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. 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. Integrated development environments (IDEs) aim to integrate all such help. However, readability is more than just programming style. 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. 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. However, readability is more than just programming style. Many applications use a mix of several languages in their construction and use. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Programs were mostly entered using punched cards or paper tape. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. 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. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. Use of a static code analysis tool can help detect some possible problems. 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. 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. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Programs were mostly entered using punched cards or paper tape. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. 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. Ideally, the programming language best suited for the task at hand will be selected.