

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. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. One approach popular for requirements analysis is Use Case analysis. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Programming languages are essential for software development. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. There exist a lot of different approaches for each of those tasks. 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. 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. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.