

Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. 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. Programming languages are essential for software development. Also, specific user environment and usage history can make it difficult to reproduce the problem.

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. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Techniques like Code refactoring can enhance readability. Integrated development environments (IDEs) aim to integrate all such help. Different programming languages support different styles of programming (called programming paradigms). 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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). Use of a static code analysis tool can help detect some possible problems. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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. 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. Ideally, the programming language best suited for the task at hand will be selected. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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.