

Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Ideally, the programming language best suited for the task at hand will be selected. It is very difficult to determine what are the most popular modern programming languages. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Integrated development environments (IDEs) aim to integrate all such help. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Scripting and breakpointing is also part of this 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. 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). 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. There exist a lot of different approaches for each of those tasks. 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. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. 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. 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. 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. Following a consistent programming style often helps readability. One approach popular for requirements analysis is Use Case analysis. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). It affects the aspects of quality above, including portability, usability and most importantly maintainability. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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.