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.  
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.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit.  
Scripting and breakpointing is also part of this process.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
To produce machine code, the source code must either be compiled or transpiled.  
 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).  
 Allen Downey, in his book How To Think Like A Computer Scientist, writes:  
 Many computer languages provide a mechanism to call functions provided by shared libraries.  
There exist a lot of different approaches for each of those tasks.  
Programming languages are essential for software development.  
Use of a static code analysis tool can help detect some possible problems.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.