

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. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Integrated development environments (IDEs) aim to integrate all such help. However, readability is more than just programming style. It is usually easier to code in "high-level" languages than in "low-level" ones. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Also, specific user environment and usage history can make it difficult to reproduce the problem. Ideally, the programming language best suited for the task at hand will be selected. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. 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 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.