They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. 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. 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. 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. Integrated development environments (IDEs) aim to integrate all such help. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Also, specific user environment and usage history can make it difficult to reproduce the problem. It is usually easier to code in "high-level" languages than in "low-level" ones. Also, specific user environment and usage history can make it difficult to reproduce the problem. Ideally, the programming language best suited for the task at hand will be selected. 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). By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Normally the first step in debugging is to attempt to reproduce the problem. Programming languages are essential for software development. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Techniques like Code refactoring can enhance readability. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. One approach popular for requirements analysis is Use Case analysis. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.