Slide 1

Administrivia

- I've started grading Homework 1; you will get grades by e-mail. (Soon, I hope, but no guarantees.)
- Homework 2 is on the Web; due a week from today.

Slide 2

More Administrivia

- A few words about minute essays and homeworks and e-mail:

  I sometimes don't look at minute essays until the next class, or at homeworks until ready to grade.

  So if you have an urgent question, put "urgent" or "question" in the subject line please!

- A few words about office hours:

  I recently posted on my Web page official office hours; they're basically the breaks between my W and T/R classes. For now you're more likely to find me somewhere in Halsell than in the CSI. If you can't make these times and need to talk to me, please try e-mail — it's not the best medium in general, but for me this semester it may work best.

Slide 3

Minute Essay From Last Lecture

- (Review the question, answer. Notice that for minute essay questions where there's an answer, it will be in the final version of the slides.)
- Many people got it, but there were some who thought the problem was with printing expressions, which — it's not. "FYI", maybe.

Slide 4

Conditional Execution

- Also as in other procedural languages, C has syntax for saying that some code should be executed only if some condition holds.
- Syntax is

  `if (` *boolean-expression* `)`
  statement1
  `else`
  statement2

  where *statement1* and *statement2* can be single statements or blocks enclosed in curly braces.
- You can build up chains of conditions by making the statement after `else` another `if`, and you can omit the `else` and following statement. (The ideas here should be very familiar, and for most of you even the syntax should be pretty much what you know.)

Slide 5

Conditional Expressions

- Scala and Python both provide a way to include if/else idea within an expression.
- C does too, but it's not as obvious — "ternary operator", e.g.,

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

Slide 6

Conditional Execution — One More Thing

- One other conditional-execution construct you may encounter — `switch`. Basically a short form of if/elseif/else. Somewhat like `match` in Scala but nowhere near as powerful. Example:

```
char c; /* code to set value omitted */
switch (c) {
    case 'a': printf("first case\n"); break;
    case 'b': printf("second case\n"); break;
    default:  printf("default\n");
}
```

Slide 7

Simple Input, Revisited

- Recall that for now at least we're using `scanf` to read from standard input, and it returns a value that indicates success failure.
- So now that we officially know about conditional execution we can check for input errors. C-idiomatic way to check for success is, e.g.,

```
if (scanf("%d %d", &var1, &var2) == 2) ....
```

Slide 8

Functions in C

- Functions in C are conceptually much like functions in other procedural programming languages. (Methods in object-oriented languages are similar but have some extra capabilities.)

  I.e., a function has a *name*, *parameters*, a *return type*, and a *body* (some code).
- One difference between C and higher-level languages: You aren't supposed to use a function before you tell the compiler about it, either by giving its full *definition* or by giving a *declaration* that specifies its name, parameters, and return type. The function body can be later in the same file or in some other file.
- Also, C functions are not supposed to be nested (though some compilers allow it.)

Slide 9

Parameter Passing in C

- In C, all function parameters are passed "by value" — which means that the value provided by the caller is copied to a local storage area in the called function. The called function can change its copy, but changes aren't passed back to the caller.
- An apparent exception is arrays — more later when we talk about them.

Slide 10

Functions, Local Variables, and Recursion

- Functions in C can contain local variables. Every time you call the function, you get a fresh copy of the variables.
- So yes, recursive functions work the way you (probably?) think they should.

Slide 11

Library Functions in C

- C does include a library of standard functions, though it's nowhere near as extensive as that of some languages.
- At least on UNIX-like systems, for each library function there should be a `man` page that tells you about it, including information about `#include` files you need and link-time options (e.g., `-lm` for `sqrt`). For now, be advised that asterisks in types denote pointers, which we will talk about soon.

Slide 12

Functions in C — Example(s)

- Examples as time permits.

Slide 13

Minute Essay

- What was interesting or difficult or otherwise noteworthy about the programming problem in Homework 1?