Normally the first step in debugging is to attempt to reproduce the problem.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
To produce machine code, the source code must either be compiled or transpiled.  
However, readability is more than just programming style.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
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.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.  
Techniques like Code refactoring can enhance readability.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
 Programmable devices have existed for centuries.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.