

Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. 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. 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. One approach popular for requirements analysis is Use Case analysis. 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. Programmable devices have existed for centuries. Techniques like Code refactoring can enhance readability. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Programs were mostly entered using punched cards or paper tape. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Techniques like Code refactoring can enhance readability. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.