

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). After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Different programming languages support different styles of programming (called programming paradigms). They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Use of a static code analysis tool can help detect some possible problems. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. Normally the first step in debugging is to attempt to reproduce the problem. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. Techniques like Code refactoring can enhance readability.