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.  
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.  
 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.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
Use of a static code analysis tool can help detect some possible problems.  
 Code-breaking algorithms have also existed for centuries.  
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.  
Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.  
Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic.  
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.  
Their jobs usually involve:  
 Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language.  
There exist a lot of different approaches for each of those tasks.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
Normally the first step in debugging is to attempt to reproduce the problem.  
 Different programming languages support different styles of programming (called programming paradigms).