These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics..  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
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.  
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.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Ideally, the programming language best suited for the task at hand will be selected.  
As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
However, readability is more than just programming style.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
  
 Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks.  
 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.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.