Types of passes :

1. Single-pass compiler:
    - Imagine you're reading a book from start to finish in one go without going back to previous pages.
    - In a single-pass compiler, it reads through your source code once, line by line.
    - As it reads, it breaks down each line into smaller units called tokens (like breaking down a sentence into words).
    - Then, it checks the structure of each line to ensure it follows the rules of the programming language.
    - Finally, it builds a tree-like structure representing the code's logic and creates tables containing information about each token.
2. Multi-pass compiler:
    - Picture going through multiple drafts of an essay, each time refining and improving it.
    - In a multi-pass compiler, it goes through your source code multiple times.
    - During each pass, it makes modifications to the code to improve it or prepare it for the next stage.
    - It might perform tasks like optimization, where it makes the code more efficient, or translation, where it converts code into a lower-level language.
    - Each pass might focus on different aspects of the code until it's ready to generate the final object code.

Related Posts:

1. Discuss the role of compiler writing tools. Describe various compiler writing tools.
2. What do you mean by regular expression ? Write the formal recursive definition of a regular expression.

3. How does finite automata useful for lexical analysis ?
4. Explain the implementation of lexical analyzer.
5. Write short notes on lexical analyzer generator.
6. Explain the automatic generation of lexical analyzer.
7. Explain the term token, lexeme and pattern.
8. What are the various LEX actions that are used in LEX programming ?
9. Describe grammar.
10. Explain formal grammar and its application to syntax analyzer.
11. Define parse tree. What are the conditions for constructing a parse tree from a CFG ?
12. Describe the capabilities of 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 ?