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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. Integrated development environments (IDEs) aim to integrate all such help. 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. Techniques like Code refactoring can enhance readability. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. 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. 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. Scripting and breakpointing is also part of this process. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Programming languages are essential for software development. Also, specific user environment and usage history can make it difficult to reproduce the problem. Computer programmers are those who write computer software. 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.