By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Whatever the approach to development may be, the final program must satisfy some fundamental properties. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Unreadable code often leads to bugs, inefficiencies, and duplicated code. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. 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. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. They are the building blocks for all software, from the simplest applications to the most sophisticated ones.