ID: 171-35-1912

Ans. To The Q. NO: 3

My throughout about the "Compiler Design" course, describing below:

Compiler: A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or code that a computer's processor uses.

A programmer writes language statement in a language such as "C" one line at a time using an editor. The file that is created contains what are called source statements. The programmer then runs the appropiate language compiler, specifying the name of the file that contains the source statements.

When executing, the compiler first parses all of the language statements syntactically one after the other and the, in one or more successive stage or passes builds the output code, making sure that statements that refer to other statements are reffered to correctly in the final code. So, the output of the compilation has been called object code or sometimes an object module. The object code is machine code that the processor can execute one instruction at a time.

Phase: Compiler operates in various phases, each phase transforms the source program from one representation to another.

There are six phases in a compiler.

1. Lexical ~~Aa~~ Analysis: Lexical Analysis is the first phase when compiler scans the source code. This process can left to right, character by character and group these characters into tokens.
   It's identify the lexical units in a source code.
   It's classify lexical units into classes like contents, reserved words and enter them in different tables.
   It's identify token which is not a part of the language.

ID: 171-35-1912

2. Syntax Analysis: Syntax Analysis is all about discovering structure in code. It determines whether or not a text follows the expected format.

It's obtain tokens from the lexical analyzer.

It's checks if the expression is syntatically correct or not.

It's report all syntax errors.

It's construct a ~~hire~~ hierarchical structure which is know as parse tree.

3. Semantic Analysis: Semantic Analysis check the semantic consistency of the code.

It's helps to store type information gathered and save it symbol table or syntax tree.

It's allows to perform type checking.

It's checks if the source language permits the operands or not.

4. Intemediate 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.

ID: 171-35-1912

It's holds the values computed during the process of translation.

It's allows to maintain precedence ordering of the source language.

It's holds the correct number of ~~pe~~ operands of the instruction.

5. Code Optimization: This phase removes unnecessary code line and arranges the sequence of statements to speed up the execution of the program.

It's helps to ~~east~~ establish a trade-off between execution and compilation speed.

It's improves the running time of the target program.

It's generates streamlined code still in intemediate representation.

ID: 171-35-1912

6. <u>Code Generation:</u> It's the last and final phase of a compiler. It gets input from code optimization phases and produces the page code or object code as a result. The objective of this phase is to allocate storage and generate relocatable machine code.

It also allocates memory locations for the variable.

The target language is the machine code. All the memory locations and registers are also selected and allotted during this phase. The code generated by this ~~p~~ phase is executed to take inputs and generate expected outputs.