Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Normally the first step in debugging is to attempt to reproduce the problem.  
Use of a static code analysis tool can help detect some possible problems.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.  
There exist a lot of different approaches for each of those tasks.  
This is interpreted into machine code.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
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.  
In 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
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.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
Integrated development environments (IDEs) aim to integrate all such help.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
Techniques like Code refactoring can enhance readability.