CSC 245: Objects an Data Abstraction

Chapter 1
Phases of Software Development

Outline

- Specifications, Design, Implementation
  - Pre- & Post-conditions.
- Running Time Analysis
  - Big-O Notation.
  - Worst-Case, Average-Case, & Best-Case
- Testing & Debugging
  - Choosing Test Data
  - Boundary Values

Software Development Phases

- Specification of the task.
- Design of a solution.
- Implementation (coding) of the solution.
- Analysis of the solution.
- Testing & debugging.
- Maintenance & evolution of the system.
- Obsolescence.

Definitions

- Specification
  - Precise description of a problem.
- Design
  - Formulation of steps to solve a problem.
- Implementation
  - Actual code (e.g., Java, C, C++,…).
- Algorithm
  - Set of instructions in solution.
  - Specified in Java, C,…, or pseudo-code.
- Pseudo-code
  - Mixture of formal English & programming language.

Design Technique

- Stepwise Refinement
    - Problem Decomposition
    - Divide and Conquer
  - Divide task into a few subtasks.
  - Decompose each subtask into smaller subtasks.
- Criteria
  - Short descriptions.
  - Uncoupled components.
    - Maximize information hiding.
  - Code reuse.

Precondition & Postcondition

- Precondition
  - What must be true when method is called.
  - Needed to guarantee correct behavior.
- Postcondition
  - What will be true after method call has completed.
  - Valid precondition & correct method implementation guarantee postcondition.

Running Time Analysis

- Definition
  - Estimate of algorithm’s execution time.
  - Reasoning about an algorithm’s speed.
- Estimate the number of operations.
  - Decide what operations count.
    - Multiplication vs. addition.
    - Method call vs. arithmetic operation.
  - Estimate as function of problem size.
- Use Big-O notation.

Stair-Counting Problem: Eiffel Tower

- Walk down & keep a tally.
  - Make a mark for each step on way down.
  - Walk back up.
- Walk down, let Judy keep the tally.
  - Walk down one step.
    - Leave marker on steps.
  - Go back to start & add marker on page.
  - Go back to marker.
- Jervis to the rescue.
  - Read sign: 2689 steps!

Stair-Counting Problem…

- Operations
  - Walking up or down a step.
  - Marking a symbol on the paper.
- Walk down & keep a tally
  - 2689 steps down, 2689 steps up, 2689 marks on paper.
  - Total = 8067 operations.
- Walk down, but let Judy keep tally
  - Downward or upward steps: (1 + 2 + … + 2689) = 3,616,705.
  - 2689 marks.
  - Total = 7,236,099 operations!
- Jervis to the rescue
  - 4 operations (1 per digit).

Stair-Counting Problem…

- Generalization
  - Assume n steps.
- Technique 1: Walk down & count
  - 3n operations.
- Technique 2: Judy counts
  - n + 2 [n (n + 1) / 2]
  - $n^2 + 2n$
- Technique 3: Read sign
  - Number of digits in n.
  - $\lfloor \log_{10} n \rfloor + 1$

Stair-Counting Problem…

- Big-O Notation
  - Order of magnitude estimate of operation count.
- Technique 1: Walk down & count
  - O(n)
  - Linear time.
- Technique 2: Judy counts
  - $O(n^2)$
  - Quadratic time.
- Technique 3: Read sign
  - O(log n)
  - Logarithmic time.

Stair-Counting Problem…

Number of stairs (*n*)	Logarithmic $O(\log n)$ **Technique 3, with $\lfloor \log_{10} n \rfloor + 1$ operations**	Linear $O(n)$ **Technique 1, with $3n$ operations**	Quadratic $O(n^2)$ **Technique 2, with $n^2 + n$ operations**
10	**2**	**30**	**120**
100	**3**	**300**	**10,200**
1000	**4**	**3000**	**1,002,000**
10000	**5**	**30,000**	**100,020,000**

Search Example

- Specification
  - public static boolean search(double[] data, double target)
  - search an array for a specified number.
- Parameters
  - data—an array of double numbers.
  - target—a particular number that we are searching for.
- Returns
  - true—indicates that target occurs in the array.
  - false—indicates that target does not occur in the array.

