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..  
Many applications use a mix of several languages in their construction and use.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
Ideally, the programming language best suited for the task at hand will be selected.  
Techniques like Code refactoring can enhance readability.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
One approach popular for requirements analysis is Use Case analysis.  
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.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
  
 Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks.