Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment..  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
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.  
 Programmable devices have existed for centuries.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 Code-breaking algorithms have also existed for centuries.  
 Computer programmers are those who write computer software.  
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.  
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.  
Ideally, the programming language best suited for the task at hand will be selected.  
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.  
 Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line.  
 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.