

Programming languages are essential for software development. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. 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. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Programming languages are essential for software development. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. One approach popular for requirements analysis is Use Case analysis. 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Following a consistent programming style often helps readability. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Computer programmers are those who write computer software. Programming languages are essential for software development.