

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. 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. Different programming languages support different styles of programming (called programming paradigms). Use of a static code analysis tool can help detect some possible problems. Integrated development environments (IDEs) aim to integrate all such help. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Following a consistent programming style often helps readability. However, readability is more than just programming style. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. One approach popular for requirements analysis is Use Case analysis. There are many approaches to the Software development process. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. 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, 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. 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. Integrated development environments (IDEs) aim to integrate all such help. 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. One approach popular for requirements analysis is Use Case analysis. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.