

Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Many applications use a mix of several languages in their construction and use. 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. 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. 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. Use of a static code analysis tool can help detect some possible problems. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. Many applications use a mix of several languages in their construction and use. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Integrated development environments (IDEs) aim to integrate all such help. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Integrated development environments (IDEs) aim to integrate all such help. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Normally the first step in debugging is to attempt to reproduce the problem. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. There are many approaches to the Software development process.