

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. Use of a static code analysis tool can help detect some possible problems. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. 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. Computer programmers are those who write computer software. 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. 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. However, readability is more than just programming style. 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, readability is more than just programming style. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Normally the first step in debugging is to attempt to reproduce the problem. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Scripting and breakpointing is also part of this process.