

There are many approaches to the Software development process. Following a consistent programming style often helps readability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. 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. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Different programming languages support different styles of programming (called programming paradigms). In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). 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 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Techniques like Code refactoring can enhance readability. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. It is very difficult to determine what are the most popular modern programming languages. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.