Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
To produce machine code, the source code must either be compiled or transpiled.  
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.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
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.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.  
However, readability is more than just programming style.  
 It is very difficult to determine what are the most popular modern programming languages.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
However, readability is more than just programming style.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
It is usually easier to code in "high-level" languages than in "low-level" ones.