Following a consistent programming style often helps readability. 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. Normally the first step in debugging is to attempt to reproduce the problem. 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. 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). 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. There exist a lot of different approaches for each of those tasks. It is usually easier to code in "high-level" languages than in "low-level" ones. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Many applications use a mix of several languages in their construction and use. Use of a static code analysis tool can help detect some possible problems. 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. There exist a lot of different approaches for each of those tasks. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. They are the building blocks for all software, from the simplest applications to the most sophisticated ones.