These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics..  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
 Allen Downey, in his book How To Think Like A Computer Scientist, writes:  
 Many computer languages provide a mechanism to call functions provided by shared libraries.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
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.  
 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.  
One approach popular for requirements analysis is Use Case analysis.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.