

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. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. 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. 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. Scripting and breakpointing is also part of this process. Integrated development environments (IDEs) aim to integrate all such help. 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. Many applications use a mix of several languages in their construction and use. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Use of a static code analysis tool can help detect some possible problems. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Many applications use a mix of several languages in their construction and use. 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. Different programming languages support different styles of programming (called programming paradigms). 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. Programming languages are essential for software development. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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*. Use of a static code analysis tool can help detect some possible problems. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. 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.