Unreadable code often leads to bugs, inefficiencies, and duplicated code..  
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.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
 Code-breaking algorithms have also existed for centuries.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
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).  
 It is very difficult to determine what are the most popular modern programming languages.  
Many applications use a mix of several languages in their construction and use.  
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.