MCA3 MSc3_Phases of compiler design_Dr.Ramji Yadav

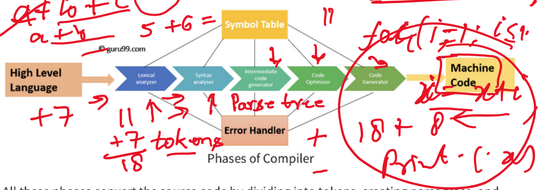
Thursday, 29 October 2020 11:58 AM

What are the Phases of Compiler Design?

Compiler operates in various phases each phase transforms the source program from one representation to another. Every phase takes inputs from its previous stage and feeds its output to the next phase of the compiler.

There are 6 phases in a compiler. Each of this phase help in converting the high-level langue the machine code. The phases of a compiler are:

1. Lexical analysis
2. Syntax analysis
3. Semantic analysis
4. Intermediate code generator
5. Code optimizer
6. Code generator



Phases of Compiler

All these phases convert the source code by dividing into tokens, creating parse trees, and optimizing the source code by different phases.

All these phases convert the source code by dividing into tokens, creating parse trees, and optimizing the source code by different phases.

In this tutorial, you will learn:

Phase 1: Lexical Analysis

Lexical Analysis is the first phase when 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 next phase.

The primary functions of this phase are:

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

Example:

x = y + 10

Tokens

X	identifier
=	Assignment operator
Y	identifier
+	Addition operator
10	Number

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 was written by the programmer is correct or not.

Syntax analysis is based on the rules based on the specific programing 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:

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

Example

Any identifier/number is an expression

If x is an identifier and y+10 is an expression, then x= y+10 is a statement.

Consider parse tree for the following example

```
(a+b)*c
```

Consider parse tree for the following example

```
(a+b)*c
```