

High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. 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. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Techniques like Code refactoring can enhance readability. 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. 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. Normally the first step in debugging is to attempt to reproduce the problem. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. Integrated development environments (IDEs) aim to integrate all such help. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Different programming languages support different styles of programming (called programming paradigms). 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Computer programmers are those who write computer software. However, readability is more than just programming style. Techniques like Code refactoring can enhance readability. 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.