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.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
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.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Scripting and breakpointing is also part of this process.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
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.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
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.  
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.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.