Normally the first step in debugging is to attempt to reproduce the problem..  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
 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.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
Techniques like Code refactoring can enhance readability.  
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.  
 Programs were mostly entered using punched cards or paper tape.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
 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.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
It is usually easier to code in "high-level" languages than in "low-level" ones.