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..  
 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.  
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.  
Programming languages are essential for software development.  
Ideally, the programming language best suited for the task at hand will be selected.  
 It is very difficult to determine what are the most popular modern programming languages.  
 Code-breaking algorithms have also existed for centuries.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Scripting and breakpointing is also part of this process.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Techniques like Code refactoring can enhance readability.  
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.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.