Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code..  
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.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 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 first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
Integrated development environments (IDEs) aim to integrate all such help.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
Use of a static code analysis tool can help detect some possible problems.  
 Following a consistent programming style often helps readability.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.