

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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. 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, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. Computer programmers are those who write computer software. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. 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. Different programming languages support different styles of programming (called programming paradigms). Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Also, specific user environment and usage history can make it difficult to reproduce the problem. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. However, readability is more than just programming style. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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). Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.