It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Use of a static code analysis tool can help detect some possible problems. Programs were mostly entered using punched cards or paper tape. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. However, readability is more than just programming style. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. One approach popular for requirements analysis is Use Case analysis. 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. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. It is very difficult to determine what are the most popular modern programming languages. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Ideally, the programming language best suited for the task at hand will be selected. Different programming languages support different styles of programming (called programming paradigms). Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. It is usually easier to code in "high-level" languages than in "low-level" ones. Techniques like Code refactoring can enhance readability. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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.