

Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary 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. Normally the first step in debugging is to attempt to reproduce the problem. 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). Ideally, the programming language best suited for the task at hand will be selected. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Unreadable code often leads to bugs, inefficiencies, and duplicated code. Normally the first step in debugging is to attempt to reproduce the problem. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Programmable devices have existed for centuries. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Following a consistent programming style often helps readability. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.