

The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Integrated development environments (IDEs) aim to integrate all such help. Use of a static code analysis tool can help detect some possible problems. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. 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. However, readability is more than just programming style. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Programming languages are essential for software development. 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. 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 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. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Scripting and breakpointing is also part of this 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). Different programming languages support different styles of programming (called programming paradigms). 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. 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.