

Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Unreadable code often leads to bugs, inefficiencies, and duplicated code. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Ideally, the programming language best suited for the task at hand will be selected. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. It is very difficult to determine what are the most popular modern programming 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. 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. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Techniques like Code refactoring can enhance readability. 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). The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. In 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. Normally the first step in debugging is to attempt to reproduce the problem. Code-breaking algorithms have also existed for centuries. 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. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).