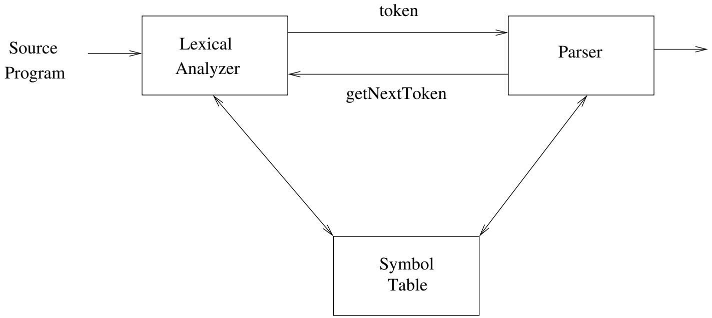
Lexical Analysis

Sudakshina Dutta

IIT Goa

28th January, 2022

- Lexical analyzer reads character from input and groups them into lexemes, produce as a output a sequence of tokens for each lexemes of the source program
- The stream of token is sent to parser for syntax analysis
- Parser invokes the lexical analyzer by *getNextToken* command
- Lexical analyzer reads the characters from input until it finds the next lexeme and produce token



Tokens, Patterns and Lexemes

- **Lexeme** : It is a sequence of characters that matches the pattern for a token. It is identified by the lexical analyzer as an instance of that token
- When the lexical analyzer discovers a lexeme constituting an identifier, it enters that to the symbol table
- **Pattern** : A description that the lexeme may take. It is matched by many strings
  - A variable can start with a letter and can have letters, digits and it is expressed as $letter[letter|digit]^*$
  - Patterns can be represented using regular expressions

Tokens, Patterns and Lexemes

- **Token** : It is a pair consisting of a token name and an optional attribute value. The token name represents the kind of lexical unit (keyword/identifier)
  - Token is of the form $\langle token - name, attribute - values \rangle$
  - The tokens are terminal symbols of the grammar

Example of tokens

TOKEN	INFORMAL DESCRIPTION	SAMPLE LEXEMES
keywords	characters i, f	if
keywords	characters e, l, s, e	else
comparison	$<$ or $>$ or $\leq$ or $\geq$ or $==$ or $\neq$	$\leq$, $\neq$
id	letter followed by letters and digits	pi, score, D2
number	any numeric constant	3.14159, 0, 6.02e23
literal	anything but ‘‘, surrounded by "’s	‘‘core dumped"

If the input statement is
if (a $>$ 5)
    b $=$ 7;
then the tokens are $\langle keywords, if\rangle$, $\langle (\rangle$, $\langle id, a\rangle$, $\langle comparison, >\rangle$, $\langle number, 5\rangle$, $\langle id, b\rangle$, $\langle = \rangle$, $\langle number, 7\rangle$ and $\langle ; \rangle$

Example..continued

Syntax analysis/Parser phase encounters a statement position = initial + rate * 60 and calls LA phase

- The lexemes are position, =, initial, +, rate, *, 60
- Patterns
  - position, initial, rate match with the pattern for identifier
  - 60 matches the patterns for number
  - =, +, * are the tokens itself

Attribute for tokens

- Often lexical analyzer returns to the parser some attribute values
- The attributes help in translation phase after parsing
- Example of such attributes — lexeme, type, location where it is found first, error messages about the identifier, location in the symbol table, etc.
- Attribute values are kept in the symbol table

Example..continued

The lexical analysis phase on input position $=$ initial $+$ rate * 60 returns the following to the parser

- $\langle id, 1 \rangle$, $\langle = \rangle$, $\langle id, 2 \rangle$, $\langle + \rangle$, $\langle id, 3 \rangle$, $\langle * \rangle$, $\langle num, 60 \rangle$

Context-free Grammar

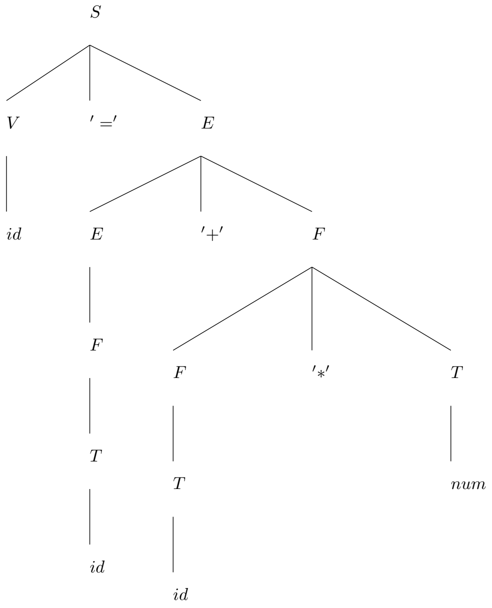
- Automata is a model of computation
- Context-free grammars (CFG) are required for specifying programming languages
- Generated language is called Context-free language (CFL)

