Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation..  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
One approach popular for requirements analysis is Use Case analysis.  
Use of a static code analysis tool can help detect some possible problems.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
Integrated development environments (IDEs) aim to integrate all such help.  
 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.  
Ideally, the programming language best suited for the task at hand will be selected.  
  
 The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Many applications use a mix of several languages in their construction and use.  
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.  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.