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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. It is usually easier to code in "high-level" languages than in "low-level" ones. 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). This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. Use of a static code analysis tool can help detect some possible 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. 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. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. 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. Allen Downey, in his book How To Think Like A Computer Scientist, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Following a consistent programming style often helps readability. There exist a lot of different approaches for each of those tasks. Use of a static code analysis tool can help detect some possible problems. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. Ideally, the programming language best suited for the task at hand will be selected. 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 a very important task in the software development process since having defects in a program can have significant consequences for its users.