BAHIRDAR INSTUETE OF TECHNOLOGY

FACULITY OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

PRINCIPLE OF COMPILER DESIGN

INDIVUDUAL PROJECT

ERMIYAS MISGANAW…………………………………………………………………..1505942

QUESTION -30

How does lexical analysis integrate with syntax analysis?

HOW DOES LEXICAL ANALYSIS INTEGRATE WITH SYNTAX ANALYSIS

➲Lexical analysis and syntax analysis are two consecutive phases in the compilation process, and they are tightly integrated. Here's how they work together.

1. Lexical Analyzer Feeds Tokens to the Parser

- The **lexical analyzer** (scanner) reads the source program character by character and groups them into meaningful sequences called **lexemes**, which are then classified into **tokens**.
- Each token is passed to the **syntax analyzer** (parser) for further processing.

"The parser takes the token produced by lexical analysis and builds the syntax tree."

2. Tokens Form the Terminal Symbols of the Grammar

- Tokens such as id, number, if, relop, etc., become the **terminal symbols** in the context-free grammar used by the parser.

3. Lexical Analyzer Strips Out Non-Essential Elements

- Before passing tokens to the parser, the lexical analyzer removes **comments, whitespace, and other separators**, simplifying the parser's job.

4. Symbol Table Interaction

- Both the lexical analyzer and the parser interact with the **symbol table**.
- The lexer may enter identifiers into the symbol table; the parser uses this table for semantic checks and context management.
- **Reference**:

5. Error Handling Coordination

- Lexical errors (e.g., illegal characters) are caught by the lexer; syntactic errors (e.g., misplaced tokens) are caught by the parser.
- The lexer can help in error recovery by skipping invalid characters and resynchronizing token streams.

6. Regular Expressions (Lexical) vs. Context-Free Grammars (Syntax)

- Lexical tokens are specified using **regular expressions**.
- Syntax structures are specified using **context-free grammars**.
- The lexer recognizes small-scale patterns; the parser recognizes large-scale grammatical structures.

7. Example of Integration

From *Ch2_Lexical Analysis.pdf, Page 32* and *Ch3_Syntax Analysis.pdf, Page 34*:

- Lexer tokenizes:
  if (i == j) z = 0; else z = 1;
  → Tokens: if, (, id, relop, id,), id, =, number, ;, else, id, =, number, ;
- Parser uses grammar:

  ```text
  stmt → if expr then stmt else stmt
  expr → term relop term
  term → id | number
  ```

- The parser builds a parse tree using these tokens as terminals.

Summary of Integration

Aspect	Lexical Analyzer	Syntax Analyzer
Input	Source program (character stream)	Token stream from lexer
Output	Token stream	Parse tree / syntax tree
Specification	Regular expressions	Context-free grammar
Errors handled	Invalid characters, malformed tokens	Invalid token sequences, grammar violations
Symbol table	Enters identifiers	Uses for context and type checking

lexical analysis acts as a **front-end filter** for syntax analysis. It transforms raw source code into a structured token stream, allowing the parser to focus on grammatical structure without being burdened by low-level character processing. The two phases are connected via a clean interface: tokens and the symbol table.