Ideally, the programming language best suited for the task at hand will be selected..  
 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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display.  
 A similar technique used for database design is Entity-Relationship Modeling (ER Modeling).  
 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.