Practical - 1

2CS701 – Compiler Construction

Harshit Gajipara
19BCE059

Aim:

To implement lexical analyse to recognize all distinct token classes: use flex/lex tool to recognize all distinct token classes (Data type, Identifier, constant (Integer, Float, Char, String), Operator (Arithmetic, Relational, Assign, Unary +/-, Increment), Single line/Multi-line comments, Special symbol (;,{}())) .

Generate Lexical error reports for invalid lexeme.

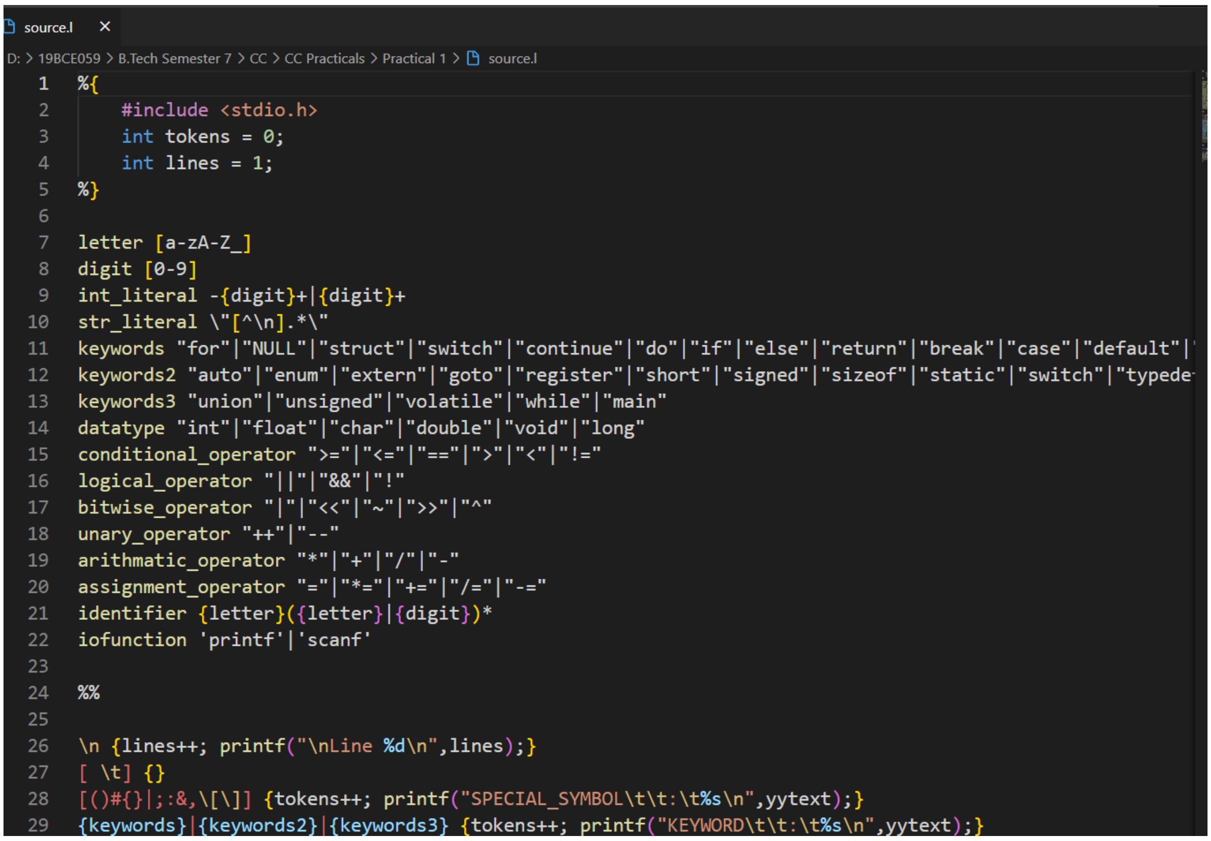
Code:

```
%{
	#include <stdio.h>
	int tokens = 0;
	int lines = 1;
%}

letter [a-zA-Z_]
digit [0-9]
int_literal -{digit}+|{digit}+
str_literal \"[^\n].*\"
keywords
"for"|"NULL"|"struct"|"switch"|"continue"|"do"|"if"|"else"|"return
"|"break"|"case"|"default"|"const"
keywords2
"auto"|"enum"|"extern"|"goto"|"register"|"short"|"signed"|"sizeof
"|"static"|"switch"|"typedef"
keywords3 "union"|"unsigned"|"volatile"|"while"|"main"
```

```
datatype "int"|"float"|"char"|"double"|"void"|"long"
conditional_operator ">="|"<="|"=="|">"|"<"|"!="
logical_operator "||"|"&&"|"!"
bitwise_operator "|"|"<<"|"~"|">>"|"^"
unary_operator "++"|"--"
arithmatic_operator "*"|"+"|"/"|"-"
assignment_operator "="|"*="|"+="|"/="|"-="
identifier {letter}({letter}|{digit})*
