

This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. One approach popular for requirements analysis is Use Case analysis. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. Many applications use a mix of several languages in their construction and use. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Different programming languages support different styles of programming (called programming paradigms). The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Code-breaking algorithms have also existed for centuries. Use of a static code analysis tool can help detect some possible problems. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Integrated development environments (IDEs) aim to integrate all such help. Many applications use a mix of several languages in their construction and use. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). It affects the aspects of quality above, including portability, usability and most importantly maintainability. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.