Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
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.  
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.  
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.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
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.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
 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.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.