iofunction 'printf'|'scanf'

%%

\n {lines++; printf("\nLine %d\n",lines);}
[ \t] {}
[()#{}|;:&,\[\]] {tokens++;
printf("SPECIAL_SYMBOL\t\t:\t%s\n",yytext);}
{keywords}|{keywords2}|{keywords3} {tokens++;
printf("KEYWORD\t\t:\t%s\n",yytext);}
{arithmatic_operator} {tokens++;
printf("ARITHMATIC_OPERATOR\t:\t%s\n",yytext);}
{assignment_operator} {tokens++;
printf("ASSIGNMENT_OPERATOR\t:\t%s\n",yytext);}
{conditional_operator} {tokens++;
printf("CONDITIONAL_OPERATOR\t:\t%s\n",yytext);}
{unary_operator} {tokens++;
printf("UNARY_OPERATOR\t:\t%s\n",yytext);}
```

```
{datatype} {tokens++; printf("DATA_TYPE\t\t:\t%s\n",yytext);}

{identifier} {tokens++; printf("IDENTIFIER\t\t:\t%s\n",yytext);}

{int_literal} {tokens++;
printf("INT_LITERAL\t\t:\t%s\n",yytext);}

{int_literal}"."{int_literal} {tokens++;
printf("FLOAT_LITERAL\t\t:\t%s\n",yytext);}

{str_literal}
{tokens++;printf("STR_CONSTANT\t\t:\t%s\n",yytext);}

{iofunction} {tokens++;
printf("IO_FUNCTION\t\t:\t%s\n",yytext);}

\/\/[^\n].* {printf("COMMENT\t\t:\t%s\n",yytext);} //single
line comment

\/\*[^(*/)]*\*\/ {printf("COMMENT\t\t:\t%s\n",yytext);}
//multi line comment


%%


int yywrap(){}


int main(){

printf("Line 1\n");

yylex();

printf("\n\nTotal tokens = %d",tokens);

return 0;

}
```

