

After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Integrated development environments (IDEs) aim to integrate all such help. Ideally, the programming language best suited for the task at hand will be selected. Computer programmers are those who write computer software. Computer programmers are those who write computer software. 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. 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 languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. 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. There are many approaches to the Software development process. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). There exist a lot of different approaches for each of those tasks. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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.