

Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. Code-breaking algorithms have also existed for centuries. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Different programming languages support different styles of programming (called programming paradigms). It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). 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. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Unreadable code often leads to bugs, inefficiencies, and duplicated code. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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.