Experiment – 3

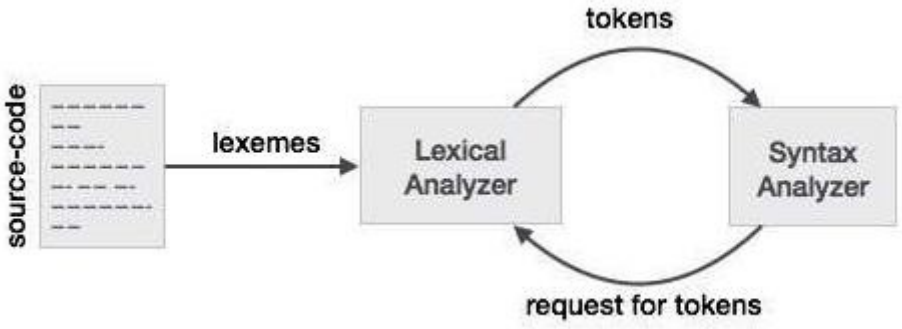
AIM- Write a program to implement simple lexical analyzer.

Description -

Lexical Analysis

Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.



Tokens

Lexemes are said to be a sequence of characters (alphanumeric) in a token. There are some predefined rules for every lexeme to be identified as a valid token. These rules are defined by grammar rules, by means of a pattern. A pattern explains what can be a token, and these patterns are defined by means of regular expressions.

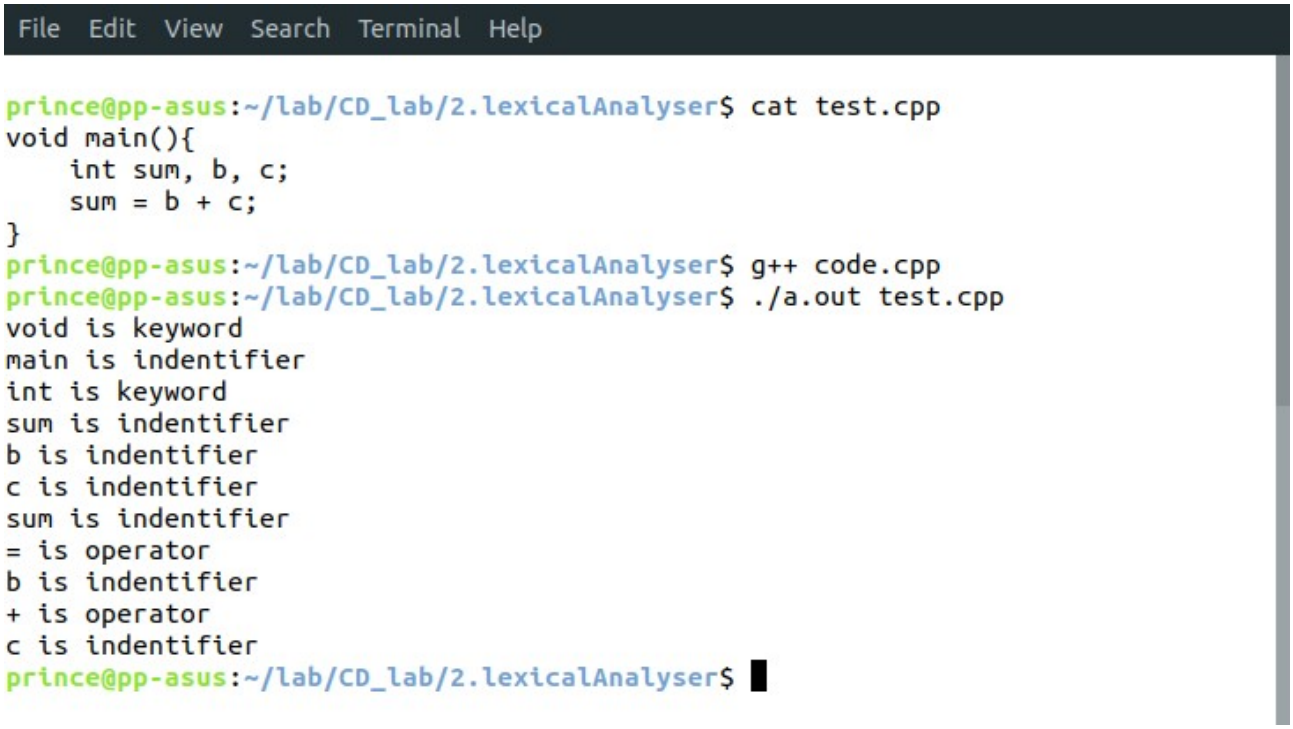
In programming language, keywords, constants, identifiers, strings, numbers, operators and punctuations symbols can be considered as tokens.

Code -

```
#include <bits/stdc++.h>

using namespace std;

const int keywordCount = 32;
char keywords[keywordCount][10] = {
   "auto", "break", "case", "char", "const", "continue",
   "default", "do", "double", "else", "enum", "extern",
   "float", "for", "goto", "if", "int", "long", "register",
   "return", "short", "signed", "sizeof", "static",
   "struct", "switch", "typedef", "union", "unsigned",
   "void", "volatile", "while"
};

const int operatorCount = 6;
char operators[operatorCount] = {
   '+', '-', '*', '/', '%', '='
};



int isKeyword(char token[]){
   for (int i = 0; i < keywordCount ; ++i){
      if (strcmp(keywords[i], token) == 0){
         return 1;
      }
   }
   return 0;
}

int main(int argc,char *argv[]){
   char *filename;
   char ch, token[15];
   int i, j = 0;

   if(argc>1){
```

```
        filename = argv[1];
    }else{
        cout<<"Please Enter name of the code file: ";
        cin>>filename;
    }

    ifstream fin(filename);
    if (!fin.is_open()){
        cout << "Error opening the file\n";
        exit(0);
    }

    while (!fin.eof()){

        ch = fin.get();
        for (i = 0; i < operatorCount; ++i){
            if (ch == operators[i])
                cout << ch << " is operator\n";
        }

        if (isalnum(ch)){
            token[j++] = ch;
        }else if ((ch == ' ' || ch == '\n') && (j != 0)){
            token[j] = '\0';
            j = 0;
            if (isKeyword(token) == 1){
                cout << token << " is keyword\n";
            }else{
                cout << token << " is indentifier\n";
            }
        }
    }

    fin.close();
    return 0;
}
```

Output –



Learnings – Through this program, we came to know about the phases of a compiler, specifically, lexical analysis. We learnt about the lexical analyser and the concept of tokens in a programming language. We made a program to differentiate between the different types of tokens.