Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages..  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
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.  
Normally the first step in debugging is to attempt to reproduce the problem.  
 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.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
Scripting and breakpointing is also part of this process.  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
 It is very difficult to determine what are the most popular modern programming languages.  
 Following a consistent programming style often helps readability.  
Ideally, the programming language best suited for the task at hand will be selected.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
  
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.