

Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Normally the first step in debugging is to attempt to reproduce the problem. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. One approach popular for requirements analysis is Use Case analysis. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Unreadable code often leads to bugs, inefficiencies, and duplicated code. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Techniques like Code refactoring can enhance readability. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Code-breaking algorithms have also existed for centuries. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.