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.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 Following a consistent programming style often helps readability.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
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.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
 Programmable devices have existed for centuries.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
Many applications use a mix of several languages in their construction and use.  
 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).