What kinds of bugs are there?

(1) program produces incorrect results
    a. incorrect logic

(2) program crashes
    b. unhandled exception caused by bug or external conditions

(3) program never terminates

(4) program is to slow

(5) program uses too many resources
    a. memory, disk space, network bandwidth, cpu, open files, db connections

How many bugs are there?

A typical bug density in industrial code is between 1 and 25 bugs per 1000 LOC

The Applications Division at Microsoft reports between 10 and 20 bugs per 1000 LOC, and about 0.5 bugs per 1000 LOC in released software

How are bugs distributed in the code?

It's natural to assume that bugs are evenly distributed throughout a program, but it's not true.

Research has shown that:
80% of a system's errors are found in only 20% of its classes or routines
50% of a system's errors are found in only 5% of its classes or routines

Why would that be?

Becoming an effective debugger

Some people seem to find bugs with relative ease, while others struggle mightily. What makes one person so much better at debugging than another?

Debugging involves diagnosing problems in a system, must like a medical doctor or car mechanic. In order to effectively diagnose problems in a system, one must have an accurate and detailed mental model of how the system works (or should work). Based on such a mental model, one can effectively relate symptoms back to their root causes.

Customer: “My car makes a strange pinging sound when I brake while in reverse.”
Car Mechanic: “That sound usually means your left-rear gonculator gasket is leaking and needs to be replaced.”

In programming, what constitutes our “mental model”?
    a. Understanding of the program’s design
    classes, interfaces, object interactions, control flow

    b. Understanding of programming language semantics

binary data representations, runtime stack, heap, language constructs, pointers, OS and hardware interactions, etc.

Experience also plays a role in effective debugging. A bug is easier to find if you've seen a similar one before.

Avoiding Bugs

The best way to debug your program is to avoid introducing bugs in the first place.

(1) pay attention to compiler warnings

(2) unit testing

(3) assertions

(4) parameter checking

Debugging Process

(1) Find a reproducible test case that causes the program to fail (often it finds you)

(2) Reduce the size of the input data as much as possible while still preserving the failure (the simpler the test case, the better)

a. reduces the amount of code that is executed, thus narrowing the possibilities

b. reduces the amount of time it takes to reproduce the error

c. reduces the volume of debugging output

(3) Determine where in the program the bug manifests itself (not where the bug occurred, but where it shows up)

a. the code that produces erroneous output

b. the place where the program crashed or threw an exception

(4) Locate the code that caused the bug

a. the part of the program that caused the erroneous state may be far removed and seemingly unrelated to the part of the program where the bug manifests itself.

b. any code that executes before the point of failure is suspect

c. inspect the state of the program at the failure point to understand in what ways the program's state is incorrect, and work back from there

d. minimizing the distance, in time and space, between these two points will help reduce debugging time
    i. unit testing
    ii. parameter checking
    iii. assertions

Debugging Techniques

a. Code Reading

b. Trace Debugging
    print statements that indicate:
        (1) where the program is executing
        (2) inspect variable values

    helps to determine location of crash and/or source of incorrect or corrupted data

    toString method on classes (prints contents of object)

    take trace statements out after bug is found, comment them out, or disable with boolean flag

c. Logging
    send output to destinations other than the screen
        (files, network logging servers)

    different logs for different parts of the program
        (classes, threads, etc.)

    different severity levels (DEBUG, INFO, WARNING, ERROR, FATAL, ...)

    turn logging on and off depending on what part of the program you're in and severity level

d. Progressive Code Elimination
    Blue Wolf

    home in on buggy code by commenting out calls or using stubs to eliminate code

Interactive Debuggers

Eclipse Debugging Tutorial