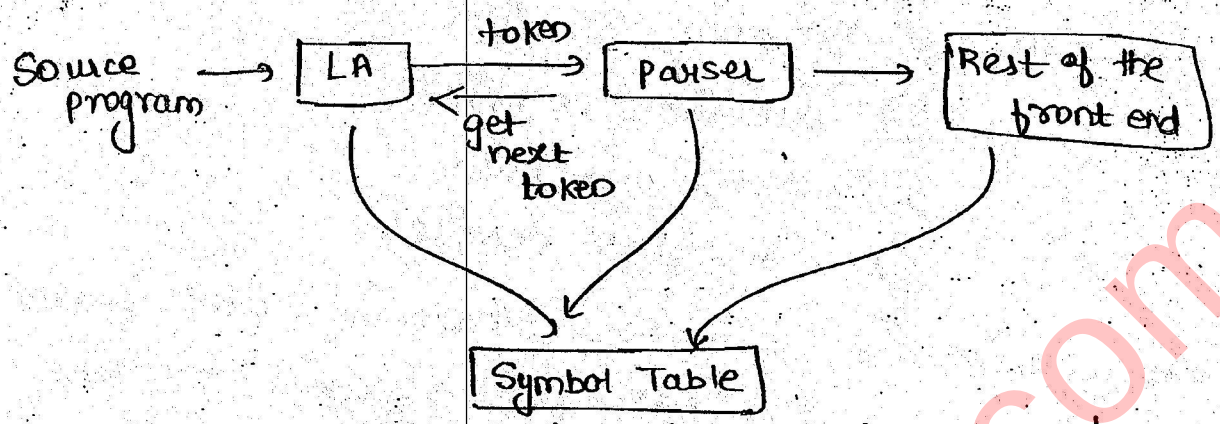
Lexical Analysis



The lexical analyser scans the source statement character by character, then breaks the statements into stream of tokens. Therefor, Lexical analyser is called as token recognizer.

Other functions -:

1. maintains separate track of line no.
2. skips out white spaces.
3. skips out the comment lines.
4. To implement a lexical analyser module we use two Concepts -: (a) finite automata
   (b) Regular expressions.

→ i.e The process of lexical analyser generator starts with token encryption

Each token represented in finite automata from that finite automata lexical analyser module will be return.

Regular expressions are used to describe tokens of the programming language whereas FA is used to recognize these tokens, i.e. - - - -

* Finite automata will be regarded as token recognizer or token scanner.
* Regular expression as token generator or token descriptor.

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com

token (description) → RE → NFA-ε → NFA → DFA → mini DFA → code

eg. RE for Integer :-

eg. 123

one or more digit

Let d → digit.

So, expression = d^+

$\Rightarrow dd^*$

where $d = 0|1|2|3 \cdots |9$

eg. for float :-

12.12

$R.E = d^+ . d^+$

$= d d^* . d d^*$

Q. Give a Regular expression for language identifier which defined as letter followed by any no. of letters followed by any no. of digits.

Let l denotes the letters, d denotes the digit

$\Rightarrow ll^* d^*$ where $l = a|b|c \cdots |z$

$d = 0|1|2|3 \cdots |9$.

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com

Q. Give the C language identifier RE.
let l denotes letters, d denotes digit,
u denotes underscore.

~~$(l+u)(l+u)^*d^*(l+u)^*$~~

$(l+u)(l+u+d)^*$ ← if after digit, letter or _ is permitted.
↓ But this will give ∞. which is not possible in C, max limit is 31.

$\boxed{(l+u)(l+u+d+\varepsilon)^{30}}$ ← correct

Q. If a in programming language P, an identifier defined as letter followed by any no. of letters and digits combinations.

⇒ $l(l+d)^*$.

Q. In some programming language, identifier defined over the alphabet digit, letters, Sp Sp (Special symbol)
beginning with letter ending with digit.
Let l denote the letter, d denote the digit and S denote the special symbol.

⇒ $l(l+d+S)^*d$.

RE → C – Keyword
C → int | float | char | main | . . . . | extern

RE for operators ?.
Relational = < | <= | == | != | > | >=.
for special symbol -:
SP : ; | · | # | → | ? | (|) . . . .

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com

Tokens Types :-

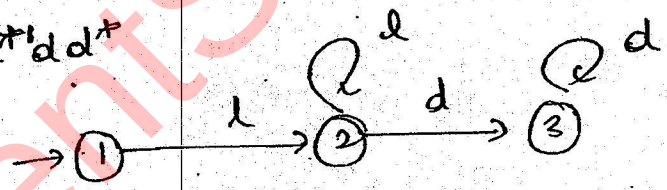
(a) Identifier
(b) operator
(c) Constant
(d) Keywords
(e) Special Symbol / punctuation symbol.

