Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages..  
 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.  
 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.  
 A similar technique used for database design is Entity-Relationship Modeling (ER Modeling).  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
 New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation).  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
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.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
 Code-breaking algorithms have also existed for centuries.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.