

Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. One approach popular for requirements analysis is Use Case analysis. Also, specific user environment and usage history can make it difficult to reproduce the problem. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Following a consistent programming style often helps 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. However, readability is more than just programming style. Different programming languages support different styles of programming (called programming paradigms). 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. 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. Scripting and breakpointing is also part of this process. Integrated development environments (IDEs) aim to integrate all such help. 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. Code-breaking algorithms have also existed for centuries. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. Code-breaking algorithms have also existed for centuries. 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.