1. Describing Programming Languages: CFG is like a set of rules that help us describe how sentences (or in this case, code) are structured in programming languages. It's a tool that programmers and language designers use to define the syntax of programming languages.
2. Automatic Parser Construction: With a well-designed CFG, we can create parsers – which are programs that analyze code and make sense of its structure – automatically. This means if we have a CFG for a programming language, we can build tools to check if code follows the rules of that language without needing to do it manually.
3. Handling Expressions: CFGs can also handle expressions in programming languages. This means they can help us understand things like arithmetic operations or logical comparisons. CFGs can even consider things like the order of operations (like multiplication before addition) and parentheses in expressions.
4. Dealing with Nested Structures: CFGs are handy for describing nested structures in code. For example, they can help us ensure that for every open parenthesis, there's a corresponding closing one, or that each 'if' statement has a corresponding 'else'. Essentially, CFGs help us manage the complexity of programming languages by describing how different parts of the code fit together.

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. 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.
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 ?