Slide 1

Administrivia

- Reminder: Midterm exam next Tuesday. Review sheet on Web. Also on Web — quiz solutions, homework solutions (not all yet), sample programs.
- Watch your mail for possible information on graded homeworks.

Slide 2

A Little About the Midterm

- Review class notes, example programs from class, minute essays, quizzes, and homeworks. Focus more on those than on the textbook.
- Most questions will likely be more difficult (or at least longer) than quiz questions, but similar in format. Might be a few short-answer / multiple-choice questions too.
- Open book, open notes, but no access to Web, so if you want material from the course Web site, please print out hardcopy. (You can use our printers.)
- (Topic by topic through the review sheet.)

Slide 3

Functions in C — Recap

- Functions in C (as in other programming languages) are a way to break up a big problem into more manageable pieces and also to avoid duplication of code/effort.
- The basic idea is similar to mathematical functions (something that transforms input(s) to output), but functions in C (again as in many — though not all! — programming languages) can have “side effects”.

Slide 4

Functions and Scope

- In addition to having a type and a name, each variable has a *scope* in which it’s valid. Variables declared inside a function can be used only within that function. Variables declared outside all functions can be used anywhere — *global variables*, almost always a bad idea.
- One result — variables with the same name in different functions are different variables.

Functions and Parameters

- We said last time that functions have *parameters*. Another word for them is *arguments* (you will see this is some compiler error messages). More terminology:
  - *Formal parameters* are the parameters as viewed from the function — can think of these as additional variables whose scope is the function.
  - *Actual parameters* are the values with which the function is called.
- When a function is called, actual parameters are copied to formal parameters — “pass by value”, meaning that changes made in the function to its copies are not reflected in the calling program’s copies.

Slide 5

Function Return Values

- Most functions return a value (but only one); it’s the value of the expression following the keyword `return`, in the function definition. The type of this value is given as part of the function definition. If you don’t want to return anything, can make this `void`. If you want to return two things? Later . . .
- Function calls are expression, so they have a value — whatever is returned by the function.

Slide 6

Slide 7

C Library Functions, Revisited

- We've used `scanf` and `printf`. Notice that both of them return a value. We can use this to check whether the input from the user is what we wanted (e.g., a number rather than text).
- Another useful library function — `sqrt`.
- Its `man` page describes what it does, and also tells you two things you need to do to use it:
  - In your program, `#include <math.h>`.
  - When you compile, `-lm`, e.g.

    ```
    gcc myprogram.c -lm
    ```

    (You may not need this, unless you get an error message about `sqrt` being undefined. Seems to depend on version/setup of compiler.)
- Example — program to compute hypotenuse of a right triangle. Other example(s) as time permits.

Slide 8

Minute Essay

- What are you finding most difficult about the homeworks so far? most interesting or helpful?