Many applications use a mix of several languages in their construction and use..  
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.  
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.  
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.  
 Computer programmers are those who write computer software.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
Integrated development environments (IDEs) aim to integrate all such help.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
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.  
Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers.  
 New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation).  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.