Use of a static code analysis tool can help detect some possible problems.  
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.  
Relatedly, software engineering combines engineering techniques and principles with software development.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
One approach popular for requirements analysis is Use Case analysis.  
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).  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
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).  
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.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
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.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 It is very difficult to determine what are the most popular modern programming languages.  
Relatedly, software engineering combines engineering techniques and principles with software development.