Use of a static code analysis tool can help detect some possible problems..  
There exist a lot of different approaches for each of those tasks.  
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.  
 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.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
 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.  
 Programs were mostly entered using punched cards or paper tape.  
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.  
Programming languages are essential for software development.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
It is usually easier to code in "high-level" languages than in "low-level" ones.