**DIFFERENCE BETWEEN COMPILER AND INTERPRETER**

|  |  |  |
| --- | --- | --- |
| **Difference Types** | **COMPILER** | I**NTERPRETER** |
| Programming Steps | * Write a program in source code. * Compile will analyze your program statements and check their correctness. If an error is found in a program, it throws an error message. * If the program contains no error, then the compiler will convert the source code program into machine code. * The compiler links all the code files into a single runnable program, which is known as the exe file. * Finally, it runs the program and generates output. | * Write a program in source code. * No linking of files happens, or no machine code will generate separately. * The source code programming statements are executed line-by-line during their execution. If an error is found at any specific statement interpreter, it stops further execution until the error gets removed. |
| Translation type | A compiler translates complete high-level programming code into machine code at once. | An interpreter translates one statement of programming code at a time into machine code. |
| Advantage | As the source code is already converted into machine code, the code execution time becomes short. | As the source code is interpreted line-by-line, error detection and correction become easy. |
| DisAdvantage | If you want to change your program for any reason, either by error or logical changes, you can do it only by going back to your source code. | Interpreted programs can run on only those computers which have the same interpreter. |
| Machine code | It stores the converted machine code from your source code program on the disk. | It never stores the machine code at all on the disk. |
| Running time | A compiler takes an enormous time to analyze source code. However, overall compiled programming code runs faster as compression to an interpreter. | An interpreter takes less time to analyze source code as compared to a compiler. However, overall interpreted programming code runs slower as compression to the compiler. |
| Program generation | The compiler generates an output of a program (in the form of an exe file) that can run separately from the source code program. | The interpreter doesn't generate a separate machine code as an output program. So it checks the source code every time during the execution. |
| Execution | The process of program execution takes place separately from its compilation process. Program execution only takes place after the complete program is compiled. | The process of program execution is a part of interpretation steps, so it is done line-by-line simultaneously. |
| Memory requirement | A compiled program is generated into an intermediate object code, and it further required linking. So there is a requirement for more memory. | An interpreted program does not generate an intermediate code. So there is no requirement for extra memory. |
| Best suited for | The compiled program is bounded to the specific target machine. It requires the same compiler on the machine to execute; C and C++ are the most popular programming language based on the compilation model. | In web environments, compiling takes place relatively more time to run even small code, which may not run multiple times. As load time is essential in the web environment, interpreters are better. JavaScript, Python, Ruby are based on the interpreter model. |
| Error execution | The compiler shows the complete errors and warning messages at program compilation time. So it is not possible to run the program without fixing program errors. Doing debugging of the program is comparatively complex while working with a compiler. | An interpreter reads the program line-by-line; it shows the error if present at that specific line. You must have to correct the error first to interpret the next line of the program. Debugging is comparatively easy while working with an Interpreter. |