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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Programmable devices have existed for centuries. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Use of a static code analysis tool can help detect some possible problems. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Following a consistent programming style often helps readability. 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. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. 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. Code-breaking algorithms have also existed for centuries. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.