Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages..  
Use of a static code analysis tool can help detect some possible problems.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic.  
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.  
Ideally, the programming language best suited for the task at hand will be selected.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
 Code-breaking algorithms have also existed for centuries.  
Integrated development environments (IDEs) aim to integrate all such help.  
There exist a lot of different approaches for each of those tasks.  
Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
 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.  
There are many approaches to the Software development process.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.