

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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.

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. Computer programmers are those who write computer software. 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. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. It is usually easier to code in "high-level" languages than in "low-level" ones. Programs were mostly entered using punched cards or paper tape. 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. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. Scripting and breakpointing is also part of this process. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Also, specific user environment and usage history can make it difficult to reproduce the problem.