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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. 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. Ideally, the programming language best suited for the task at hand will be selected. It affects the aspects of quality above, including portability, usability and most importantly maintainability. There exist a lot of different approaches for each of those tasks. Programmable devices have existed for centuries. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). There exist a lot of different approaches for each of those tasks. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. Programs were mostly entered using punched cards or paper tape. Also, specific user environment and usage history can make it difficult to reproduce the problem.