COMPILER

B. HARSHITHA
19551A1067

LEXICAL ANALYSER

→ also called as scanner
→ Reach the successive line.
→ It breaks into terms like identifier, operator, delimiter.
→ Analyses constructs symbol table.
→ The symbol table allocates memory. TOOLS: LEX

How it works?

1. Input Processing: This stage involves cleaning up input text and preparing it for lexical analysis. This may include removing comments, whitespace and other non-essential characters from input text.
2. Tokenization: This is the process of breaking input text into sequence of tokens. This is usually done by matching characters in input text against set of patterns.
3. Token classification: The lexer checks that each token is valid according to rules of programming language.
4. Token Validation: Lexer determines type of each token. Might classify keywords, identifiers, operators.
5. Output generation: Lexer generates output of the lexical analysis process which is typically a list of tokens.

SYNTACTIC ANALYSER

→ The syntactic analyser refers to the expression, statement, declaration identified.
→ It is aided by formal grammar by programming language.
→ also called as parsing.

★ PURPOSE: It to draw exact meaning.
→ checks text for meaningfulness comparing to rules of grammar.

★ Create parse tree or abstract syntax tree of source code which is a hierarchical representation of source code that reflects grammatical structure of program.

Features: • syntax tree • Context free Grammar
• Top-down & bottom-up parsing • Error detection
• Intermediate code generation • optimization

ADVANTAGES

1. Structural validation: Syntax analysis allows compiler to check if source code follows grammatical rules of programming language, which helps to detect & report errors in source code.
2. Improved code generation: Syntax analyser can generate in a parse tree or abstract syntax tree of source code.

SEMANTIC ANALYSER

→ It is also called as phase bridge.
→ Analysis phase of syntax.
→ Last phase of translation is code generation.

Errors recognized are
• Type mismatch • Undeclared variables
• Reserved identifier misuse.

FUNCTION:

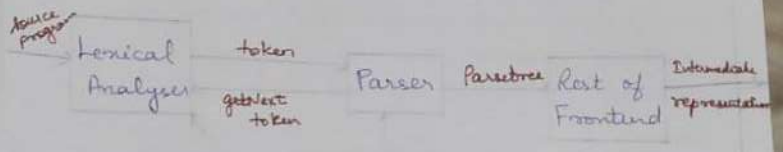
1. Type checking - Ensures that data types are used in a way consistent with their definition.
2. Label checking - A program should contain labels references.
3. Flow control check - keeps a check that control structures are used in a proper manner.

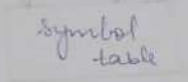
TYPES: 1. Static semantic - These are checked at compile time.
2. Dynamic semantic - Different units of program like expression & statements

★ Makes sure that declarations & statements of program are semantically correct. Collection of procedures which is called by parser as & when required by grammar.
★ Syntax tree & symbol table to check whether the given program is semantically consistent.

LEXICAL

SYNTACTIC ANALYSER





SEMANTIC ANALYSER

assignment statement