

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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Many applications use a mix of several languages in their construction and use. 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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. Many applications use a mix of several languages in their construction and use. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.