

There exist a lot of different approaches for each of those tasks. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Computer programmers are those who write computer software. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. It is very difficult to determine what are the most popular modern programming languages. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. There exist a lot of different approaches for each of those tasks. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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). One approach popular for requirements analysis is Use Case analysis. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.