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..  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
There exist a lot of different approaches for each of those tasks.  
Ideally, the programming language best suited for the task at hand will be selected.  
One approach popular for requirements analysis is Use Case analysis.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
 Programs were mostly entered using punched cards or paper tape.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
 Code-breaking algorithms have also existed for centuries.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.