Government College of Engineering, Jalgaon

(An Autonomous Institute of Govt. of Maharashtra)

Name of Student:……
PRN:……. …
Subject teacher: Sharayu Bonde

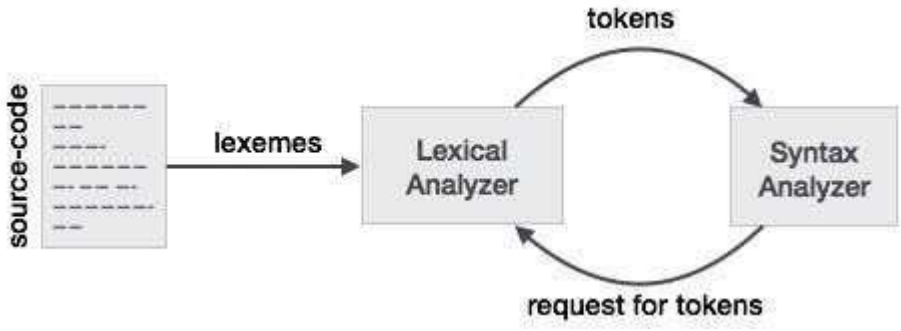
Experiment No.: Group A: 0

Title: Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines.

Theory:

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.

For example, in C language, the variable declaration line

Int val 100;

contains the tokens:

int (keyword), value (identifier), = (operator), 100 (constant) and ; (symbol) .

Representing language tokens using regular expressions

Decimal = $(sign)^+(digit)^+$

Identifier = (letter)(letter | digit)*

The only problem left with the lexical analyzer is how to verify the validity of a regular expression used in specifying the patterns of keywords of a language. A well-accepted solution is to use finite automata for verification.

Finite automata is a state machine that takes a string of symbols as input and changes its state accordingly. Finite automata is a recognizer for regular expressions. When a regular expression string is fed into finite automata, it changes its state for each literal. If the input string is successfully processed and the automata reaches its final state, it is accepted, i.e., the string just fed was said to be a valid token of the language in hand.

The mathematical model of finite automata consists of:

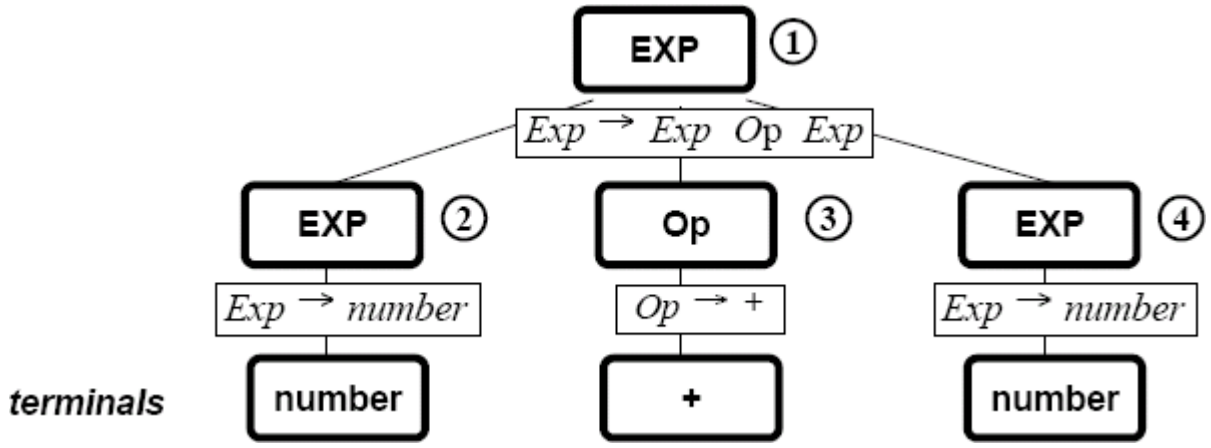
- Finite set of states (Q)
- Finite set of input symbols (Σ)
- One Start state (q0)
- Set of final states (qf)
- Transition function (δ)

The transition function (δ) maps the finite set of state (Q) to a finite set of input symbols (Σ), $Q \times \Sigma \rightarrow Q$

Importance of Lexical Analyser:

- Lexical analysis makes writing a parser much easier.
- Instead of having to build up names such as "net_worth_future" from their individual characters, the parser can start with tokens and concern itself only with syntactical matters.
- This leads to efficiency of programming, if not efficiency of execution.
- However, since the lexical analyzer is the subsystem that must examine every single character of the input, it can be a compute-intensive step whose performance is critical, such as when used in a compiler

Example:



Result:

Thus the program for designing of lexical analyserfor a given language is performed successfully.

Course Teacher

Sharayu N. Bonde