

It is usually easier to code in "high-level" languages than in "low-level" ones. Computer programmers are those who write computer software. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Code-breaking algorithms have also existed for centuries. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. 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. 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. 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. There are many approaches to the Software development process. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. There exist a lot of different approaches for each of those tasks. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. There are many approaches to the Software development process. Code-breaking algorithms have also existed for centuries.