What is parser ? Write the role of parser. What are the most popular parsing techniques ? OR Explain about basic parsing techniques. What is top-down parsing ? Explain in detail.

What is Parsing?

Parsing is like analyzing the structure of a sentence in a language. When you read a sentence, you understand it by breaking it down into subject, verb, object, etc. Similarly, in computing, parsing is understanding the structure of code or text according to some rules or grammar.

Role of Parser:

1. Check Syntax: It checks if the code follows the rules of the language’s grammar. Just like in language, a sentence must follow grammar rules to be meaningful, in programming, code must follow syntax rules.
2. Error Reporting and Recovery: If there’s a mistake in the code, the parser identifies it and tries to suggest where the mistake might be. It can help programmers understand and fix errors.
3. Construct Parse Tree: After checking the syntax, the parser builds a tree-like structure called a parse tree. This tree represents the syntactic structure of the code, which is used in later stages of compilation or interpretation.

Parsing Techniques:

1. Top-down Parsing:

What is it?

- Imagine you’re trying to understand a sentence by breaking it down starting from the beginning.

How does it work?

- It starts from the highest level of the grammar (like the start symbol) and tries to match the input by expanding the rules of the grammar.
- As it reads the input from left to right, it builds the parse tree top to bottom, hence the name.

In simple terms:

- It's like understanding a complex sentence by breaking it down from the main idea into smaller parts.

2. Bottom-up Parsing:

What is it?

- Here, you start from the smallest parts and combine them to form larger structures.

How does it work?

- It tries to find the rightmost derivation of the input string by reversing the process.
- It matches small pieces of the input to the right side of grammar rules and combines them until it reaches the start symbol.

In simple terms:

- It's like solving a puzzle by starting with the pieces and gradually assembling them into the whole picture.

Both techniques have their advantages and are used in different scenarios depending on the complexity of the language and the efficiency needed.

Related Posts:

1. What are the types of passes in compiler ?
2. Discuss the role of compiler writing tools. Describe various compiler writing tools.
3. What do you mean by regular expression ? Write the formal recursive definition of a regular expression.
4. How does finite automata useful for lexical analysis ?
5. Explain the implementation of lexical analyzer.
6. Write short notes on lexical analyzer generator.
7. Explain the automatic generation of lexical analyzer.
8. Explain the term token, lexeme and pattern.
9. What are the various LEX actions that are used in LEX programming ?
10. Describe grammar.
11. Explain formal grammar and its application to syntax analyzer.
12. Define parse tree. What are the conditions for constructing a parse tree from a CFG ?
13. Describe the capabilities of CFG.
14. What are the common conflicts that can be encountered in shift-reduce parser ?
15. Differentiate between top-down and bottom-up parser.Under which conditions predictive parsing can be constructed for a grammar ?
16. Differentiate between recursive descent parsing and predictive parsing.
17. What is the difference between S-attributed and L-attributed definitions ?
18. What is intermediate code generation and discuss benefits of intermediate code ?
19. Define parse tree. Why parse tree construction is only possible for CFG ?
20. Discuss symbol table with its capabilities ?
21. What are the symbol table requirements ? What are the demerits in the uniform structure of symbol table ?