A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it..  
 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.  
  
 Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks.  
Use of a static code analysis tool can help detect some possible problems.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
 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.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
 It is very difficult to determine what are the most popular modern programming languages.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.