Difference Between Compiler and Interpreter

A compiler and an interpreter are two different types of software programs used for converting human-readable source code to machine-executable code, but they operate in different ways. Compiler and interpreter are important topics in the GATE CSE syllabus. A computer program is usually written in a high-level language (source code) which is further converted into machine language using compilers and interpreters.

Key Differences Between Compiler and Interpreter

Compiler	Interpreter
The compiler examines the entire program and converts it all at once into machine code.	The interpreter only converts one statement at a time into machine code.
Because the compiler reads the code all at once, any errors (if any) are displayed at the conclusion.	Errors in the interpreter are displayed line by line since it reads code one line at a time.
In the case of compilers, the program code is already translated into machine code, so the code execution time is significantly reduced.	Even for a beginner, interpreters are simple to use and perform.
The compilation and the execution of the program can be separated. As a result, you can only do it once you've finished compiling the complete output.	The program's execution is one of the phases in the interpretation process. As a result, you may do it line by line.
While compiling, a compiler displays all errors and warnings. As a result, you won't be able to execute this software until you correct the issues.	An interpreter reads each sentence and shows them if any problems are found. To comprehend the next line, the user must correct these inaccuracies.
Compilers are used Java, Scala, C#, C, and C++.	Interpreters have used Perl, Ruby, and PHP.

Difference between Compiler and Interpreter PDF

A compiler translates the entire source code into machine code that can be executed multiple times without recompilation, while an interpreter translates the code line by line at runtime without producing a standalone executable file. Download the PDF from the direct link given below:

Compiler and Interpreter

A compiler will check to see whether all language statements are valid. It will provide an error notice if it comes across something that is wrong. The compiler will transform the source code into machine code if no mistakes are found. The compiler assembles the various code files into executable programs, such as exe. Finally, the program is launched.

An interpreter creates the program. It does not create machine code or connect files. Interpreters are important as per the GATE exam. During the program's execution, the source statements are implemented line by line.

What is a Compiler?

A compiler is computer software that converts high-level programming languages (source code) into a machine-readable format known as low-level programming languages (machine code). It reads the entire source code, compiles it, and then turns it into an executable file, which the user performs or runs for the command designed to be carried out. And, if there are any faults, it returns them all at once while reading the source code.

It is quicker than the interpreter since the source code has already been built, and we simply need to run the executable file that has been created. The compiler generates secure executable files that may be run on any of your customers or other computers without the requirement for actual source code. As a result, your software is unhackable, safe, and private. For running the shared executable file of your source code, your client or anybody else does not require the installation of any compiler, interpreter, or third-party application on their machine.

What is an Interpreter?

Interpreters, like compilers, perform the same function. It can also convert high-level languages to low-level ones. However, unlike a compiler, an interpreter analyses the source code line by line and informs you if there is a mistake simultaneously, making it easier to debug but slower than a compiler.

We directly share the source code in interpreted languages, which may run on any machine without system incompatibility issues. Code analysis is easier with interpreters since they read the code line by line and return the error message immediately. In addition, if the client has access to the source code, they may quickly debug or alter it. Interpreters, unlike compilers, do not create new distinct files. So it doesn't take up any more memory. The execution control interpreter reads code line by line, allowing you to pause and alter the code at any time.

Conclusion: Key Differences between Compilers and Interpreters

The main differences between a compiler and an interpreter are:

- A compiler translates entire source code written in a high-level programming language into an executable file that can run on a specific hardware or operating system. On the other hand, an interpreter reads and executes the code line by line, translating each line of code into machine code at runtime.
- A compiler produces a stand-alone executable file that can be executed multiple times without needing to recompile the source code. On the other hand, an interpreter executes the code on the fly, without producing any stand-alone executable file.
- A compiled program typically runs faster than an interpreted program, as the executable code generated by the compiler is optimized for the target hardware and can be executed directly. In contrast, an interpreted program needs to be translated and executed line by line at runtime, which can be slower.
- Debugging an interpreted program is easier than debugging a compiled program. In an interpreter, you can see the result of each line of code in real time, whereas, with a compiler, you need to run the entire program to identify the errors.