

Use of a static code analysis tool can help detect some possible problems. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. One approach popular for requirements analysis is Use Case analysis. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. 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. 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. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Computer programmers are those who write computer software. Also, specific user environment and usage history can make it difficult to reproduce the problem. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. Code-breaking algorithms have also existed for centuries. 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. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Programs were mostly entered using punched cards or paper tape. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices.