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..  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 Computer programmers are those who write computer software.  
There exist a lot of different approaches for each of those tasks.  
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.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 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.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
 Following a consistent programming style often helps readability.  
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.  
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.  
Integrated development environments (IDEs) aim to integrate all such help.