High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware..  
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.  
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.  
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.  
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.  
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.  
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.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
 Code-breaking algorithms have also existed for centuries.  
Ideally, the programming language best suited for the task at hand will be selected.  
 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).  
One approach popular for requirements analysis is Use Case analysis.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
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.