Lab Activity for 24th Oct 2, 2022

094230

Front-end constitutes the **Lexical analyzer**, **semantic analyzer**, **syntax analyzer**, and **intermediate code generator**

Use a string example e.g. x+2*y to show the work of the various phases of the front end.

Phase 1: Lexical Analysis

Lexical Analysis is the first phase when the compiler scans the source code. This process can be left to right, character by character, and group these characters into tokens.

Here, the character stream from the source program is grouped in meaningful sequences by identifying the tokens. It makes the entry of the corresponding tickets into the symbol table and passes that token to the next phase.

The primary functions of this phase are:

1. Identify the lexical units in a source code
2. Classify lexical units into classes like constants, reserved words, and enter them in different tables. It will Ignore comments in the source program
3. Identify token which is not a part of the language

EXAMPLE

X + 2 * Y

X - IDENTIFIER

+ - ADDITION OPERATOR

2 - NUMBER

* - MULTIPLICATION OPERATOR

Y – IDENTIFIER

Phase 2: Syntax Analysis

Syntax analysis is all about discovering structure in code. It determines whether or not a text follows the expected format. The main aim of this phase is to make sure that the source code written by the programmer is correct or not.

Syntax analysis is based on the rules based on the specific programming language by constructing the parse tree with the help of tokens. It also determines the structure of source language and grammar or syntax of the language.

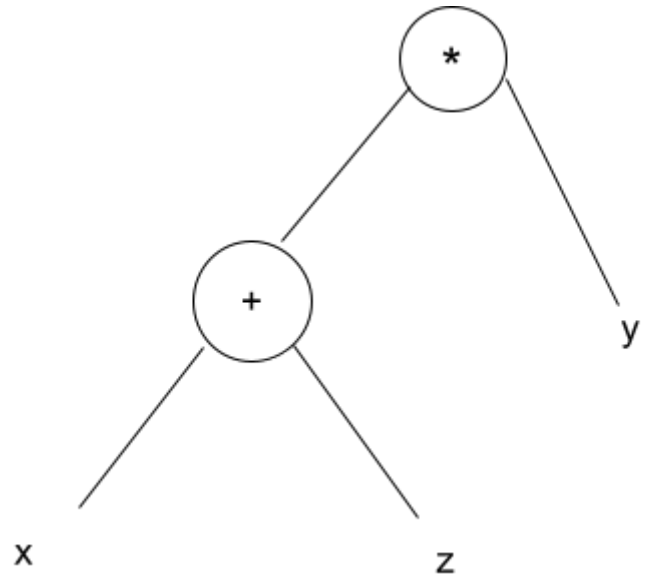
Here, is a list of tasks performed in this phase:

1. Obtain tokens from the lexical analyzer
2. Checks if the expression is syntactically correct or not
3. Report all syntax errors
4. Construct a hierarchical structure which is known as a parse tree

EXAMPLE

Consider parse tree for the following example

X + 2 * Y



Phase 3: Semantic Analysis

Semantic analysis checks the semantic consistency of the code. It uses the syntax tree of the previous phase along with the symbol table to verify that the given source code is semantically consistent. It also checks whether the code is conveying an appropriate meaning.

Semantic Analyzer will check for Type mismatches, incompatible operands, a function called with improper arguments, an undeclared variable, etc.

Functions of Semantic analysis phase are:

1) Helps you to store type information gathered and save it in symbol table or syntax tree
2) Allows you to perform type checking
3) In the case of type mismatch, where there are no exact type correction rules which satisfy the desired operation a semantic error is shown
4) Collects type information and checks for type compatibility
5) Checks if the source language permits the operands or not

EXAMPLE

X + 2 * Y

In the above code, the semantic analyzer will perform type checking

Phase 4: Intermediate Code Generation

Once the semantic analysis phase is over the compiler generates intermediate code for the target machine. It represents a program for some abstract machine.

Intermediate code is between the high-level and machine level language. This intermediate code needs to be generated in such a manner that makes it easy to translate it into the target machine code.

Functions on Intermediate Code generation:

1) It should be generated from the semantic representation of the source program
2) Holds the values computed during the process of translation
3) Helps you to translate the intermediate code into target language
4) Allows you to maintain precedence ordering of the source language
5) It holds the correct number of operands of the instruction

EXAMPLE

X + 2 * Y

t1 = 2

t2 = Y

t3 = t1 * t2

t4 = X

t5 = t4 + t3