High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Techniques like Code refactoring can enhance readability. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Integrated development environments (IDEs) aim to integrate all such help. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. There are many approaches to the Software development process. Whatever the approach to development may be, the final program must satisfy some fundamental properties. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. 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. Programming languages are essential for software development. There exist a lot of different approaches for each of those tasks. 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. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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.