In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. One approach popular for requirements analysis is Use Case analysis. Ideally, the programming language best suited for the task at hand will be selected. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. One approach popular for requirements analysis is Use Case analysis. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. Programmable devices have existed for centuries.