Ideally, the programming language best suited for the task at hand will be selected..  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
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.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
Use of a static code analysis tool can help detect some possible problems.  
However, readability is more than just programming style.  
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.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
Many applications use a mix of several languages in their construction and use.  
For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.  
 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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Techniques like Code refactoring can enhance readability.