Administrivia

Slide 1

- Homework 1 on the Web. Due next week. Meant to be fairly easy starter problems; you can (and probably should) do them using only what we've talked about up through today's lecture.

  Turn in by e-mail. *Please* mention course name/number and assignment in subject line.

- I say in the syllabus that I try to respond promptly to e-mail. Exceptions are minute essays and homeworks, which I don't always look at right away. If you need a quick reply, make that apparent on the subject line please!

C Basics — Quick Overview

Slide 2

- Unlike Python and Scala scripts (but like Java programs), C programs include some standard boilerplate (`#include` for library functions, explicit `main`).

- Variables must be explicitly declared (including type).

- Expressions similar to those in Python/Scala/Java but with a few differences.

- Statements are also similar, but assignments are considered to be expressions too, with a value. Allows chaining, e.g.,

```
a = b = 10;
```

Expressions in C, Recap/Review

- C (like many other programming languages) has a notion of an *expression*.

- Every expression has a *value*, and computing this value is called *evaluating the expression*. Evaluate the above expressions,

Slide 3

- Sometimes evaluating an expression also produces changes to variables in the expression or other variables; these are called *side effects*. E.g., a call to `printf` is an expression; evaluating it produces a result (yes, really!) and a side effect.

- Many, many operators of different kinds. For now we'll look only at the ones for arithmetic.

Arithmetic Expressions — Operators

- Usual arithmetic operators +, −, * (multiplication), / (division). (+ and − can be unary too.)

  Notice that division, applied to integers, discards any remainder. This is so the result will be an integer too, and can even be useful. What if you want a fraction? Later.

Slide 4

- Also % operator for getting remainder; e.g., x % 2 is 0 if x is even, 1 if it's odd.

- Other useful arithmetic operators include pre/post increment/decrement, bit shifts.

- Expressions can be quite complex. How they're evaluated depends on rules of precedence and associativity. My advice — when in doubt, use parentheses! Example: $(x + y) / 2$ versus $x + y / 2$.

Expressions — "Caveat Programmer"

Slide 5

- C standard is somewhat imprecise about details of expression evaluation —
  e.g., in evaluating

  `f() + g()`

  two functions could be called in either order. (Why? To allow greater flexibility
  for implementers, possible allow for more-efficient programs.)

- C syntax allows programmers to write statements/expressions in which a
  variable's value is changed more than once, e.g.,

  `i = (i++) + (i--);`

  Syntactically legal, but standard says that such expressions invoke "undefined
  behavior". Best to avoid that!

Statements in C

Slide 6

- C programs are made up of *statements* (usually collected inside *functions*).

- Statements come in several types:

  - Null (`;`).

  - Expression (*expression* `;`).

  - Return (`return` *expression* `;`).

  - Compound (more later).

Simple Output

- Simple/typical way to produce output (to "standard output" — terminal for now) is with library function `printf`.

- Parameters are "format string", which may include "conversion specifications", followed by zero or more expressions, one for each conversion specification. E.g., to print value of `int` variable `x`:

  `printf("the value of x is %d\n", x);`

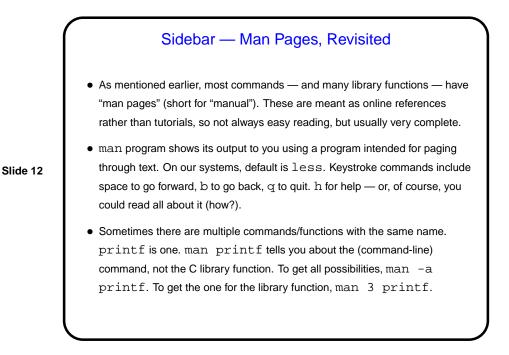 Full details in `man` page for `printf`. (Find with `man 3 printf`.)

Slide 7

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 8

Slide 9

Simple Input

- Simple way to get integer/float input (from "standard input") is with library
  function `scanf`. Parameters are "format string" (similar to the one for
  `printf`) and list of pointers (more later) to variables, e.g.:

  `scanf("%d %d", &var1, &var2);`

  Behaves somewhat like library functions for reading from standard input in
  other languages, except that it skips whitespace (including newlines) and
  stops when it encounters something other than what it needs (e.g.,
  non-numeric characters when number is wanted).

- Considered as an expression, call to `scanf` has a value, namely the number
  of variables successfully read. C-idiomatic way to check for success is

  `if (scanf("%d %d"&var1, &var2) == 2) ....`
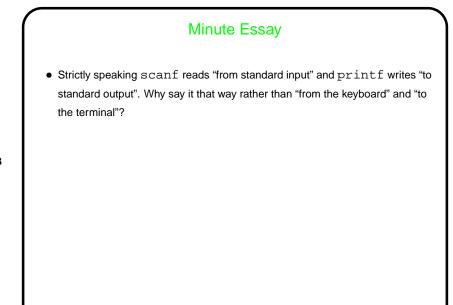
Slide 10

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 11

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 12

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`.

Minute Essay

- Strictly speaking `scanf` reads "from standard input" and `printf` writes "to standard output". Why say it that way rather than "from the keyboard" and "to the terminal"?

Slide 13

Minute Essay Answer

- By default "standard input" is the keyboard and "standard output" is the terminal, but I/O redirection allows for other possibilities (from/to files, pipes, etc.).

Slide 14