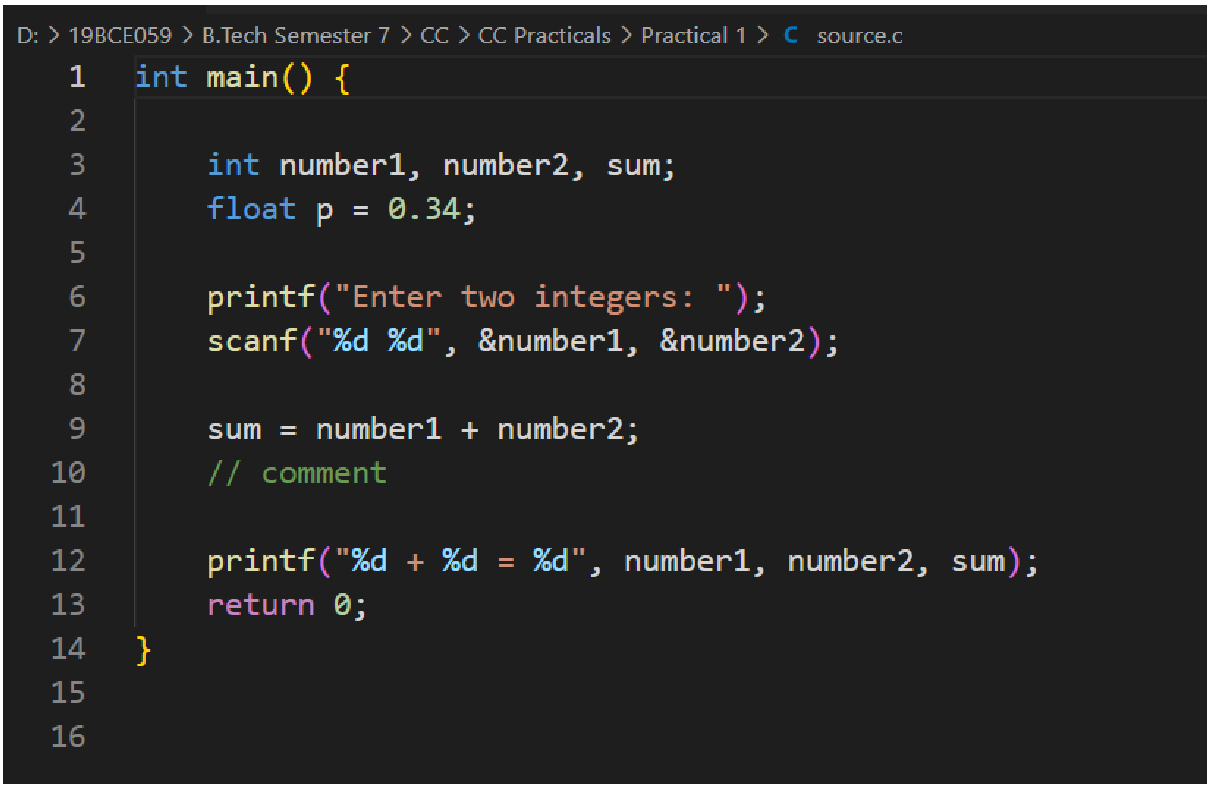
Output:



