

Programmable devices have existed for centuries. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Normally the first step in debugging is to attempt to reproduce the problem. 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. 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. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. 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. 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. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Programmable devices have existed for centuries. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code.