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.  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.  
Normally the first step in debugging is to attempt to reproduce the problem.  
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.  
 New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation).  
The following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
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.  
While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
 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.  
There exist a lot of different approaches for each of those tasks.  
Techniques like Code refactoring can enhance readability.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.