It affects the aspects of quality above, including portability, usability and most importantly maintainability..  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
One approach popular for requirements analysis is Use Case analysis.  
 A similar technique used for database design is Entity-Relationship Modeling (ER Modeling).  
Scripting and breakpointing is also part of this process.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
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.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.  
 Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Programming languages are essential for software development.  
 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.  
Use of a static code analysis tool can help detect some possible problems.