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..  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
 Following a consistent programming style often helps readability.  
Programming languages are essential for 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.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
Scripting and breakpointing is also part of this process.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.