

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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Computer programmers are those who write computer software. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. There are many approaches to the Software development process. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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). Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. 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. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.