Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages..  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
 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.  
 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.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
Techniques like Code refactoring can enhance readability.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
 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.  
Use of a static code analysis tool can help detect some possible problems.  
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.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).