

Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Programming languages are essential for software development. 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. 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. Many applications use a mix of several languages in their construction and use. There are many approaches to the Software development process. 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). Many applications use a mix of several languages in their construction and use. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Different programming languages support different styles of programming (called programming paradigms). 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. 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.

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. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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). It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. It is usually easier to code in "high-level" languages than in "low-level" ones. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. There are many approaches to the Software development process.