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.  
 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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
There exist a lot of different approaches for each of those tasks.  
Normally the first step in debugging is to attempt to reproduce the problem.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
One approach popular for requirements analysis is Use Case analysis.  
 Different programming languages support different styles of programming (called programming paradigms).  
 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).  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.