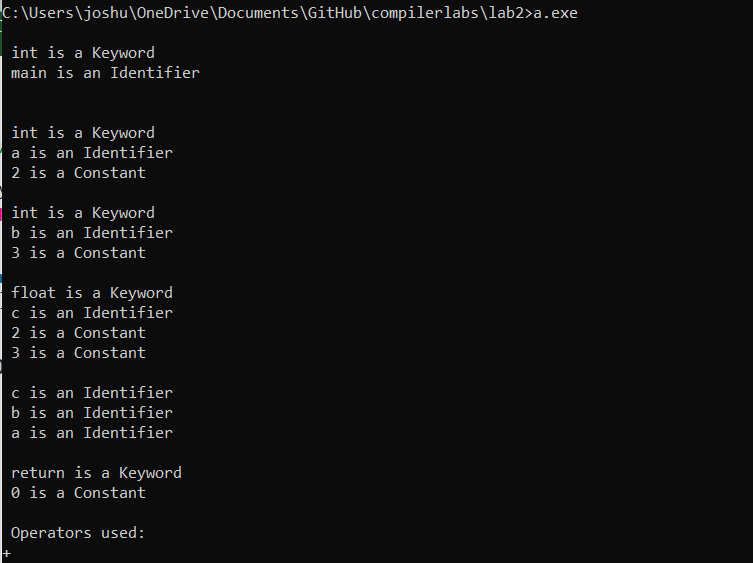
The Lexical analyzer:

1. Divides the program into valid tokens (Tokenization)
2. Removes white space characters.
3. Removes comments.
4. It also provides help in generating error messages by providing row numbers and column numbers.

## Text file



## Lab 2 Output



**From the text file, all the valid tokens are:**

‘int’ , ‘main’ , ‘(’ , ‘)’ , ‘{’ , ‘int’ , ‘a’ , ‘=’ , ‘2’ , ‘;’ , ‘int’ , ‘b’ , ‘=’ , ‘3’ , ‘;’ , ‘float’ , ‘c’ , ‘=’ , ‘2.3’ , ‘;’ , ‘c’ , ‘=’ , ‘b’ , ‘+’ , ‘a’ , ‘;’ , ‘return’ , ‘0’ , ‘;’ , ‘}’

Total tokens: 30

**The expression is then tokenized using the input file source program where:**

int is a keyword, main is an identifier

int is a keyword, a is an identifier, 2 is a constant

int is a keyword, b is an identifier, 3 is a constant

float is a keyword, c is an identifier, 2 is a constant, 3 is a constant

c is an identifier, b is an identifier, a is an identifier

return is a keyword, 0 is a constant

+ is an operator

The tokenized output is then displayed.