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..  
One approach popular for requirements analysis is Use Case analysis.  
However, readability is more than just programming style.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Many applications use a mix of several languages in their construction and use.  
 Computer programmers are those who write computer software.  
 Following a consistent programming style often helps readability.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line.  
 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).  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.  
 Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code.  
 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.  
Integrated development environments (IDEs) aim to integrate all such help.  
Normally the first step in debugging is to attempt to reproduce the problem.