Lexical Analysis

Linear analysis: lexical analysis, scanning

e.g., position:= initial+rate*60

1.Identifier position

2.Assignment symbol “: =“

3.Identifier initial

4.“+” sign

5.Identifier rate

6.“*” sign

7.number 60

→Removal of White Space and Comments

Blanks, tabs, newlines

→Constants

Adding production to the grammar for expressions

Creating a token num for constants

31 + 28 + 59

<num, 31> <+, > < num, 28> <+, > < num, 59>

→Recognizing Identifiers and Keywords

Keywords are reserved

```
Begin                /* keyword */
count = count + increment;     /* id = id + id */
end
```

Interface to the Lexical Analyzer

→A lexical analyzer reads characters, group into lexemes , and passes the tokens formed by the lexemes, together with their attribute values to the later stages of the compiler.

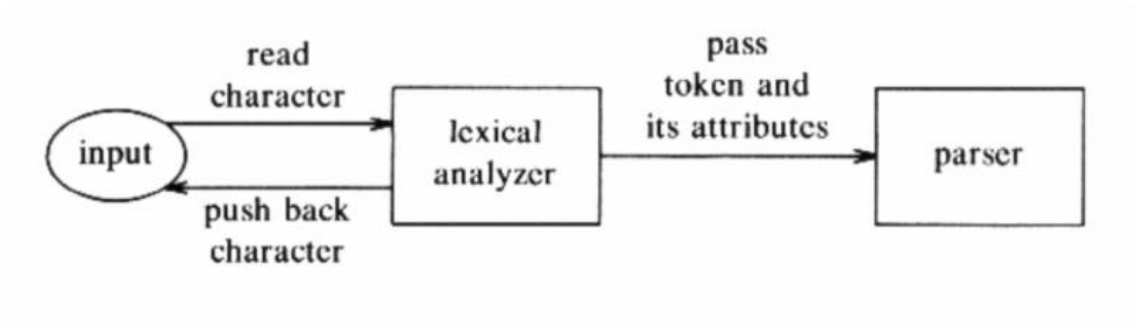


Fig. 2.25. Inserting a lexical analyzer between the input and the parser.

→In some situations, the lexical analyzer has to read some characters ahead before it can decide on the token to be returned to the parser.

Decide ‘>’ or ‘>=‘

Push back if need

Using an input buffer and a pointer keeping track the next character.

→The lexical analyzer produces a token and the parser consumes the token.

→Usually, the parser call the lexical analyzer to return tokens on demand.

A Lexical Analyzer

A lexical analyzer allows white space and numbers to appear within expressions.

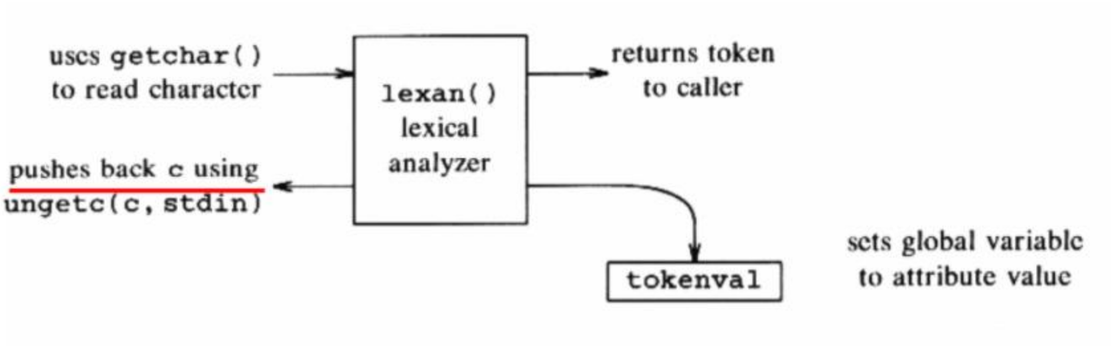

uses getchar() to read character
pushes back c using ungetc(c, stdin)
lexan() lexical analyzer
returns token to caller
tokenval
sets global variable to attribute value

Fig. 2.26. Implementing the interactions in Fig. 2.25.

→If a data structure does not be allowed to be returned, then tokens and their attributed have to be passed separately.

