Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Also, those involved with software development may at times engage in reverse engineering, which is the practice of seeking to understand an existing program so as to re-implement its function in some way.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
This is interpreted into machine code.  
Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language.  
Use of a static code analysis tool can help detect some possible problems.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.  
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.  
For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.  
Also, those involved with software development may at times engage in reverse engineering, which is the practice of seeking to understand an existing program so as to re-implement its function in some way.  
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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
To produce machine code, the source code must either be compiled or transpiled.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.