

Different programming languages support different styles of programming (called programming paradigms). Programmable devices have existed for centuries. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. It is very difficult to determine what are the most popular modern programming languages. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Also, specific user environment and usage history can make it difficult to reproduce the problem. There are many approaches to the Software development process. 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. Following a consistent programming style often helps readability. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Integrated development environments (IDEs) aim to integrate all such help. 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. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. 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. Many applications use a mix of several languages in their construction and use. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. 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. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.