Following a consistent programming style often helps readability..  
 Programmable devices have existed for centuries.  
 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.  
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.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
 Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display.  
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.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
 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.  
Techniques like Code refactoring can enhance readability.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
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.  
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.  
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.