source.l

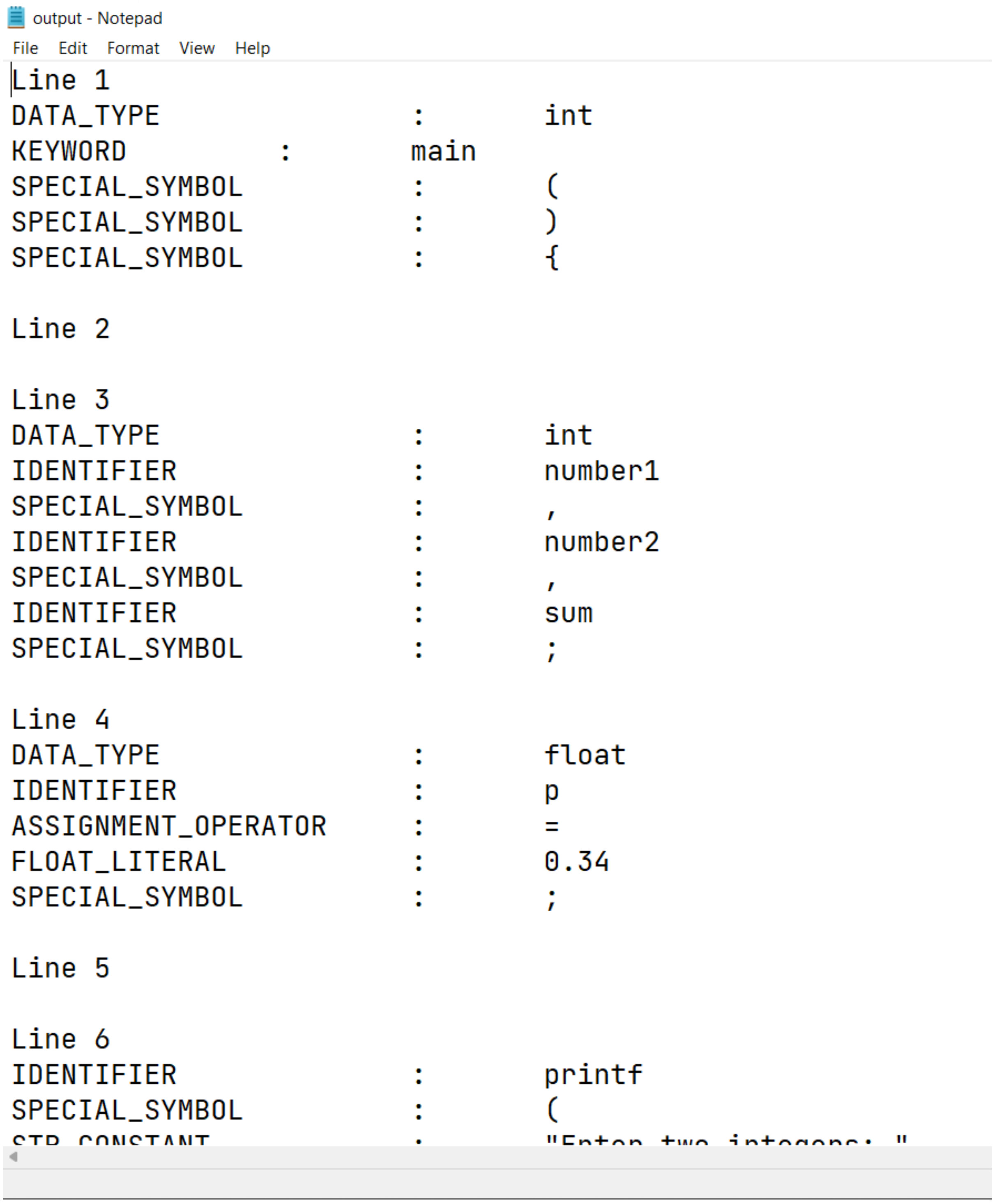
D: > 19BCE059 > B.Tech Semester 7 > CC > CC Practicals > Practical 1 > source.l

```
%{
    #include <stdio.h>
    int tokens = 0;
    int lines = 1;
%}

letter [a-zA-Z_]
digit [0-9]
int_literal -{digit}+|{digit}+
str_literal \"[^\n].*\"
keywords "for"|"NULL"|"struct"|"switch"|"continue"|"do"|"if"|"else"|"return"|"break"|"case"|"default"|
keywords2 "auto"|"enum"|"extern"|"goto"|"register"|"short"|"signed"|"sizeof"|"static"|"switch"|"typede
keywords3 "union"|"unsigned"|"volatile"|"while"|"main"
datatype "int"|"float"|"char"|"double"|"void"|"long"
conditional_operator ">="|"<="|"=="|">"|"<"|"!="
logical_operator "||"|"&&"|"!"
bitwise_operator "|"|"<<"|"~"|">>"|"^"
unary_operator "++"|"--"
arithmatic_operator "*"|"+"|"/"|"-"
assignment_operator "="|"*="|"+="|"/="|"-="
identifier {letter}({letter}|{digit})*
iofunction 'printf'|'scanf'

%%

\n {lines++; printf("\nLine %d\n",lines);}
[ \t] {}
[()#{}|;:&,\[\]] {tokens++; printf("SPECIAL_SYMBOL\t\t:\t%s\n",yytext);}
{keywords}|{keywords2}|{keywords3} {tokens++; printf("KEYWORD\t\t:\t%s\n",yytext);}
```

