

Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Different programming languages support different styles of programming (called programming paradigms). By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. There exist a lot of different approaches for each of those tasks. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Programs were mostly entered using punched cards or paper tape. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. 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. 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 is usually easier to code in "high-level" languages than in "low-level" ones. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Ideally, the programming language best suited for the task at hand will be selected. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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*. There exist a lot of different approaches for each of those tasks. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.