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. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. 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. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. 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.
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. One approach popular for requirements analysis is Use Case analysis. Integrated development environments (IDEs) aim to integrate all such help. It is very difficult to determine what are the most popular modern programming languages. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Different programming languages support different styles of programming (called programming paradigms). However, readability is more than just programming style. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Scripting and breakpointing is also part of this process. 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. 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 are many approaches to the Software development process. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.