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.  
Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 Programmable devices have existed for centuries.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
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.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.  
 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.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).