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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). One approach popular for requirements analysis is Use Case analysis. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. 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. Following a consistent programming style often helps readability. 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. Following a consistent programming style often helps readability. Use of a static code analysis tool can help detect some possible problems. Unreadable code often leads to bugs, inefficiencies, and duplicated code. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. It is usually easier to code in "high-level" languages than in "low-level" ones. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. One approach popular for requirements analysis is Use Case analysis. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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). Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Following a consistent programming style often helps readability.