

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. 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. Ideally, the programming language best suited for the task at hand will be selected. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. 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. Use of a static code analysis tool can help detect some possible problems. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. 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. 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. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. 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. 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. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Also, specific user environment and usage history can make it difficult to reproduce the problem. Code-breaking algorithms have also existed for centuries. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. However, readability is more than just programming style.