→Usually, lexan returns an integer encoding of a token

→Use integer '256' to encode num

→tokenval: token attribute value

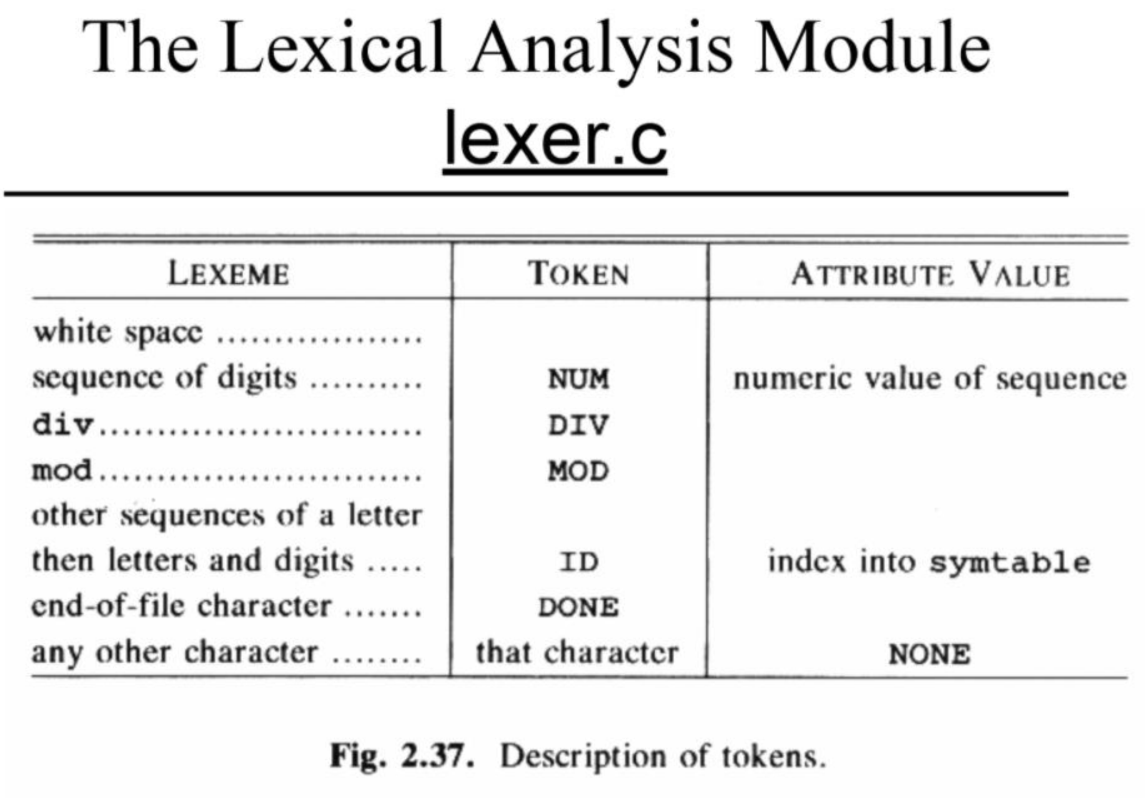
When scans an integer 13, token num (256) and tokenval (13) are returned to parser

When scans an identifier initial, token id (259) and tokenval (symbol table index

p) are returned to parser.

→Allowing numbers within expressions requires a change in grammar expr → factor

factor → (expr) | num {print(num.value)}



The Lexical Analysis Module
lexer.c

LEXEME	TOKEN	ATTRIBUTE VALUE
white space		
sequence of digits	NUM	numeric value of sequence
div............................	DIV	
mod...........................	MOD	
other sequences of a letter then letters and digits	ID	index into symtable
end-of-file character	DONE	
any other character	that character	NONE

Fig. 2.37. Description of tokens.

The Role of The Lexical Analyzer

→Its main task is to read the input characters and produce as output a sequence of tokens that the parser uses for syntax analysis.

→It also performs certain secondary tasks such as stripping out comments and white space and correlating error messages with the source program

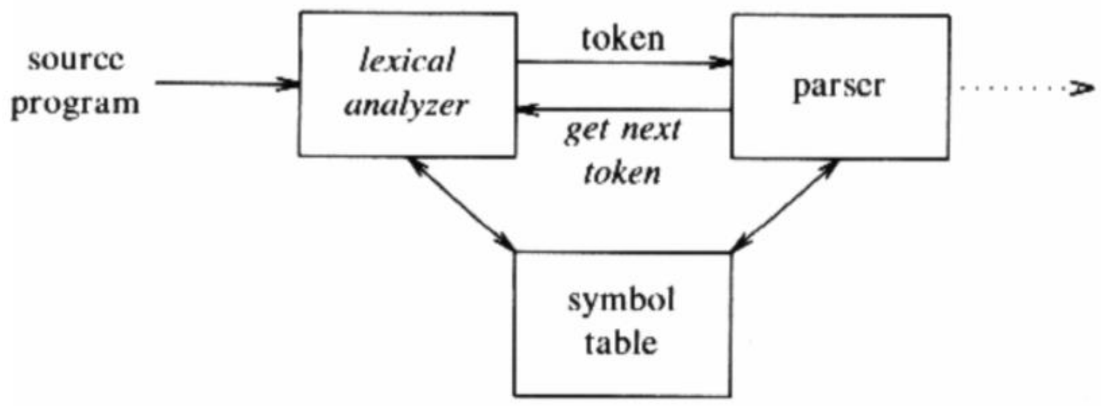


Fig. 3.1. Interaction of lexical analyzer with parser.