

By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. 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. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Scripting and breakpointing is also part of this process. 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. 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. 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. 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. It is very difficult to determine what are the most popular modern programming languages. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. Integrated development environments (IDEs) aim to integrate all such help. There exist a lot of different approaches for each of those tasks. One approach popular for requirements analysis is Use Case analysis. 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). Techniques like Code refactoring can enhance readability. Programmable devices have existed for centuries. Programming languages are essential for software development. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).