Programming languages are essential for software development.  
There exist a lot of different approaches for each of those tasks.  
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.  
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.  
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.  
Their jobs usually involve:  
 Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
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.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
  
 Computer programming is the process of performing particular computations (or more generally, accomplishing specific computing results), usually by designing and building executable computer programs.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.  
Use of a static code analysis tool can help detect some possible problems.