Search Example…

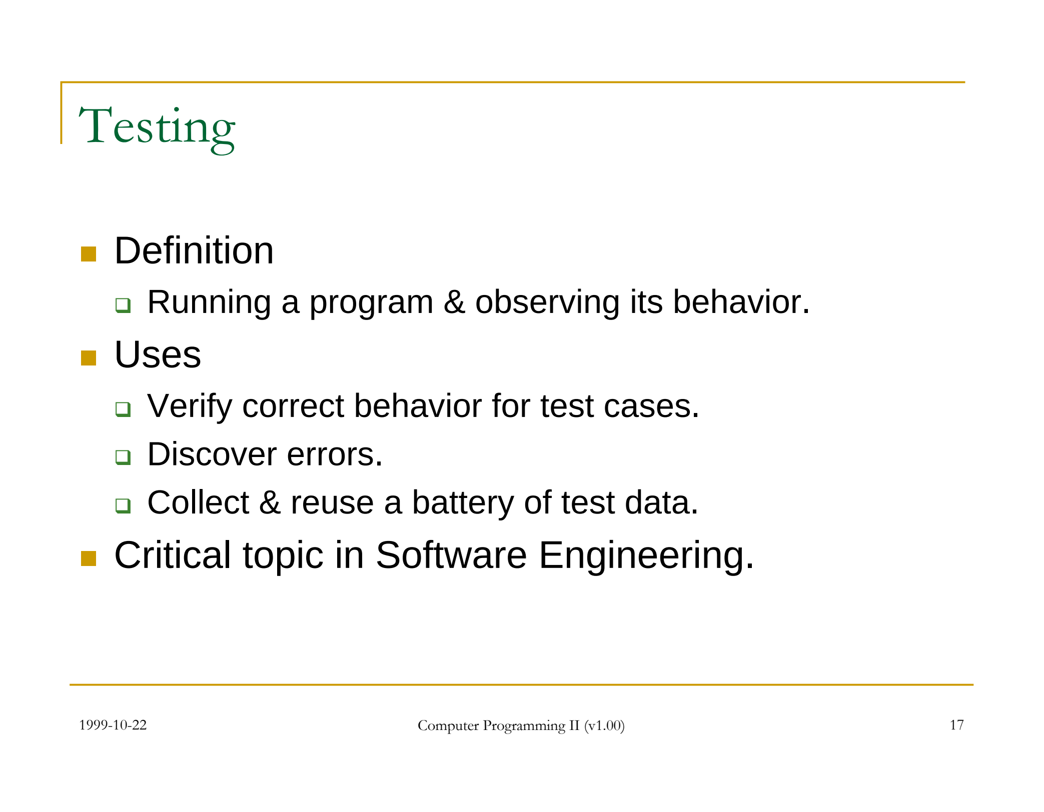
```
public static boolean search(double[] data, double target)
{
  int i;
  for (i = 0; i < data.length; i++)
  {
    // check whether the target is at data[i]
    if (data[i] == target)
      return true;
  }
  // Loop finished without finding the target.
  return false;
}
```

Search Example: Time Analysis

- Number does not occur in array!
- Loop start
  - Initialize loop variable (i = 0).
  - Evaluate loop condition (i < data.length).
- Loop body
  - n iterations.
  - k operations per iteration (3 or 4).
- Loop finishes
  - 1 operation (return).
- Total
  - k n + 3 = O(n)

