Lexical Analysis

Monday, February 8, 2021 8:25 AM

Lexical analysis:
Input: source program
Process: read input and group into meaningful words (lexemes)
Output: tokens

Lexical analyzer interacts with the symbol table.

Symbol table:

Name	Type	Value
A	Int	10

Functions of the lexical analyzer(depending on the language):
- Removes whitespace (blank, newline, tab)
- Tracks line number
- Macro expansion

Token
`<token_name, attribute>`

Token name: a symbol/label that we assign to a particular lexical unit
Attribute: optional, pointer to symbol variable

Classes:
- Keyword
- Identifier
- Operator
- Numerical constants
- String constants

Pattern: rule that describes what is the form of a valid lexeme for which a token can be generated

Keyword	'i' 'f'
identifier	Letter followed by other letters

Lexeme: sequence of characters in the source program that obeys a pattern for which a token can be generated

Input buffering:

- In C, space marks the end of a lexeme. (space-sensitive)
- Fortran is not space-sensitive

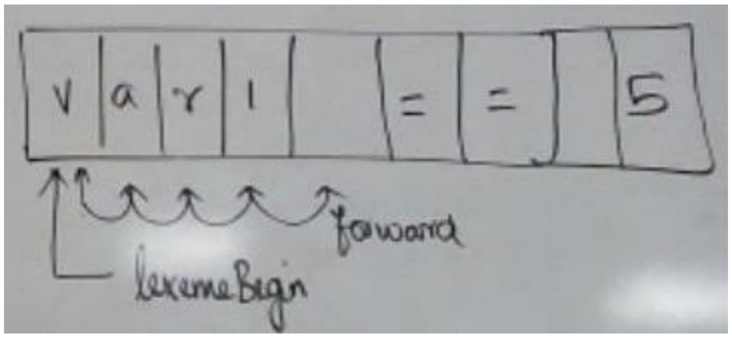
Lookahead process: after seeing the space after a lexeme, the lexical analyzer can confirm that it has found a lexeme. (in space-sensitive languages)

Use 2 buffers:

Buf1	Buf2
n=size of disk block 1 read 4096 bytes Source pgm is of 150 bytes	

When analyzer encounters a eof, stop reading file.
Eof to denote end of buffer also (helps in the lookahead process)

Two pointers:
lexemeBegin: placed at the beginning of the lexemes
Forward: scans and moves ahead



- Forward pointer is set to the rightmost end of the lexeme (backtrack to '1')
- Copy the valid lexeme to the attribute field of the token
- `<id, var1>`
- Place forward at the start of the next lexeme '='

Sentinels: mark beginning and ending of a character sequence

Lexical errors:
- Panic mode recovery: keep deleting characters from where you can identify a lexeme
- *Dkfjalkd***if**
- Delete/insert missing characters/replace/transpose characters

Use regular expressions as a tool to define patterns
- ∑ {0,1} (alphabet)
- S = 00, |S| = 2

- Concatenation

- Exponentiation
- Union
- Closure (kleene, positive)