

Programs were mostly entered using punched cards or paper tape. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Many applications use a mix of several languages in their construction and use. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Following a consistent programming style often helps readability. Different programming languages support different styles of programming (called programming paradigms). 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. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Computer programmers are those who write computer software. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. 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. 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. 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. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.