

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. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. 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. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Programs were mostly entered using punched cards or paper tape. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Following a consistent programming style often helps readability. 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. Many applications use a mix of several languages in their construction and use. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Integrated development environments (IDEs) aim to integrate all such help. 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. Code-breaking algorithms have also existed for centuries. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. 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.