Normally the first step in debugging is to attempt to reproduce the problem.  
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.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
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.  
There exist a lot of different approaches for each of those tasks.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
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.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
This is interpreted into machine code.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
 Computer programmers are those who write computer software.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
Scripting and breakpointing is also part of this process.