Slide 1

Administrivia

- Reminder: Homework 2 due today. Please remember in the subject line to identify both the course and the assignment!

Slide 2

Minute Essay From Last Lecture

- Most people had seen some form of conditional execution in other environments (mostly Matlab). I think that helps!
- Most people had *some* exposure to binary numbers, but most said at least part of the material from last week was new. So, not a waste of class time — I'm glad!

Slide 3

Condiional Execution — Recap/Review

- C, like most if not all programming environments, supports conditional execution. Syntax is

  `if` (*boolean-expression*)

  statement1

  `else`

  statement2

  where *statement1* and *statement2* can be single statements or blocks enclosed in curly braces.

- `else` part can be omitted if not needed. Can "chain" testing several conditions with `else if` (as in example from last time).
- Note that *boolean-expression* can be something involving a side effect, such as the example last time of checking the value returned by `scanf`.

Slide 4

Conditional Execution, Continued

- Chains of `else if` are seful, but sometimes there's a shorter way: If all of the conditions are of the form

  integer_expression == *value*

  then we can use the `switch` construct. Notice that characters (`char`) count as integers in this context.

- Example (similar to calculator example in textbook) on next slide.

Slide 5

Conditional Execution, Continued

-
  ```
  char menu_pick;  /* should be one of '+', '-' */
  /* .... */
  switch (menu_pick) {
      case '+':
          result = input1 + input2;
          break;
      case '-':
          result = input1 + input2;
          break;
      default:
          result = 0;
          printf("operator not recognized\n");
  }
  ```

Slide 6

Conditional Expressions

- C also provides a short way to express things of the form

  `if` (*condition*)

  variable = *value1*

  `else`

  variable = *value2*

  namely the ternary (three operands) operator `?`.

- Example:

  ```
      sign = (x >= 0) ? 1 : -1;
  ```

  assigns 1 to `sign` if `x` is non-negative, -1 otherwise.

- (Use with caution — compact, but can easily lead to code that's difficult for humans to understand.)

Slide 7

Conditional Execution, Continued

- A simple use for conditional execution is checking for input that doesn't make sense. But of course there are many others!
- Challenging part in many applications is to make sure you've covered all the possibilities.

Slide 8

Example — Finding Roots of a Quadratic Equation

- As a rather math-y example, let's write a program to compute and print the roots of a quadratic equation

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

- We'll use the formula

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

and try to account for as many cases as we can ...

Slide 9

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 10

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 :)"

Minute Essay

- What about Homework 2 did you found interesting, difficult, or otherwise noteworthy?