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)..  
 Programs were mostly entered using punched cards or paper tape.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
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.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers.  
Use of a static code analysis tool can help detect some possible problems.  
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.  
Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Techniques like Code refactoring can enhance readability.