

The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. Also, specific user environment and usage history can make it difficult to reproduce the problem. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Programming languages are essential for software development. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Techniques like Code refactoring can enhance readability. However, readability is more than just programming style. One approach popular for requirements analysis is Use Case analysis. Techniques like Code refactoring can enhance readability. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. Different programming languages support different styles of programming (called programming paradigms).