Problem Statement: Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language.

AIM: Write a C/C++ program to implement the design of a Lexical analyzer to recognize the tokens defined by the given grammar.

Lexical analysis reads the characters in the source program and groups them into stream of tokens in which each token represents a logically cohesive sequence of characters such as an identifier, keyword, and punctuation character. The character sequence forming a token is called lexeme of the token.

ALGORITHM / PROCEDURE:

We make use of the following two functions in the process.
Look up() – it takes string as argument and checks its presence in the symbol table. If the string is found then returns the address else it returns NULL.
insert() – it takes string as its argument and the same is inserted into the symbol table and the corresponding address is returned.

1. Start
2. Declare an array of characters, an input file to store the input;
3. Read the character from the input file and put it into character type of variable, say ‘c’.
4. If ‘c’ is blank then do nothing.
5. If ‘c’ is new line character line=line+1.
6. If ‘c’ is digit, set token Val, the value assigned for a digit and return the ‘NUMBER’.
7. If ‘c’ is proper token then assign the token value.
8. Print the complete table with

Token entered by the user,
Associated token value.

9. Stop

[Viva Questions]

1. What is lexical analyzer?
2. Which compiler is used for lexical analysis?
3. What is the output of Lexical analyzer?
4. Which Finite state machines are used in lexical analyzer design?
5. What is the role of regular expressions, grammars in Lexical Analyzer?

Exercise: 1

1. Write a C program to recognize strings under 'a', 'a*b+', 'abb'.