Ideally, the programming language best suited for the task at hand will be selected..  
Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic.  
There exist a lot of different approaches for each of those tasks.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.  
Programming languages are essential for software development.  
Their jobs usually involve:  
 Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers.  
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.  
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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
 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.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.