|  |  |  |
| --- | --- | --- |
| **Parameter** | **Compiler** | **Interpreter** |
| 1. Steps of Programming | * Creation of the program. * The Compiler analyses all the language statements and throws an error when it finds something incorrect. * If there’s zero error, the compiler converts the source code to machine one. * It links various code files into a runnable program (execution). * It runs the program. | * Creation of the program. * It doesn’t require the linking of files or generation of machine code. * It executes the source statements line by line during the execution. |
| 1. Advantage | The code execution time is comparatively less because the program code already gets translated into machine code. | They are fairly easy to use and execute, even for a beginner. |
| 1. Disadvantage | One can’t change a program without getting back to the source code. | Only computers with the corresponding Interpreter can run the interpreted programs. |
| 1. Machine Code | It stores the machine language on the disk in the form of machine code. | It doesn’t save the machine language at all. |
| 1. Running Time | The compiled codes run comparatively faster. | The interpreted codes run comparatively slower. |
| 1. Model | It works on the basis of the language-translation linking-loading model. | It works on the basis of the Interpretation method. |
| 1. Execution | One can separate the program execution from the compilation. Thus, you can perform it only after completing the compilation of the entire output. | Execution of the program is one of the steps of the Interpretation process. So, you can perform it line by line. |
| 1. Memory Requirement | Target programs execute independently. They don’t require the Compiler in the memory. | Interpreter originally exists in the memory at the time of interpretation. |
| 1. Optimization of Code | A compiler is capable of seeing the entire code upfront. Thus, it makes the codes run faster by performing plenty of optimizations. | An interpreter sees a code line by line. The optimization is, thus, not very robust when compared to Compilers. |
| 1. Dynamic Typing | Compilers are very difficult to implement because they can’t predict anything that happens during the turn time. | The Interpreted language supports Dynamic Typing. |
| 1. Use | It works best for the Production Environment. | It works the best for the programming and development environment. |
| 1. Input | A Compiler takes a program as a whole. | An Interpreter takes single lines of a code. |
| 1. Output | The Compilers generate intermediate machine codes. | The Interpreters never generate any intermediate machine codes. |
| 1. Errors | This translator displays all the errors after compiling- together at the same time. | It displays the errors of every single line one by one. |
| 1. Programming Languages | Java, Scala, C#, C, C++ use Compilers. | Perl, Ruby, PHP use Interpreters. |