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, 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.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Use of a static code analysis tool can help detect some possible problems.  
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.  
Ideally, the programming language best suited for the task at hand will be selected.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
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.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
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.  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.