LEX is a popular lexical analyzer generator used in compiler construction. It is a tool that helps in generating lexical analyzers, also known as scanners or tokenizers, for programming languages or other formal languages.

## Introduction

- Lexical analyzer generator scans the input stream and converts sequences of characters into tokens.
- The main job of lexical analyzer (scanner) is used to break up an input stream into more usable element(tokens)
- Lex is not a complete language, but rather a generator representing a new language feature which can be added to different programming languages, called host languages.

## Here’s an overview of LEX :

1. Definition: LEX is a program that generates lexical analyzers based on user-defined rules and patterns. It takes a set of regular expressions and corresponding actions as input and produces a scanner program in C or C++.
2. Lexical Analysis: Lexical analysis is the process of breaking down the input source code into tokens or lexemes, which are the smallest meaningful units of the language. Tokens can include identifiers, keywords, operators, literals, and more.
3. Regular Expressions: LEX uses regular expressions to define the patterns of tokens. Regular expressions are a concise way of describing complex patterns, allowing you to match specific sequences of characters.
4. Actions: Along with regular expressions, you provide corresponding actions in LEX. Actions are code snippets written in the target language (C or C++) and executed when a pattern is matched. These actions typically generate output or perform other

processing based on the recognized tokens.

5. Tokenization Process: LEX generates a scanner program that reads the input source code character by character. It matches the input against the specified regular expressions and executes the corresponding actions when a match is found. This process identifies and returns tokens to the parser or other components of the compiler.
6. Integration with Compiler: The generated scanner program from LEX is typically integrated into the larger compiler framework. It is used as the initial stage of the compilation process, where it performs the lexical analysis and tokenization of the input source code.
7. Efficiency and Performance: LEX-generated scanners are efficient because they use techniques like finite automata to match patterns. This allows for fast and optimized token recognition, making the overall compilation process more efficient.
8. Flex: Flex is an open-source alternative to LEX and provides similar functionality. It is compatible with LEX and offers additional features and improvements.

## What is Token in Lex ?

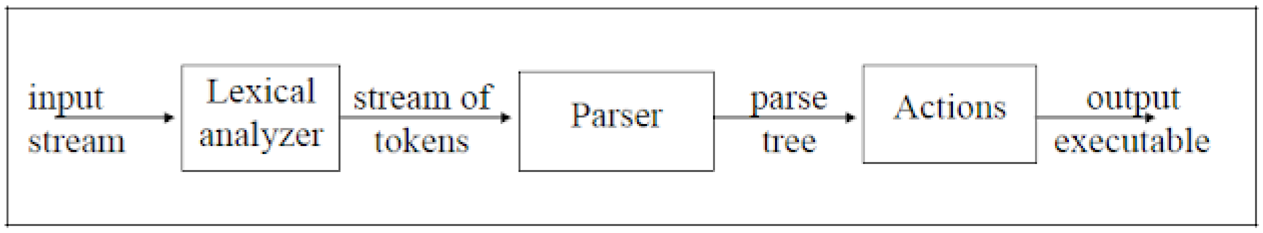
Token is a classification of groups of characters.

For example:

| Lexeme | Token |
|---|---|
| = | EQUAL_OP |
| * | MULT_OP |
| , | COMMA |

| | |
|---|---|
| ( | LEFT_PAREN |

- Lex is a tool for writing lexical analyzers.
- Lex reads a specification file containing regular expressions and generates a C routine that performs lexical analysis. Matches sequences that identify tokens.



## A Lex program consist of 3 sections:

- The first section of declaration includes declaration of variable, constants and regular definition.
- The second section is for translation rules which consists of regular expression and action with respect to it.
- The third section holds whatever program subroutine are needed by the action.