

After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. 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. 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. Normally the first step in debugging is to attempt to reproduce the problem. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. It is very difficult to determine what are the most popular modern programming languages. 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. 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. 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. Different programming languages support different styles of programming (called programming paradigms). Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Code-breaking algorithms have also existed for centuries. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Following a consistent programming style often helps readability. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.