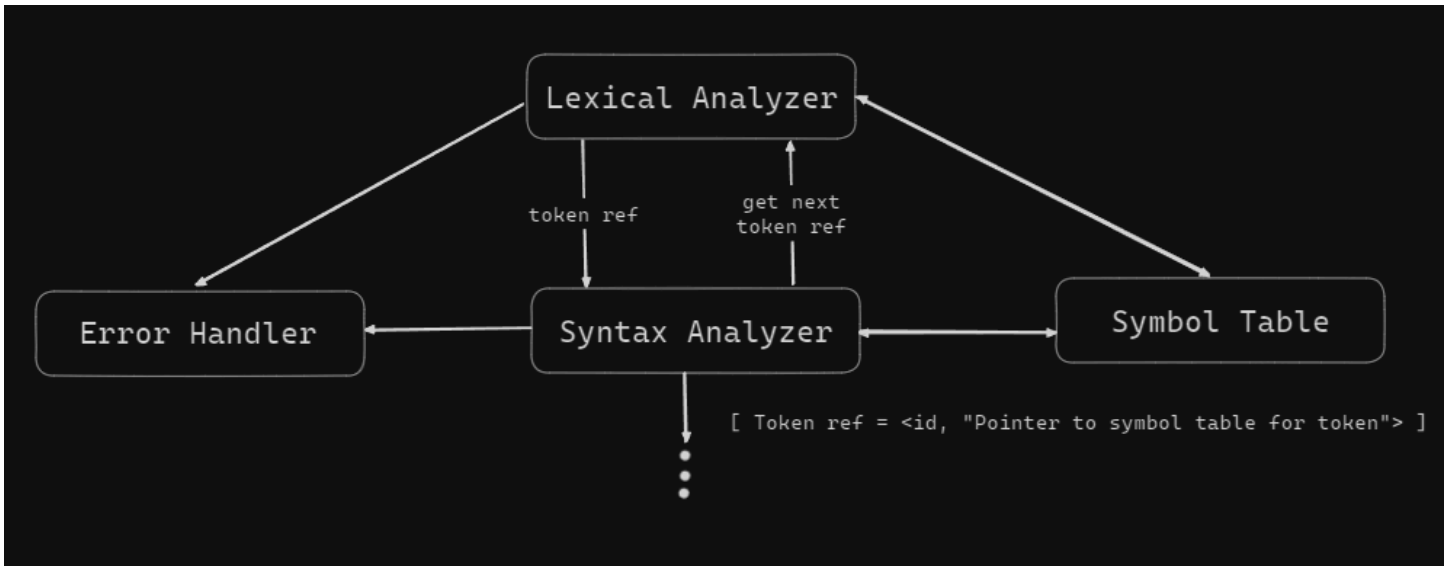
Intro:

- As the name suggests, It takes the tokens provided by lexical analyzer from the symbol table and verifies whether the tokens are in correct syntax or not, according to the Grammar of the programming language.
- As the output, the Syntax Analyzer generates a parse tree. That is, the syntax is checked by generating a parse tree for each token.
- A parse tree is a hierarchical structure of a lexeme built by referencing the grammar rules. (production rules)
- If a parse tree for a token is fully constructed, it means that the syntax is correct (token satisfies the grammar), else the Syntax analyzer informs about the syntax error to error handler, with line number and position at which the syntax is mismatched.
- The parsing methods are of two types, based on the direction of parsing:
    a. Top-Down Parsing (ex: LL Parser)
    b. Bottom-Up Parsing (ex: LR Parser)
- The grammar considered by the parser of a compiler, is often in the form of CFG (Context-Free-Grammer).
- So, Inshort, a syntax-analyzer (parser) functions are:
    - Ensuring Syntax Validity: Checks if the syntaxes of lexemes are correct.
    - Constructing a parse tree: Hierarchical Representation of a lexeme w.r.t the grammar.
    - Syntax Error Detection: Identifying Syntax Errors and reports it to the Error Handler of Compiler.



CFG: (Notes)