The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
Ideally, the programming language best suited for the task at hand will be selected.  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
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.  
 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).  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
Scripting and breakpointing is also part of this process.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
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.  
 Tasks accompanying and related to programming include testing, debugging, source code maintenance, implementation of build systems, and management of derived artifacts, such as the machine code of computer programs.  
 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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.