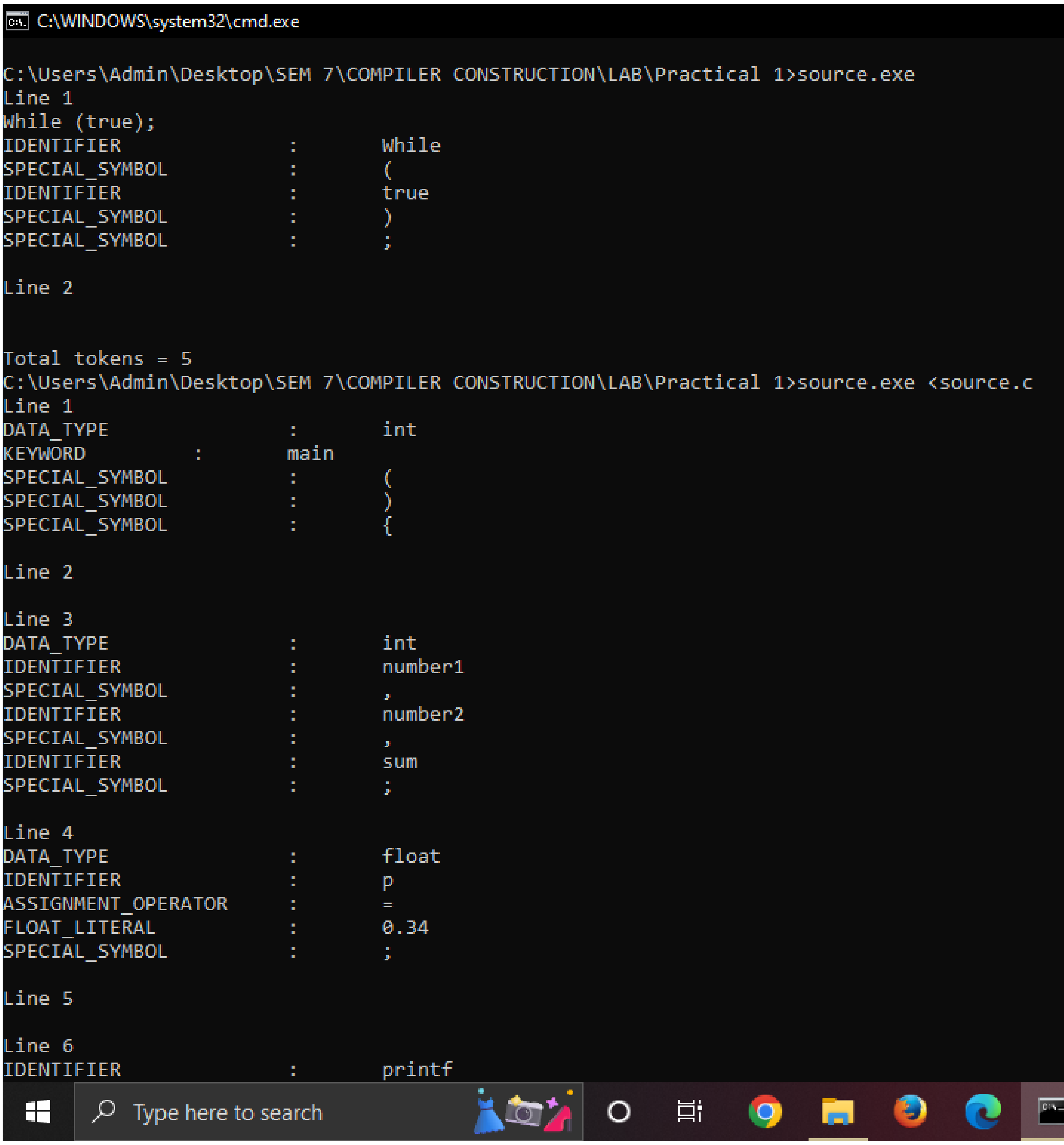


D: > 19BCE059 > B.Tech Semester 7 > CC > CC Practicals > Practical 1 > source.c

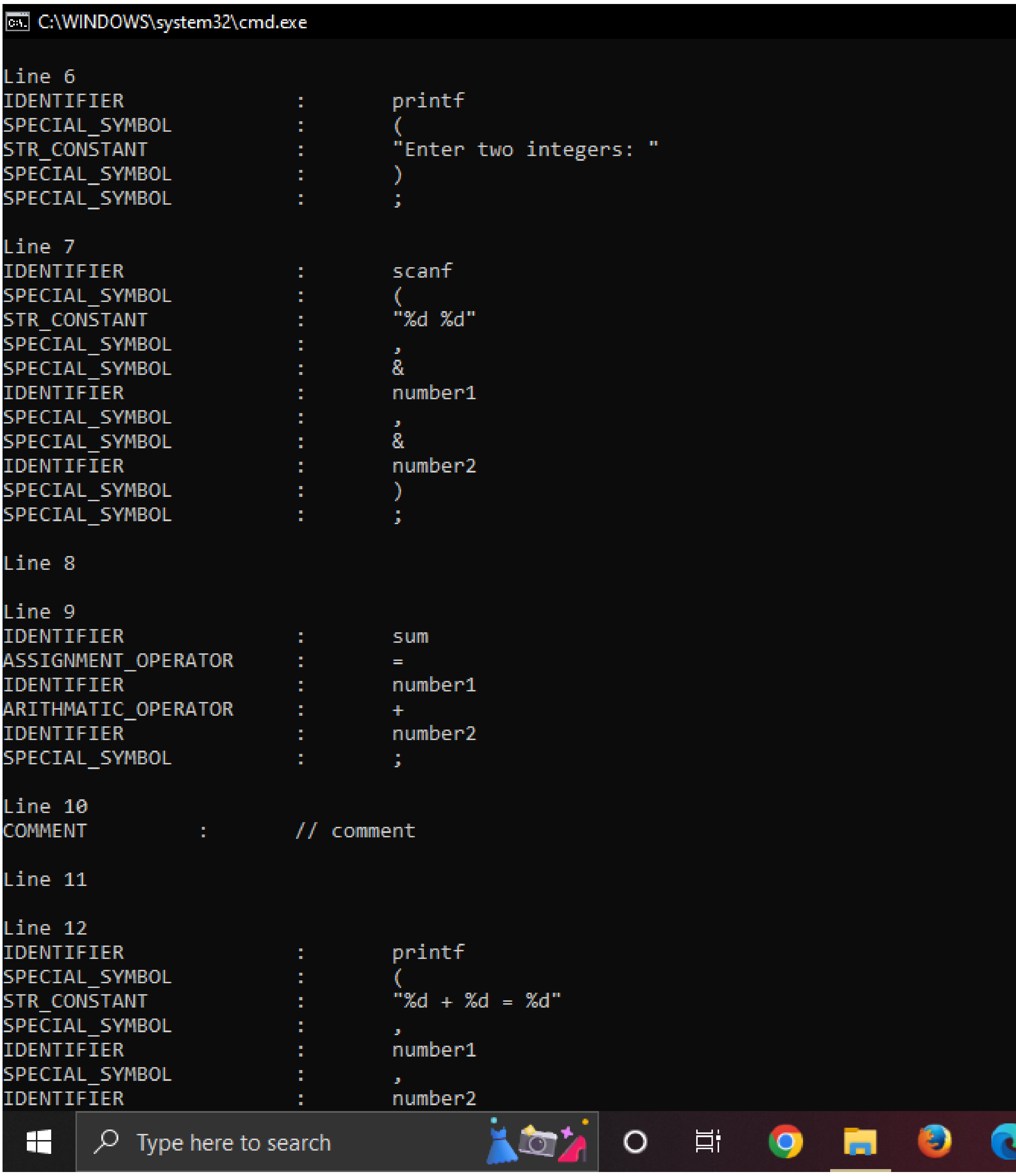
```c
int main() {

    int number1, number2, sum;
    float p = 0.34;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    sum = number1 + number2;
    // comment

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}