An example

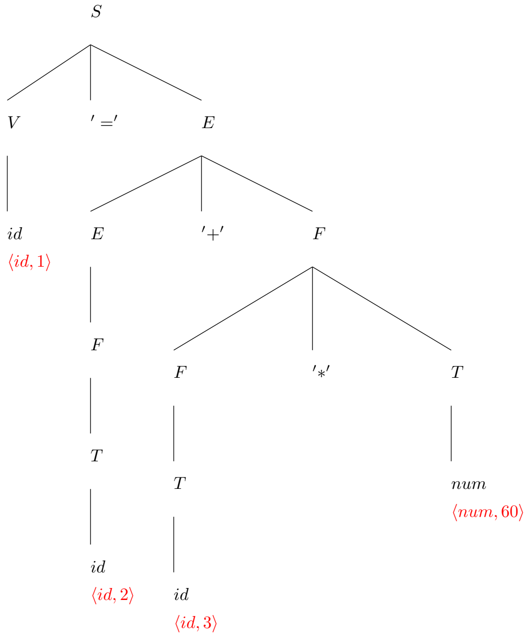
Suppose the input statement is position = initial + rate * 60 and the grammar is

- $S \rightarrow V$ '=' E
- $E \rightarrow E$ '+' F
- $E \rightarrow F$
- $F \rightarrow F$ '*' T
- $F \rightarrow T$
- $T \rightarrow id|num$
- $V \rightarrow id$

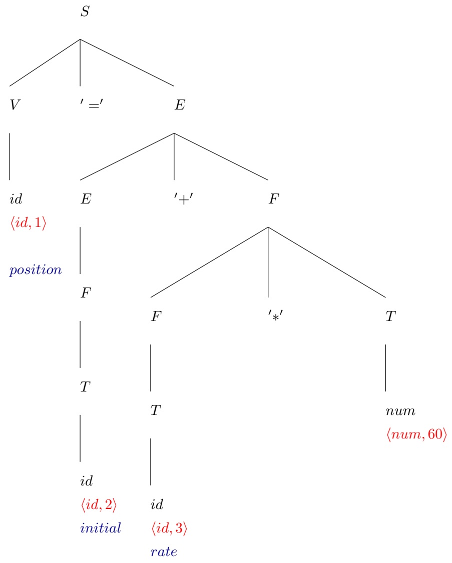
Interaction between parser and lexer



Interaction between parser and lexer



Interaction between parser and lexer



Tasks of Lexical Analyzer

- Removal of white spaces and comments
- Identifying constants
- Recognizing keywords
- Recognizing identifiers

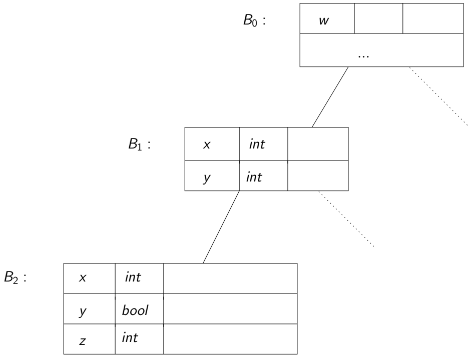
Symbol Table

- A data structure compiler uses to store information about source program constructs
- Attribute for an identifier is the pointer to the symbol-table entry
- Supports multiple declarations of the same identifier within a program
- Scope is implemented by setting up separate symbol table for each scope.

Scope

- Scope of a declaration is the portion of the program to which the declaration applies
- Scope is implemented by setting up separate symbol table for each scope e.g., each class has its own symbol table

```
{int $x_1$; char $y_1$;
    {int $w_2$; bool $y_2$; int $z_2$;
        $\cdots w_2 \cdots$ ; $\cdots x_1 \cdots$ ; $\cdots y_2 \cdots$ ; $\cdots z_2 \cdots$ ;
    }
    $\cdots w_0 \cdots$ ; $\cdots x_1 \cdots$ ; $\cdots y_1 \cdots$ ;
}
```



B_0 :
w
...
B_1 :
x
int
y
int
B_2 :
x
int
y
bool
z
int