

In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. It is usually easier to code in "high-level" languages than in "low-level" ones.

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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Programmable devices have existed for centuries. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Programmable devices have existed for centuries. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. Programming languages are essential for software development. 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. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Computer programmers are those who write computer software. 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. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.