

Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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). 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Programming languages are essential for software development. 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. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. It is very difficult to determine what are the most popular modern programming languages. 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. 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. Ideally, the programming language best suited for the task at hand will be selected.