Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages..  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 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.  
There are many approaches to the Software development process.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
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.  
 It is very difficult to determine what are the most popular modern programming languages.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
 Different programming languages support different styles of programming (called programming paradigms).  
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.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.