Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Normally the first step in debugging is to attempt to reproduce the problem. 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). This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Normally the first step in debugging is to attempt to reproduce the problem. 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. Integrated development environments (IDEs) aim to integrate all such help. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Techniques like Code refactoring can enhance readability. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Programming languages are essential for software development. Programmable devices have existed for centuries. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. 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. There exist a lot of different approaches for each of those tasks. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. 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. Techniques like Code refactoring can enhance readability. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. 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.