Lexical Analysis

Tuesday, March 29, 2022 9:28 PM

Lexical Analysis (Scanner)

It scans the stream of characters making up the source program is read from left to right and grouped into tokens, that are sequence of characters having collective meaning. Tokens are of type Keywords, Identifiers, operators, constants and special characters.

Ex:- Total = sub1 + sub2 * 2;

Identifier: Total, sub1, sub2
Operator: =, +, *
Constant: 2
Special character: ;

Symbol table

Total	float	garbage --->
sub1	float	0.0 --->
sub2	float	garbage --->

Basic functions of lexical Analysis phase

1. Remove the white spaces (tab, newline) and comments.
2. Identify the tokens.
3. Creating storage for identifiers in the symbol table.

Q:- Find the number of tokens in the following program code?

①
```
main()
{
// First program ignored
printf("Hello Students");
}
```
Tokens = 10

②
```
while(i>0)
{
printf("%d", i);
i++;
}
```
Tokens = 18

Gate-2000
The number of tokens in the following 'C' statement.

```
printf("i=%d, &i=%u", i, &i);
```

a) 3 b) 26 c) 10 d) 21.

Q:- which of the following can be recognized as token without depending upon the next input character.

a) int b) total c) ++ d) all of a,b,c

Char integer;

Types of Lexical errors :-

i) Unterminated Comments

Absent */

UI - I

1) Unterminated Comments

/* ---------- */
begining — end (Absent)

2) Nested Comment

/* .... /* ---------- */ ..... */
begining ---------- end — end

3. Length of the longest identifier:- (<=31 characters)

int interest_and_compound_interest_on_principle_amount;

4. Invalid identifier:-

int 1stmontSalary;

5. Illegal Symbol :- int Salary = ₹50000;

6. Invalid Constant:- int Salary = 50,000;

Error Recovery in Lexical Analysis

1. Delete : an extra character Ex:- int Salary = ₹50000; (remove ₹)

2. Insert : missing character. Ex:- /* ---------- */ (insert */)

3. Transpose :- interchanging of two adjacent characters. Ex:- int 1st Salary;

4. Replace : replacing a character with another one. Ex:- int 1stSalary; (F: 1 → replace)

"LEX" Tool (FLEX) (JFLEX)

→ Lex is a tool to specify lexical analyzer.

→ Lex tool is called Lex Compiler.

→ Lex program

It has 3 parts 1. declaration
2. Translation rules
3. Auxillary procedures

Lex.l

```
ID  { letter(letter/digit)* }
%%
{ID}  { yylval = install_id(); return (ID) }
%%
install_id()
    { //to store identifiers into the symbol table.
    }
```

Lex.l → LEX Compiler → lex.yy.c

Tabular representation of Transition diagram constructed for the regular expression and Standard procedures that use the table to recognize the tokens.

lex.yy.c → C Compiler → a.out — lexical Analyzer.

Source progr → a.out → Tokens.

Ex: ID: l(l/d)*

Transition diagram

	l	d
I	F	∅
F	F	F

Transition table

Transition diagram

Transition table