Also, specific user environment and usage history can make it difficult to reproduce the problem.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Normally the first step in debugging is to attempt to reproduce the problem.  
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.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
This is interpreted into machine code.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
Normally the first step in debugging is to attempt to reproduce the problem.  
 Code-breaking algorithms have also existed for centuries.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.