

Following a consistent programming style often helps readability. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Following a consistent programming style often helps readability. Computer programmers are those who write computer software. One approach popular for requirements analysis is Use Case analysis. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. 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. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Ideally, the programming language best suited for the task at hand will be selected. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Use of a static code analysis tool can help detect some possible problems. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. There exist a lot of different approaches for each of those tasks. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.