

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, 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Also, specific user environment and usage history can make it difficult to reproduce the problem. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Normally the first step in debugging is to attempt to reproduce the problem. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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).