

Code-breaking algorithms have also existed for centuries. 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. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the *Book of Ingenious Devices*. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. One approach popular for requirements analysis is Use Case analysis. Code-breaking algorithms have also existed for centuries. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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). 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. 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. Code-breaking algorithms have also existed for centuries. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.