It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
One approach popular for requirements analysis is Use Case analysis.  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line.  
There are many approaches to the Software development process.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
 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).  
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.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.