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. Techniques like Code refactoring can enhance readability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. Many applications use a mix of several languages in their construction and use. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. There exist a lot of different approaches for each of those tasks. Integrated development environments (IDEs) aim to integrate all such help. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them.

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. 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. 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 choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. There exist a lot of different approaches for each of those tasks. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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.