How to Develop Small Programming Projects*

Jill Seaman
CS 1428
Fall 2014

*without banging your head against the wall

Getting Started

- Start early: we always underestimate the complexity of the problem.

- Understand the material: study first!

- Understand the requirements (READ the directions, don't make assumptions).

- Use top-down design to break up the problem into pieces, write pseudo-code.

- Make a plan before you implement.

Develop Programs Progressively
(incremental development)

- Do not attempt to implement an entire program all at once.

- Implement a very small, but workable, part.

- Compile, fix syntax errors, execute, debug

- Add another small part, refine the code

- Compile + test. Any new errors are (probably) due to newly added code.

- Repeat until complete

Compiler (syntax) Errors

- Fix only the first one or two before re-compiling, later errors may be dependent.

- Don't speak compiler?
  Google the error text (with caution)

- Think of common syntax errors

  - Missing semicolons

  - Misspelled variable names

  - Misplaced ( ) or { }, backwards << or >>

Testing

- Testing: running (part of) the program with simulated data, checking the actual results against expected result

- Run tests for boundary conditions:
  - Empty arrays, full arrays, last element
  - Smallest and largest valid values
  - Values used in if/while conditions

Runtime Errors (bugs)

- Program executes but output is wrong, running a test gives unexpected result

- Debugging: figure out why it failed

- Add output statements in strategic places
  - check values of variables (label them!)
  - trace execution path, see which statements are being reached.