Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Techniques like Code refactoring can enhance readability. 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). It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Code-breaking algorithms have also existed for centuries. 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). Normally the first step in debugging is to attempt to reproduce the problem. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. Many applications use a mix of several languages in their construction and use. There exist a lot of different approaches for each of those tasks. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Programming languages are essential for software development. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). However, Charles Babbage had already written his first program for the Analytical Engine in 1837.