```



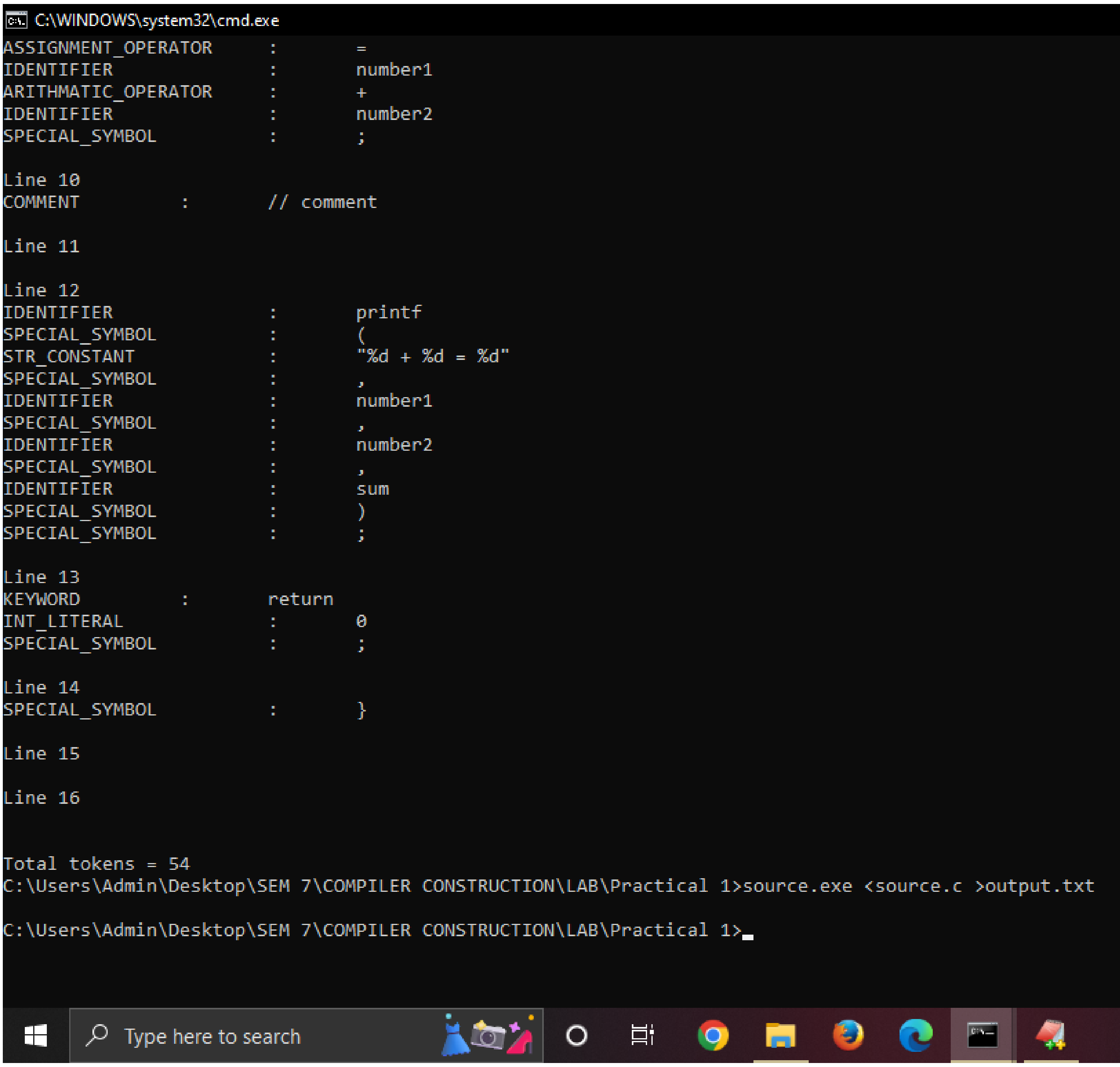
```
Line 1
DATA_TYPE                   :       int
KEYWORD         :       main
SPECIAL_SYMBOL              :       (
SPECIAL_SYMBOL              :       )
SPECIAL_SYMBOL              :       {

Line 2

Line 3
DATA_TYPE                   :       int
IDENTIFIER                  :       number1
SPECIAL_SYMBOL              :       ,
IDENTIFIER                  :       number2
SPECIAL_SYMBOL              :       ,
IDENTIFIER                  :       sum
SPECIAL_SYMBOL              :       ;

Line 4
DATA_TYPE                   :       float
IDENTIFIER                  :       p
ASSIGNMENT_OPERATOR         :       =
FLOAT_LITERAL               :       0.34
SPECIAL_SYMBOL              :       ;

Line 5

Line 6
IDENTIFIER                  :       printf
SPECIAL_SYMBOL              :       (
STR_CONSTANT                :       "Enter two integers: "
```



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Admin\Desktop\SEM 7\COMPILER CONSTRUCTION\LAB\Practical 1>source.exe
Line 1
While (true);
IDENTIFIER                :       While
SPECIAL_SYMBOL            :       (
IDENTIFIER                :       true
SPECIAL_SYMBOL            :       )
SPECIAL_SYMBOL            :       ;

