This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
However, while these might be considered part of the programming process, often the term software development is more likely used for this larger overall process – whereas the terms programming, implementation, and coding tend to be focused on the actual writing of code.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
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.  
However, readability is more than just programming style.  
 Following a consistent programming style often helps readability.  
Relatedly, software engineering combines engineering techniques and principles with software development.  
There exist a lot of different approaches for each of those tasks.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
This is interpreted into machine code.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
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).