

Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. 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). Programs were mostly entered using punched cards or paper tape. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Code-breaking algorithms have also existed for centuries. Ideally, the programming language best suited for the task at hand will be selected. 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. However, readability is more than just programming style. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Many applications use a mix of several languages in their construction and use. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. There exist a lot of different approaches for each of those tasks. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Following a consistent programming style often helps readability. It is very difficult to determine what are the most popular modern programming languages. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Unreadable code often leads to bugs, inefficiencies, and duplicated code.