

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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Programmable devices have existed for centuries. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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, Charles Babbage had already written his first program for the Analytical Engine in 1837. Programmable devices have existed for centuries. 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. 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. 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). Unreadable code often leads to bugs, inefficiencies, and duplicated code. However, readability is more than just programming style.