

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. There exist a lot of different approaches for each of those tasks. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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 Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Many applications use a mix of several languages in their construction and use. Also, specific user environment and usage history can make it difficult to reproduce the problem. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Different programming languages support different styles of programming (called programming paradigms). 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. However, readability is more than just programming style. 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, 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.