Line 2


Total tokens = 5
C:\Users\Admin\Desktop\SEM 7\COMPILER CONSTRUCTION\LAB\Practical 1>source.exe <source.c
Line 1
DATA_TYPE                 :       int
KEYWORD           :       main
SPECIAL_SYMBOL            :       (
SPECIAL_SYMBOL            :       )
SPECIAL_SYMBOL            :       {

Line 2

Line 3
DATA_TYPE                 :       int
IDENTIFIER                :       number1
SPECIAL_SYMBOL            :       ,
IDENTIFIER                :       number2
SPECIAL_SYMBOL            :       ,
IDENTIFIER                :       sum
SPECIAL_SYMBOL            :       ;

Line 4
DATA_TYPE                 :       float
IDENTIFIER                :       p
ASSIGNMENT_OPERATOR       :       =
FLOAT_LITERAL             :       0.34
SPECIAL_SYMBOL            :       ;

Line 5

Line 6
IDENTIFIER                :       printf
```



```
Line 6
IDENTIFIER                      :       printf
SPECIAL_SYMBOL                  :       (
STR_CONSTANT                    :       "Enter two integers: "
SPECIAL_SYMBOL                  :       )
SPECIAL_SYMBOL                  :       ;

Line 7
IDENTIFIER                      :       scanf
SPECIAL_SYMBOL                  :       (
STR_CONSTANT                    :       "%d %d"
SPECIAL_SYMBOL                  :       ,
SPECIAL_SYMBOL                  :       &
IDENTIFIER                      :       number1
SPECIAL_SYMBOL                  :       ,
SPECIAL_SYMBOL                  :       &
IDENTIFIER                      :       number2
SPECIAL_SYMBOL                  :       )
SPECIAL_SYMBOL                  :       ;

Line 8

Line 9
IDENTIFIER                      :       sum
ASSIGNMENT_OPERATOR             :       =
IDENTIFIER                      :       number1
ARITHMATIC_OPERATOR             :       +
IDENTIFIER                      :       number2
SPECIAL_SYMBOL                  :       ;

Line 10
COMMENT         :       // comment

Line 11

Line 12
IDENTIFIER                      :       printf
SPECIAL_SYMBOL                  :       (
STR_CONSTANT                    :       "%d + %d = %d"
SPECIAL_SYMBOL                  :       ,
IDENTIFIER                      :       number1
SPECIAL_SYMBOL                  :       ,
IDENTIFIER                      :       number2
```



C:\WINDOWS\system32\cmd.exe

```
ASSIGNMENT_OPERATOR     :       =
IDENTIFIER              :       number1
ARITHMATIC_OPERATOR     :       +
IDENTIFIER              :       number2
SPECIAL_SYMBOL          :       ;

Line 10
COMMENT         :       // comment

Line 11

Line 12
IDENTIFIER              :       printf
SPECIAL_SYMBOL          :       (
STR_CONSTANT            :       "%d + %d = %d"
SPECIAL_SYMBOL          :       ,
IDENTIFIER              :       number1
SPECIAL_SYMBOL          :       ,
IDENTIFIER              :       number2
SPECIAL_SYMBOL          :       ,
IDENTIFIER              :       sum
SPECIAL_SYMBOL          :       )
SPECIAL_SYMBOL          :       ;

Line 13
KEYWORD         :       return
INT_LITERAL             :       0
SPECIAL_SYMBOL          :       ;

Line 14
SPECIAL_SYMBOL          :       }

Line 15

Line 16


Total tokens = 54
C:\Users\Admin\Desktop\SEM 7\COMPILER CONSTRUCTION\LAB\Practical 1>source.exe <source.c >output.txt

C:\Users\Admin\Desktop\SEM 7\COMPILER CONSTRUCTION\LAB\Practical 1>_
```

Conclusion:

In this practical, we learnt that using lex tool we can generate tokens from the source file and pass it to next level in compiler. Also we can design our own tokens and language based on it using lex.