

In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. 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. There exist a lot of different approaches for each of those tasks. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Integrated development environments (IDEs) aim to integrate all such help. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Unreadable code often leads to bugs, inefficiencies, and duplicated code. However, readability is more than just programming style. Many applications use a mix of several languages in their construction and use. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. Programmable devices have existed for centuries. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Programs were mostly entered using punched cards or paper tape. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.