Unreadable code often leads to bugs, inefficiencies, and duplicated code. Also, specific user environment and usage history can make it difficult to reproduce the problem. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. 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). 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. Computer programmers are those who write computer software. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Programs were mostly entered using punched cards or paper tape. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. Code-breaking algorithms have also existed for centuries.