Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards..  
 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.  
 Code-breaking algorithms have also existed for centuries.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
 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.  
Use of a static code analysis tool can help detect some possible problems.  
One approach popular for requirements analysis is Use Case analysis.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.  
Scripting and breakpointing is also part of this process.  
However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.