However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
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.  
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.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
There exist a lot of different approaches for each of those tasks.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
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.  
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.  
Techniques like Code refactoring can enhance readability.  
 The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.