Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Integrated development environments (IDEs) aim to integrate all such help. Also, specific user environment and usage history can make it difficult to reproduce the problem. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. Code-breaking algorithms have also existed for centuries. Use of a static code analysis tool can help detect some possible problems. 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. 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. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. 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. Code-breaking algorithms have also existed for centuries. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.