Slide 1

Administrivia

- Reminder: Homework 1 due today (end of the day).
- Homework 2 will be on the Web later today / early tomorrow. I will send mail.

Slide 2

Preprocessor Directives — A Bit More

- Examples so far have started with `#include` directive to tell compiler where to find information about I/O library functions. (Roughly — "include", i.e., copy, information from header file-or-equivalent.) This is input to the "preprocessor".
- Another useful directive is `#define`, to give meaningful names to constants, e.g.,

```
#define IMPRECISE_PI 3.14159
```

Slide 3

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 (and should probably be indented, for the convenience of human readers).
- 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; only the syntax should be new.)

Slide 4

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 5

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 6

Sidebar — Man Pages, Revisited

- As mentioned earlier, most commands — and many library functions — have "man pages" (short for "manual"). These are meant as online references rather than tutorials, so not always easy reading, but usually very complete.
- `man` program shows its output to you using a program intended for paging through text. On our systems, default is `less`. Keystroke commands include space to go forward, `b` to go back, `q` to quit. `h` for help — or, of course, you could read all about it (how?).
- Sometimes there are multiple commands/functions with the same name. `printf` is one. `man printf` tells you about the (command-line) command, not the C library function. To get all possibilities, `man -a printf`. To get the one for the library function, `man 3 printf`.

Slide 7

Functions in C

- Functions in C are conceptually much like functions in other procedural programming languages. (Functions 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 8

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 9

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 10

Library Functions in C

- C does include a library of standard functions, though it's not 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 11

Simple Input in C — One More Thing

- Now that we know about conditional execution and functions, we can talk about what `scanf` does with invalid input: It returns a value, namely the number of items successfully processed.
- Typical usage:

```
int n;
if (scanf("%d", &n) == 1) {
    /* use value */
}
else {
    /* error */
}
```

Slide 12

Functions in C — Example(s)

- Examples as time permits.

Slide 13

Minute Essay

- What (if anything!) did you find interesting and/or difficult about Homework 1?