Day 01 Assignment

Q.1 - Mention the difference between interpreter and compiler.

|  |  |  |
| --- | --- | --- |
| PARAMETER | COMPILER | INTERPRETER |
| 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 (exe). * 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. |
| 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. |
| 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. |
| 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. |
| Running Time | The compiled codes run comparatively faster. | The interpreted codes run comparatively slower. |
| Model | It works on the basis of the language-translation linking-loading model. | It works on the basis of the Interpretation method. |
| Generation of Program | It generates an output program in the exe format. A user can run it independently from the originally intended program. | It doesn’t generate an output program. Meaning, it evaluates the source program every time during individual execution. |
| 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. |
| |  |  | | --- | --- | | Memory  requirement | Target programs execute | | Target programs execute independently. They don’t require the Compiler in the memory. | Interpreter originally exists in the memory at the time of interpretation. |
| |  | | --- | | Best Fitted For | | You cannot port the Compiler because it stays bound to the specific target machine. The compilation model is very common in programming languages like C and C++. | They work the best in web environments- where the load time is very crucial. Compiling takes a relatively long time, even with small codes that may not run multiple times due to the exhaustive analysis. Interpretations are better in such cases. |
| 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. |
| 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. |
| Use | It works best for the Production Environment. | It works the best for the programming and development environment. |
| Execution of Error | A Compiler displays every error and warning while compiling. So, you can’t run this program unless you fix the errors. | An Interpreter reads every statement, then displays the errors, if any. A user must resolve these errors in order to interpret the next line. |
| Input | A Compiler takes a program as a whole. | An Interpreter takes single lines of a code. |
| Output | The Compilers generate intermediate machine codes. | The Interpreters never generate any intermediate machine codes. |
| 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. |
| Programming Languages | Java, Scala, C#, C, C++ use Compilers. | Perl, Ruby, PHP use Interpreters. |