**prevent, flow, dumps,**

**Debugging** is the process of finding and resolving defects or problems within a computer program that …prevents…………….. correct operation of [computer software](https://en.wikipedia.org/wiki/Computer_software) or a [system](https://en.wikipedia.org/wiki/System).

Debugging tactics can involve [interactive](https://en.wikipedia.org/wiki/Interactive) debugging, [control …flow………………….](https://en.wikipedia.org/wiki/Control_flow) analysis, [unit testing](https://en.wikipedia.org/wiki/Unit_testing), [integration testing](https://en.wikipedia.org/wiki/Integration_testing), [log file analysis](https://en.wikipedia.org/wiki/Logfile), monitoring at the [application](https://en.wikipedia.org/wiki/Application_monitoring) or [system](https://en.wikipedia.org/wiki/System_monitoring) level, [memory …dumps………………](https://en.wikipedia.org/wiki/Memory_dump), and [profiling](https://en.wikipedia.org/wiki/Profiling_(computer_programming))

**tiresome, extent, breakpoints,**

Debugging ranges in complexity from fixing simple errors to performing lengthy and …tiresome……………… tasks of data collection, analysis, and scheduling updates. The debugging skill of the programmer can be a major factor in the ability to debug a problem, but the difficulty of software debugging varies greatly with the complexity of the system, and also depends, to some …extent……………….., on the [programming language](https://en.wikipedia.org/wiki/Programming_language)(s) used and the available tools, such as [*debuggers*](https://en.wikipedia.org/wiki/Debugger). Debuggers are software tools which enable the [programmer](https://en.wikipedia.org/wiki/Programmer) to monitor the [execution](https://en.wikipedia.org/wiki/Execution_(computers)) of a program, stop it, restart it, set…breakpoints………………., and change values in memory. The term *debugger* can also refer to the person who is doing the debugging.

**exception, assembly, corruption,**

Generally, [high-level programming languages](https://en.wikipedia.org/wiki/High-level_programming_language), such as [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), make debugging easier, because they have features such as […exception…………………… handling](https://en.wikipedia.org/wiki/Exception_handling) and [type checking](https://en.wikipedia.org/wiki/Type_checking) that make real sources of erratic behaviour easier to spot. In programming languages such as [C](https://en.wikipedia.org/wiki/C_(programming_language)) or [assembly](https://en.wikipedia.org/wiki/Assembly_language), bugs may cause silent problems such as [memory …corruption………………..](https://en.wikipedia.org/wiki/Memory_corruption), and it is often difficult to see where the initial problem happened. In those cases, [memory debugger](https://en.wikipedia.org/wiki/Memory_debugging) tools may be needed.

**rare, detect, variable, dubious ,**

In certain situations, general purpose software tools that are language specific in nature can be very useful. These take the form of [*static code analysis tools*](https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis). These tools look for a very specific set of known problems, some common and some …rare…………, within the source code. concentrating more on the semantics (e.g. data flow) rather than the syntax, as compilers and interpreters do.

Some tools claim to be able to …detect…………….over 300 different problems. Both commercial and free tools exist for various languages. These tools can be extremely useful when checking very large source trees, where it is impractical to do code walkthroughs. A typical example of a problem detected would be a …variable……………………..dereference that occurs *before* the variable is assigned a value. As another example, some such tools perform strong type checking when the language does not require it. Thus, they are better at locating likely errors in code that is syntactically correct. But these tools have a reputation of false positives, where correct code is flagged as …dubious………………. The old Unix [*lint*](https://en.wikipedia.org/wiki/Lint_programming_tool) program is an early example.

For debugging electronic hardware (e.g., [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware)) as well as low-level software (e.g., [BIOSes](https://en.wikipedia.org/wiki/BIOS), [device drivers](https://en.wikipedia.org/wiki/Device_driver)) and [firmware](https://en.wikipedia.org/wiki/Firmware), instruments such as [oscilloscopes](https://en.wikipedia.org/wiki/Oscilloscope), [logic analyzers](https://en.wikipedia.org/wiki/Logic_analyzer) or [in-circuit emulators](https://en.wikipedia.org/wiki/In-circuit_emulator) (ICEs) are often used, alone or in combination. An ICE may perform many of the typical software debugger's tasks on low-level [software](https://en.wikipedia.org/wiki/Software) and [firmware](https://en.wikipedia.org/wiki/Firmware).

**reproduce, parallel, simplified, track, crash, conquer, input,**

Normally the first step in debugging is to attempt to …reproduce…………………….the problem. This can be a non-trivial task, for example as with […parallel………………….processes](https://en.wikipedia.org/wiki/Parallel_computing) or some [unusual software bugs](https://en.wikipedia.org/wiki/Unusual_software_bugs). Also, specific user environment and usage history can make it difficult to reproduce the problem.

After the bug is reproduced, the …input…………………of the program may need to be simplified to make it easier to debug. For example, a bug in a compiler can make it [crash](https://en.wikipedia.org/wiki/Crash_(computing)) when parsing some large source file. However, after simplification of the test case, only few lines from the original source file can be sufficient to reproduce the same crash. Such simplification can be made manually, using a [divide-and-…conquer…………………](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm) approach. The programmer will try to remove some parts of original test case and check if the problem still exists. When debugging the problem in a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface), 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.

After the test case is sufficiently…simplified………………, a programmer can use a debugger tool to examine program states (values of variables, plus the [call stack](https://en.wikipedia.org/wiki/Call_stack)) and …track…………….. down the origin of the problem(s). Alternatively, [tracing](https://en.wikipedia.org/wiki/Tracing_(software)) can be used. In simple cases, tracing is just a few print statements, which output the values of variables at certain points of program execution.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]