

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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Also, specific user environment and usage history can make it difficult to reproduce the problem. Following a consistent programming style often helps readability. 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. 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. Use of a static code analysis tool can help detect some possible problems. It is usually easier to code in "high-level" languages than in "low-level" ones. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. Following a consistent programming style often helps readability. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. Integrated development environments (IDEs) aim to integrate all such help. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Ideally, the programming language best suited for the task at hand will be selected.