

### Parameter Passing Mechanisms

#### Call-by-value:

Actual parameter is evaluated. Its value is placed in a the locating of the corresponding former parameter of the called procedure.

#### Call-by-reference:

Address of the actual parameter is passed to the value of the corresponding formal parameters. The expression is evaluated before the call, and its value is stored in a location of its own.

### Parsing

A *string of terminals* -> Figure out how to derive it from the start symbol of the grammar (reports errors) (most fundamental problem of compilers).

**Parse Tree:** shows how the start symbol of a grammar derives a string in the language.

**Ambiguous grammar:** a grammar is said to be ambiguous when there are more than one parse trees for generating a given string of terminals.

### Predictive Parsing

Top-down method for syntax analysis. Set of recursive procedures is used to process the input. Predictive parsing relies on the information about the first symbols that can be generated by a production body.

### Tokens, Patterns and Lexemes

#### Token

token name and an optional attribute value.

#### Pattern

a description of the form that lexemes of a token may take.

#### Lexeme

sequence of characters in the source program that matches a pattern for the token and is identified by the lexical analyzer as an instance of that token.

### Syntax Directed Translator (SYNTAX)

#### Syntax:

of a programming language describes the proper form of its programs.

#### Semantics:

defines what its programs mean.

#### Grammar:

naturally describes the hierarchical structure of most programming languages.

### Associativity of Operators

#### Left-associative

Addition, subtraction, multiplication and division.

#### Right-associative

Exponentiation, "C" =

### Syntax-Directed Translation

Done by attaching rules or program fragments to productions in grammars.

expr -> expr1 + term

Sum of two subexpressions

#### In pseudo-code:

Translate expr1;

Translate term;

Handle +;

### Lexical Analyzer

**1. Scanning** does not require tokenization.

**2. Lexical Analysis** produces tokens from the output of the scanner.

### Abstract Syntax Tree (AST)

- condensed form of parse trees
- represent the syntax of a program.
- collapse chains of productions into single steps
- separate parsing from semantic checking

### Abstract Syntax Tree (AST) (cont)

- Can manipulate abstract syntax after concrete syntax has been checked

- Can use syntax tree as intermediate representation

#### STRUCTURE:

- a node represents program construct e.g. node for an operator
- children represent components of the construct e.g. nodes for operands

### Context-Free Grammar

1. A set of *terminal* symbols (tokens).

Elementary symbols of the language defined in its own grammar.

2. A set of *nonterminals* (syntactic values).

3. A set of *productions*. Each production consists of: (a) a *nonterminal* which is the head of left side of the production, (b) an arrow, and (c) a sequence of terminals or non-terminals.

4. A designation of one of the nonterminals as the start symbol.

### Top-Down Parsing

The top-down construction of a parse tree is done by starting at the door, labelled with the starting nonterminal statement, and repeatedly parsing.

1. At node *N*, labelled with a nonterminal *A*, select one of the productions for *A* and construct a children at *N* for the symbols in the production body.
2. Find the next node at which a subtree is to be constructed, typically the leftmost unexpanded nonterminal of the tree.



By nsuarezcanton

Published 4th May, 2016.

Last updated 4th May, 2016.

Page 1 of 2.

Sponsored by **Readability-Score.com**

Measure your website readability!

<https://readability-score.com>

### Lexical Analysis

Reads characters from input and groups them into "token objects."  
Along with a terminal symbol that is used for parsing decisions.

Token -> Terminal + More Info.

*General approach to reading ahead on the input.* Maintain an input buffer from which the lexical analyzer can read and push back characters.

RegEx -> NFA ->DFA

### Symbol Tables

Map from identifiers to meanings. Keep track of

- *binding*: associating a name with a location
- *scope*: where in the program a name has meaning

#### USAGE:

1. Lexical Analyzer: add entries to ST
2. Parser: add type info, discover scope
3. Semantic Analyzer: use type info to find semantic errors
4. Code generator: determine where data are located, generate code to access locations



By nsuarezcanton

Published 4th May, 2016.

Last updated 4th May, 2016.

Page 2 of 2.

Sponsored by **Readability-Score.com**

Measure your website readability!

<https://readability-score.com>