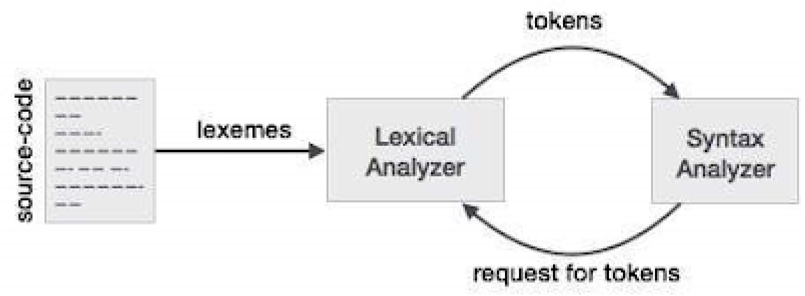
[redacted]

1. The lexical analyzer breaks syntaxes into a series of tokens, by removing any whitespace or comments in the source code.
2. If the lexical analyzer finds a token invalid, it generates an error. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.



[redacted]

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 lineint value = 100;contains the
tokens:int (keyword), value (identifier), = (operator), 100 (constant) and ; (symbol).

Lexeme	Token
=	EQUAL_OP
*	MULT_OP
,	COMMA
(	LEFT_PAREN

[redacted]

Let us understand how the language theory undertakes the following terms:

1. Alphabets
2. Strings
3. Special symbols
4. Language
5. Longest match rule
6. Operations
7. Notations
8. Representing valid tokens of a language in regular expression
9. Finite automata

1.[redacted] Any finite set of symbols

- {0,1} is a set of binary alphabets,
- {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} is a set of Hexadecimal alphabets,
- {a-z, A-Z} is a set of English language alphabets.

2.[redacted]Any finite sequence of alphabets is called a string.

3. [redacted] A typical high-level language contains the following symbols:

Arithmetic Symbols	Addition(+), Subtraction(-), Multiplication(*), Division(/)
Punctuation	Comma(,), Semicolon(;), Dot(.)
Assignment	=

Special assignment	+=, -=, *=, /=
Comparison	==, !=. <. <=. >, >=
Preprocessor	#

4. [redacted] A language is considered as a finite set of strings over some finite set of alphabets.

5. [redacted] When the lexical analyzer read the source-code, it scans the code letter by letter and when it encounters a whitespace, operator symbol, or special symbols it decides that a word is completed.

6. [redacted]The various operations on languages are:

1. Union of two languages L and M is written as, L U M = {s | s is in L or s is in M}
2. Concatenation of two languages L and M is written as, LM = {st | s is in L and t is in M}
3. The Kleene Closure of a language L is written as, L* = Zero or more occurrence of language L.

7. [redacted] If r and s are regular expressions denoting the languages L(r) and L(s), then

1. Union : L(r)UL(s)
2. Concatenation : L(r)L(s)
3. Kleene closure : (L(r))*

8. [redacted]If x is a regular expression, then:

- x* means zero or more occurrence of x.
- x+ means one or more occurrence of x.

9. [redacted] Finite automata is a state machine that takes a string of symbols as input and changes its state accordingly.If the input string is successfully processed and the automata reaches its final state, it is accepted.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), $Q \times \Sigma \rightarrow Q$

Related Posts:

1. Introduction to Compiler
2. Analysis and synthesis model of compilation
3. Bootstrapping and Porting
4. Lexical Analyzer: Input Buffering
5. Storage Allocation Strategies
6. Type Checking
7. Front end and back end of the compiler
8. LEX
9. Analysis synthesis model of compilation

10. Data structure in CD
11. Register allocation and assignment
12. Loops in flow graphs
13. Dead code elimination
14. Syntax analysis CFGs
15. L-attribute definition
16. Operator precedence parsing
17. Analysis of syntax directed definition
18. Recursive descent parser
19. Function and operator overloading
20. Storage allocation strategies
21. Equivalence of expression in type checking
22. Storage organization
23. Parameter passing
24. Run time environment
25. Type checking
26. Code generation issue in design of code generator
27. Boolean expression
28. Declaration and assignment in intermediate code generation
29. Code optimization
30. Sources of optimization of basic blocks
31. Loop optimization
32. Global data flow analysis
33. Data flow analysis of structure flow graph (SFG)