Slide 1

Administrivia

- Reminder: Homework 2 due today.
- Homework 3 on the Web; due next Wednesday.

Slide 2

Minute Essay From Last Lecture

- *Almost* everyone had worked with conditional execution before. The concepts are indeed similar, but syntax may be different (though for Java it's pretty much identical — no accident).
- Not from the minute essays, but from another e-mail:

  "Why when I try to input a large number do I get weird results (wrong and/or negative)?"

  Consequence of integers being represented as fixed size. Not much to be done about it at this point. Something for future discussion?

Slide 3

Example of Conditional Execution, Continued

- Recall problem from last time: Compute and print the roots of a quadratic equation

$$ax^2 + bx + c = 0$$

  using the formula

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- As noted, a key issue is identifying all the cases that need to be dealt with differently . . .

Slide 4

Choosing Good “Test Data”

- After you’ve written a program, you need to try it with various input (“test it”).
- Choosing good tests is maybe a bit of an art, but you should try to choose ones that:
  - Demonstrate that all parts of your code work (i.e., that you explore all the “paths” through it).
  - Allow you to easily know whether the answer is right! i.e., choose inputs where you can easily figure out what the answer should be.

Slide 5

Quotes of the Day/Week/?

- From a key figure in the early days of computing:

  “As soon as we started programming, we found to our surprise that it wasn’t as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs.” (Maurice Wilkes: 1948)

- From someone in a discussion group for the Java programming language:

  “Compilers aren’t friendly to anybody. They are heartless nitpickers that enjoy telling you about all your mistakes. The best one can do is to satisfy their pedantry to keep them quiet :)”

Slide 6

Functions and Problem Decomposition

- So far all our programs have been one big chunk of code. This is okay for simple programs, but quickly becomes difficult to understand as problems get bigger.
- Further, some things we don’t want to, or can’t, really write ourselves, such as the code for input/output.
- So C, like many/most other programming languages, gives you a way of decomposing problems into subproblems. C calls them *functions*. Using this feature to good effect is something of an art, but may teach you something about problem decomposition in general, which is a useful skill.

Functions in C

- C functions are similar to functions in math, except that they can have side effects (similar to how evaluation of expressions can have side effects).
- Next time we'll look at details, but for now, notice that every program you / we have written so far defines a function called `main`, and most of them use system library functions `scanf` and `printf`.

Minute Essay

- Anything noteworthy (interesting, difficult, etc.) about Homework 2?