

Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Programs were mostly entered using punched cards or paper tape. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Scripting and breakpointing is also part of this process. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Many applications use a mix of several languages in their construction and use. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). However, readability is more than just programming style. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.