Assembly languages were soon developed that let the programmer specify instruction in a text format (e..g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
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.  
Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
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.  
Use of a static code analysis tool can help detect some possible problems.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Normally the first step in debugging is to attempt 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.  
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.  
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 following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
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.