Objectives:

1. To gather knowledge about Lexical Analysis
2. To get handoff practice of FLEX (Fast Lexical Analysis Generator)
3. To identify and categorize keywords, variables and other language component
4. To design our own language syntax **Introduction:**

Lexical analysis, also known as scanning or tokenization, is the first phase of compiling or interpreting a programming language. It involves breaking down the source code into smaller units called tokens.

1. Tokenization: Lexical analysis is like reading a sentence and splitting it into individual words. In programming, it reads the source code and breaks it into tokens, which are the smallest meaningful units like keywords, identifiers, operators, and literals.

2. Whitespace and Comments: Lexical analysis ignores whitespace (spaces, tabs, line breaks) and comments because they are not meaningful in most programming languages. It focuses on extracting meaningful code elements.3

3. Error Detection: Lexical analysis also identifies and reports errors, such as syntax errors, when it encounters invalid or unexpected tokens. These errors can help programmers find and fix issues in their code.

4. Symbol Table: Lexical analysis often maintains a symbol table or a dictionary that keeps track of identifiers (variable names, function names) encountered in the code. This table is used later in the compilation process.

5. 5. Output: The output of lexical analysis is typically a stream of tokens that serve as input for the next phases of the compiler or interpreter. These tokens represent the building blocks of the program and are used to create a structured representation for further processing.

In summary, lexical analysis is the process of reading source code, breaking it into tokens, ignoring whitespace and comments, detecting errors, and preparing a token stream for further processing in the compilation or interpretation of a programming language.

Basic Structure of my language:

Keywords:

SHURU → Used for start the program.

SHESH → Mark the program as ended.

NAO → Used for taking input from user.

DAO → Used for printing a value.

INC → Used for increment a decrement of a variable value. Can use any arithmetic operation.

<--Text--> → Used for multiline comment.

 # Text → Used for single line comment

JODI → used for conditional statement.

JODI	- Start of JODI
SHORTO a>50	- Condition to be checked
HOY	- If condition is true
Statement	- Do this
NAHOY	- When condition is false
Statement	- Do this
IDOJ	-End of Jodi

LOOP → used for repetitive work. (Nested loop also works)

PURNO a=0	- Initialization of loop variable
LOOP	- Loop start
SHORTO a>50	-Condition to be checked as true

INC a++ - After every loop operation loop variable decreased or increased

Statement - Condition true. Then do this

POOL - Loop sesh

Data Types:

Integer:

PURNO a, b, c = 30 → The type of integer data type in my program is PURNO. We can declare one variable, multiple Variable with or without assigning value.

Float:

VOGNO a, b = 30.3 → The type of integer data type in my program is PURNO. We can declare one variable, multiple Variable with or without assigning value.

Char:

OKKHOR x="a", y="0", z → The type of integer data type in my program is OKKHOR. We can declare one variable, multiple Variable with or without assigning value.

SYMBOL TABLE:

```
struct symbolTable {
        char type [10];
        char name [256];
};
```

Structure Data type is used for representing symbol table.
All the variable name and keywork name is stored in this symbol table for preventing error from

- Declaring same variable more than one time
- Using a variable without declaring
- Using a keyword name as variable

Operators:

Arithmetic Operator:

+ → used for addition

- → used for subtraction

* → used for Multiplication

/ → used for division

Relational Operator:

> →less than operator

< →greater than operator

>= →less than equal operator

<= →greater than operator

== →equal operator

!= → not equal operator

INPUT:

```
SHURU
PURNO a,b
VOGNO c=90,d=84.0
OKKHOR x="a",y="0",z
NAO a
DAO b
LOOP
SHORTO a<500
INC a += b
LOOP
SHORTO b<500
INC a += b
INC a += 9
   a=b
POOL
POOL
JODI
SHORTO a<56 HOY
   a=b
NAHOY
```

```
   a=b
IDOJ
<--
This is a multicomment jgjk
-->
#this is a single comment
SHESH
```

OUTPUT:

```
Total Variable = 7
a -> PURNO b
-> PURNO c -
> VOGNO d -
> VOGNO x -
> OKKHOR y
-> OKKHOR z
-> OKKHOR
Total Statement: 11
Total Loop: 2
Total IfElse: 1
Your source code is OK :)
```

Discussion:

In this assignment, I learned flex software. I did Lexical Analysis with this Software. I tried to create my custom syntax for my own language. I used regular expression to distinguish between keyword, variable etc. I learned the idea how a compiler checks the given input is syntactically matched with the language's syntax or not. It's been a fascinating journey into the world of language design and compiler construction.