Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. There are many approaches to the Software development process. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Normally the first step in debugging is to attempt to reproduce the problem. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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). 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. 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. Normally the first step in debugging is to attempt to reproduce the problem. 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. One approach popular for requirements analysis is Use Case analysis. 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. Normally the first step in debugging is to attempt to reproduce the problem. 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.