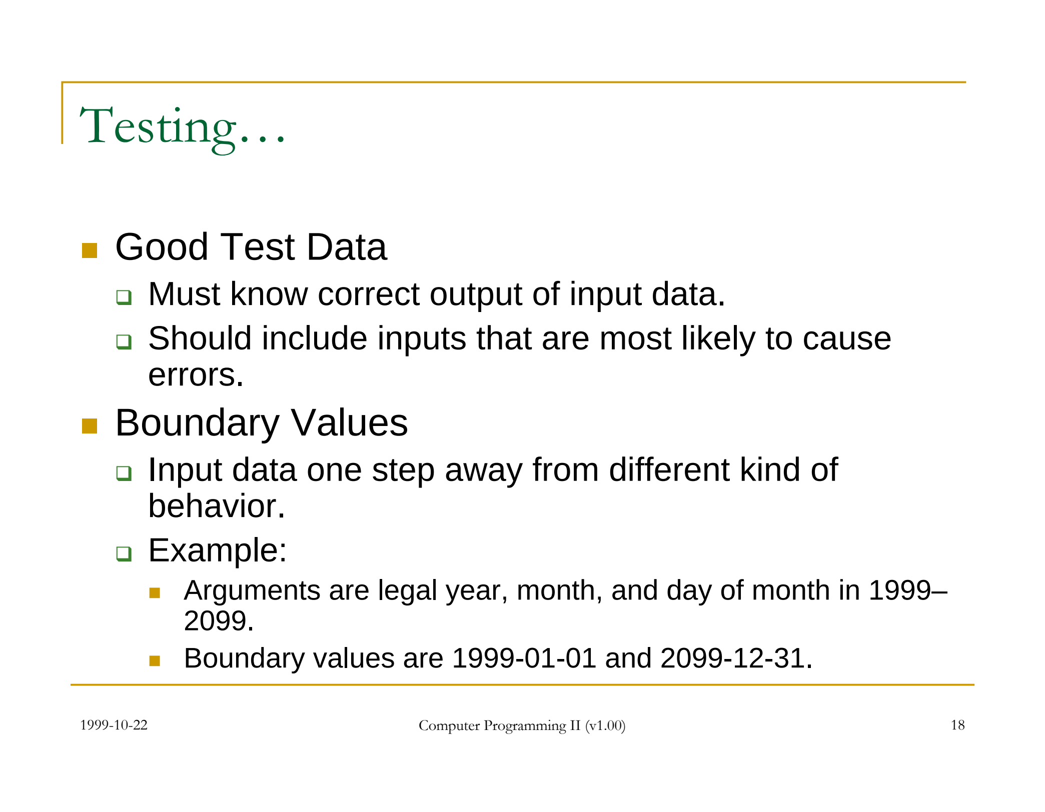
Running Time Analysis…

- Worst-case
  - Maximum number of operations.
- Average-case
  - Average number of operations.
- Best-case
  - Smallest number of operations.



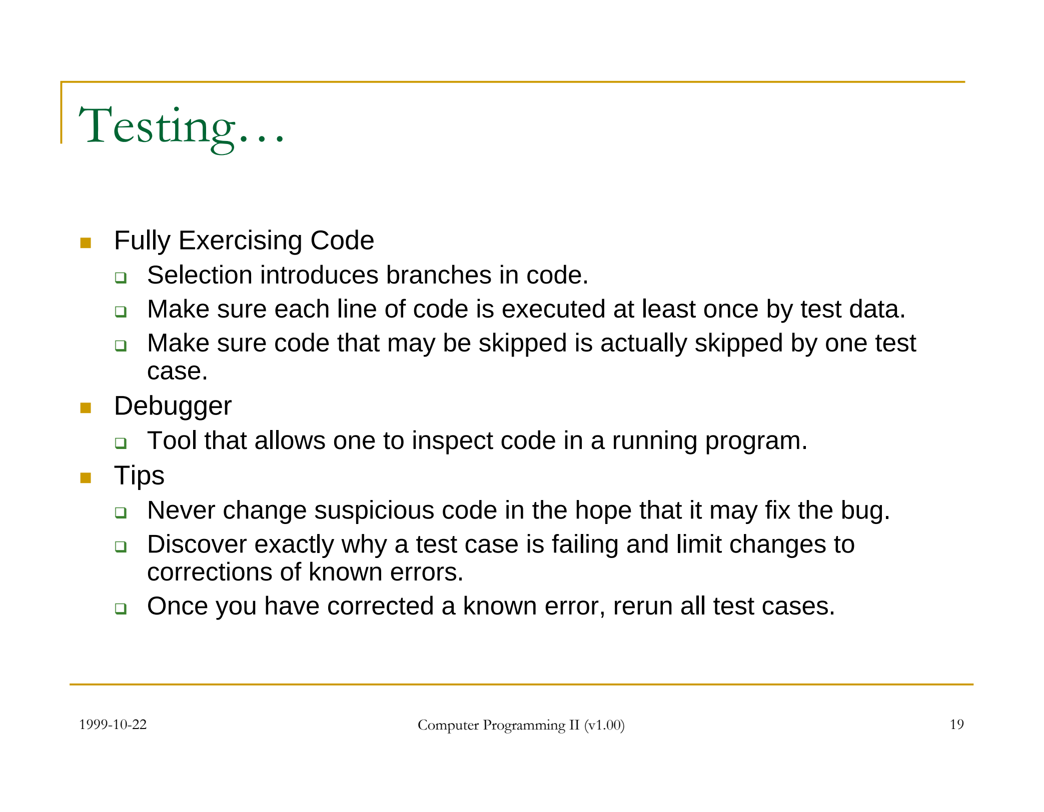
Testing

- Definition
  - Running a program & observing its behavior.
- Uses
  - Verify correct behavior for test cases.
  - Discover errors.
  - Collect & reuse a battery of test data.
- Critical topic in Software Engineering.



Testing…

- Good Test Data
  - Must know correct output of input data.
  - Should include inputs that are most likely to cause errors.
- Boundary Values
  - Input data one step away from different kind of behavior.
  - Example:
    - Arguments are legal year, month, and day of month in 1999–2099.
    - Boundary values are 1999-01-01 and 2099-12-31.



Testing…

- Fully Exercising Code
  - Selection introduces branches in code.
  - Make sure each line of code is executed at least once by test data.
  - Make sure code that may be skipped is actually skipped by one test case.
- Debugger
  - Tool that allows one to inspect code in a running program.
- Tips
  - Never change suspicious code in the hope that it may fix the bug.
  - Discover exactly why a test case is failing and limit changes to corrections of known errors.
  - Once you have corrected a known error, rerun all test cases.