Integrated development environments (IDEs) aim to integrate all such help.  
Normally the first step in debugging is to attempt to reproduce the problem.  
Use of a static code analysis tool can help detect some possible problems.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
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.  
To produce machine code, the source code must either be compiled or transpiled.  
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.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
Ideally, the programming language best suited for the task at hand will be selected.  
There exist a lot of different approaches for each of those tasks.  
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.  
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.  
Ideally, the programming language best suited for the task at hand will be selected.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.