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 following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
Use of a static code analysis tool can help detect some possible problems.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
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.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
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.  
Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
However, while these might be considered part of the programming process, often the term software development is more likely used for this larger overall process – whereas the terms programming, implementation, and coding tend to be focused on the actual writing of code.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.