

They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. There exist a lot of different approaches for each of those tasks. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Ideally, the programming language best suited for the task at hand will be selected. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Also, specific user environment and usage history can make it difficult to reproduce the problem. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. Different programming languages support different styles of programming (called programming paradigms). 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). When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. 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. Programs were mostly entered using punched cards or paper tape. 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. 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.