Q. In some programming language an identifier define as all names formed using the alphabet letters, digits, beginning with digits ending with 2 letters followed by dot of length atmost 32 characters.
Let l denotes letters, d denote digits, S denot dot; P denotes special Symbol.

⇒ $\boxed{d(d+l+\varepsilon)^{28}\, l\, l\, S}$

Let RE = ll^*dd^*

FA =



State 1 :-
```
c = getchar()
if (c == l) goto State (2)
else
    fail()
```

State 2 :-
```
c = getchar()
if (c == l) go to state (2)
else if (c == d) go to state (3)
else
    return();
```

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com

But this is very complex process, instead of this nowadays Lex, Flex is used. These are open source softwares.

Q. consider the following FORTRAN statement identifying the no. of tokens in the statement.

DO 10 I = 6.25

- DO → Keyword
- 10 → label
- I → identifier
- = → operator
- 6.25 → constant

So total 5.

Q. consider the following C statement.

```
if (x <= y) printf(y = z)
z = z+1;
```

Total = 18.

```
printf("%d", &x);
```

Total 8

Q. find the no. of tokens in the following statement :-

```
printf(" Total = %d", sum);
```

Total = 7

Download all Notes and papers from StudentSuvidha.com

Q. Which one of the following is not a token of C program.?

(a). 1.02e + 2
(b). MAX
(c). 123.33
(d). None of these.

Q. Which one of the following is not a valid token in C?

(a). ab123
(b). 123
(c). #include → as it is a group of token. not a single token.
(d). 12.2

CONTEXT FREE GRAMMARS

Context free grammars are useful to define syntaxes of high level languages.
Hence, all high level language are called CFL's.
eg. C, C++, Java.

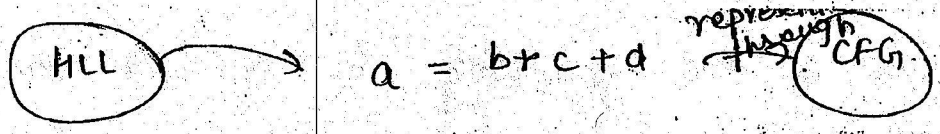
A context free grammar is a 4 tuple system (V, T, P, S) and every production is of the form $A \rightarrow \alpha$

where, $A \in V$
$\alpha \in (V \cup T)^*$.

i.e, left side symbol must be non-terminal, right side one can be combination of variables and terminals.

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com



eg. Integer : 123

<Integer> → <Integer><digit> | <digit>

<digit> → 0|1|2|3 ... |9.

↳ Here assume that start symbol is <integer>.

eg. floating : 123.123.

<Floatno> → <Int><dot><Int>

<Int> → <Int><digit> | <digit>

<digit> → 0|1|2|3 ... |9.

<dot> → .

eg. a*b+c-(d)

⇒ <Exp> → <Exp> * <Exp> | <Exp> + <Exp> | (E) | <Exp> - (Exp) | a | b | c | d.

eg. & in TOC, we write as :-

E → E+T | T

T → T*F | F

F → (E) | a

↳ Here Exp → Exp + Term | Term

Term → Term * Factor | Factor

Factor → (EXP) | a.

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com

Let $w = E * E : a * b + c$

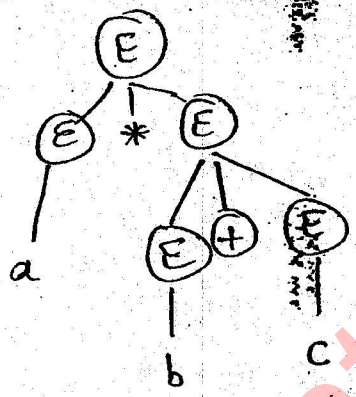
LMD

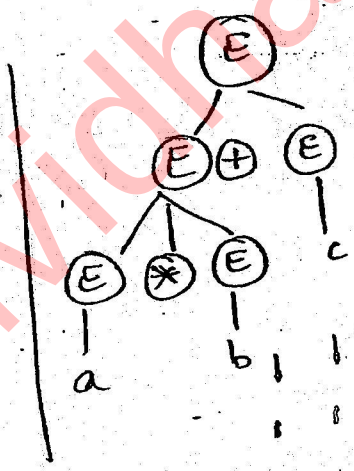
$E \to E * E$
$\to a * E$
$\to a * E + E$
$\to a * b + E$
$\to a * b + c$

RMD.

$E \to E * E$
$E \to E * E + E$
$E \to E + E + c$
$E \to E * b + c$
$E \to a * b + c$

Parse tree -





Since 2 parse tree possible, so ambiguous

StudentSuvidha.com

Download all Notes and papers from StudentSuvidha.com