

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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Programs were mostly entered using punched cards or paper tape. 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. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. However, readability is more than just programming style. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). 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. There exist a lot of different approaches for each of those tasks. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. 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. Integrated development environments (IDEs) aim to integrate all such help. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Code-breaking algorithms have also existed for centuries. Following a consistent programming style often helps readability. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).