IT101

Debugging

What is Debugging?

- Debugging is the process of <u>finding and resolving of defects</u> that prevent correct operation of computer software or a system. Debugging tends to be harder when various subsystems are tightly coupled, as changes in one may cause bugs to emerge in another. (Wikipedia definition)

- Typically debugging is done with the aid of special purpose software. The software can control (stop, resume) the program flow, watch variables, log files or memory dumps.

The terms "bug" and "debugging" are popularly attributed to Admiral Grace Hopper in the 1940s.[1] While she was working on a Mark II Computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system. However the term "bug" in the meaning of technical error dates back at least to 1878 and Thomas Edison, and "debugging" seems to have been used as a term in aeronautics before entering the world of computers. (Wikipedia)
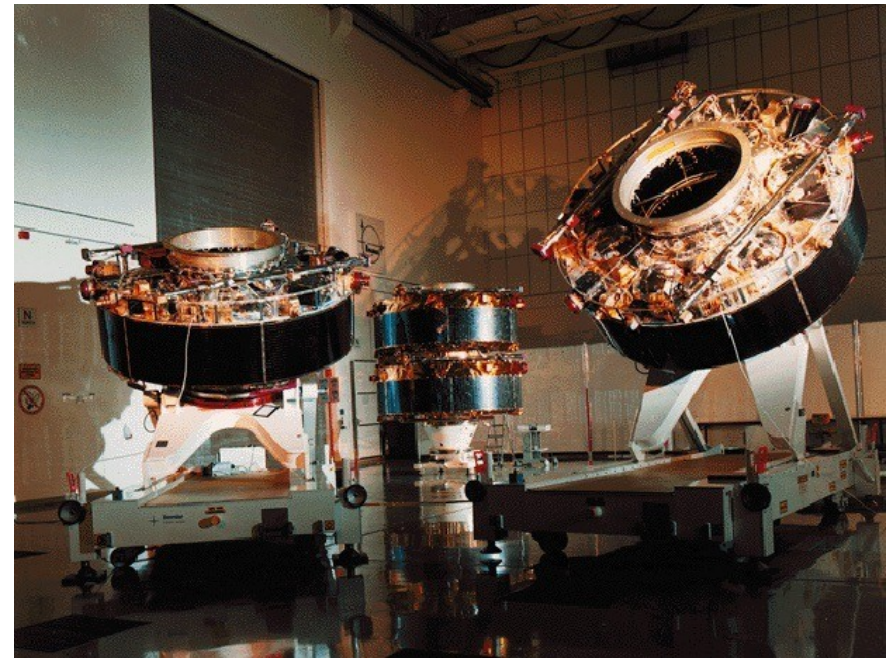
Why Debug?

- The debugging skills of a software developer are a major factor when it comes to eliminating defects.

- Software tends to become more and more complex, which makes debugging harder.

- Bugs may occur through memory corruption, which is difficult to isolate and track. The error is not necessarily the origin of the problem.

Software bug in onboard guidance software caused the destruction of four satellites (known collectively as Cluster) and the Ariane 5 rocket on which they were riding. Shortly after launch, the guidance software tried to convert the horizontal velocity (which was greater than anticipated) from a 64-bit floating point number to a 16-bit signed integer, which caused an overflow error. The guidance system (and its backup, which had the same bug) then shut down, causing the rocket to veer off course and, ultimately, self destruct 30 seconds after launch on June 4, 1996.

http://www.itworld.com/article/2823083/enterprise-software/88716-8-famous-software-bugs-in-space.html

Debugging Software

- Interactive Debugging Software monitors the flow of the program and it can modify this flow, inspect and change variable values, pause execution at predefined locations and influence the outcome of computations.

- Static Debugging Software checks the code to identify potential problems. Static code analysis tools come handy to identify such problems as memory leaks, redundant code and infinite recursive structures.

- The first step in debugging is to reproduce the problem, which is non-trivial in case of memory leaks, distributed systems and multithreaded programs.

- Once the problem is repoduced, the input is simplified to isolate the problem to the minimum factors required for debugging.

- Then the developer can use an interactive debugger, such as the one that comes with NetBeans, to examine local variables, program control flow and method outputs.

Exercise

- Run the Tennis Game version distributed for this exercise.

- Notice the bug in score counter.

- Add a breakpoint inside the TennisGame.changeScore method.

- Run the Tennis Game in Debug mode (Debug → Debug Project).

- The program execution will stop once it reaches the changeScore method.

- Open Window → Debugging → Variables and examine the variables' values.

  - What is the value of   the "score" and "increment" variables?

  - Use "step into" or "step over" buttons to move one step further. What is the value of the "score" and "increment" variables now?

  - Change the "increment" variable's value and continue the program execution.

  - Where from is the "changeScore" method called?

- Fix the program and test again.

Important Debugging Buttons