How to Develop Small Programming Projects*

Jill Seaman
CS 3358
Summer I 2012

*without banging your head against the wall

Getting Started

- Start early: we always underestimate the complexity of the problem.
- Understand the requirements (READ the directions, don't make assumptions).
- Understand the material: study first!
- Use some top-down design to break up the problem into pieces.
- Make a plan before you implement.

Develop Programs Progressively
(incremental development)

- Do not attempt to implement and test an entire program all at once.
- Implement a very small, but workable, part.
- Compile, fix syntax errors, execute, debug
- Add another small part
- Compile + test. Any new errors are (probably) due to newly added code.

Develop Programs Progressively

- Add testcases as you go, keep running them all to make sure nothing was broken.
- Always have code that compiles and runs correctly.
- Makes it easy to break up the programming effort over multiple sittings.
- If you can't complete the whole project, you will get "partial credit".

Re-use cautiously:

- Sometimes it helps to start from an existing solution:
- Duplicate, modify.
- Keep this on a small scale.

Always write good code

- Use good variable and function names from the start.
- Maintain good indentation from the start.
- Add in-code comments as you go.
  - can add variable and function comments later
- Code is always neat, readable, won't have to make it pretty later.

Testing

- Have test cases for boundary conditions:
  - Empty arrays, full arrays, last element
  - Smallest and largest valid values
  - Values used in if/while conditions
  - Negative numbers
- Have test cases for every line of code.

Compiler 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 >>

Runtime Errors

- Program executes but output is wrong, Testcase gives unexpected result
- Could learn to use a debugger (gdb?)
- Add output statements in strategic places
  - check values of variables (Label!)
  - trace execution path

Runtime Errors

- Don't forget to remove couts when the error is discovered!
- Think of common programming errors
  - one-off array indexes
  - redeclare a variable inside a loop
  - using = instead of ==
  - forgetting to update a var in a loop