

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. Following a consistent programming style often helps readability. Many applications use a mix of several languages in their construction and use. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. Programs were mostly entered using punched cards or paper tape. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. Computer programmers are those who write computer software. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. Integrated development environments (IDEs) aim to integrate all such help. 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. Following a consistent programming style often helps readability. 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. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. Programmable devices have existed for centuries. Different programming languages support different styles of programming (called programming paradigms). Code-breaking algorithms have also existed for centuries. Programming languages are essential for software development. There are many approaches to the Software development process. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.