Slide 1

Administrivia

- Reminder: Homework 1 due today, 11:59pm.
- Homework 2 on the Web; due next week. First programming homework!
- I've put a copy of the textbook on 1-day reserve at the library, for anyone who doesn't have one (yet?).
- About the readings, it's probably not necessary to read every page carefully. You might wait until after class to read, paying particular attention to anything that was mentioned in class but that wasn't clear to you.

Slide 2

More Administrivia

- When turning in homework, please do identify in the subject line both the course and the assignment. I ask for this so it's easy for me to drop it into the right "folder" for grading — I'll be getting assignments by e-mail for four courses this semester!
- As with the minute essays, if there's an urgent question in what you're sending, put "question" or "urgent" in the subject line.
- If you need/want to mail homework to me when you're logged in remotely (or if you just want to know one more thing you can do from the command line!), see the `mail-files` script on the "sample programs" page.

Slide 3

Minute Essay From Last Lecture

- Many people said what we had done in C so far looked at least a little like what they had done in Matlab, only more detailed/tedious. I don't know much about Matlab, but my guess is that it lets you do easily things that would take more work in C, but that C gives you more flexibility and may be more efficient.

Slide 4

Defining Named Constants with Preprocessor Directives

- Sometimes it makes sense to use numeric constants in programs — e.g., in "count change" example.
- But sometimes it's more readable, for humans, to give these constants a name. Can do this with `#define`. Examples (somewhat contrived):

  ```
  #define NICKEL_CENTS 5
  ```

  Then when you write `NICKEL_CENTS`, compiler (strictly speaking, its preprocessor) replaces it with 5.

Slide 5

A Few Words About "Readability"

- In writing classes you're probably encouraged to keep in mind your audience. Applies to writing programs too — but you're writing for two audiences!
- One is the compiler, which is incredibly picky about some things (e.g., variables `X` and `x` are different), but doesn't care at all about others (e.g., names of variables in general, as long as they meet its restrictions).
- The other is any human readers who might look at your code — me, tutors you ask for help, even *you* when what you did is no longer fresh in your mind. Humans aren't as apt to notice syntax glitches but care about other things . . .

Slide 6

Writing Code for Human Readers

- Use meaningful variable names.
- Use consistent indentation that reflects program structure. Consider adding whitespace between logical parts of the program.
- Use comments to explain anything not apparent from the code. I like to start each program with a header block of comments that say what the program does (not *how*, but *what*.)

  Note that program examples in tutorials and textbooks often have more comments than programs written for other purposes. Consider their audience!

Slide 7

Types and Type Conversions

- Textbook goes into some detail about what happens when you have an expression that mixes types. Generally speaking what happens is the reasonable thing — types are converted as needed.
- Can also explicitly ask the computer to convert, with a "cast", e.g.

  `(float) 10 / 4`

  divides a floating-point 10 by 4.

Slide 8

Operator Precedence

- Useful to know that C has rules for the order in which operators are applied, e.g., in the expression

  `x * 2 + 1`

  the multiplication happens first.

  I don't find it useful to try to remember all the rules, though others might.
- Can override the default precedence with parentheses, and it can be a good idea (for readability) to use parentheses any time you're not sure.

Slide 9

Compound Assignment and Prefix/Postfix Operators

- Often happens that we want to apply some operation to a variable and put the result back in the variable (as with `remainder` in the counting-change program).
- Syntax for this is, e.g.,

  `x += 1` to add 1 to `x`
- C also has prefix/postfix operators `++` and `--`. Often used purely for their side effects, e.g.,

  `++x` and `x++` both add 1 to `x`

  What's the difference? Considered as expressions, the value of `x++` is the old value, the value of `++x` the new value.

Slide 10

Binary Numbers

- We humans usually use the decimal (base 10) number system, but other (positive integer) bases work too. (Well, maybe not base 1.) Binary (base 2) is more widely used in computers because it makes the hardware simpler.
- In base 10, there are ten possible digits, with values 0 through 9.

  In base 2, there are 2 possible digits (*bits*), with values 0 and 1.
- In base 10, 1010 means what? What about in base 2?

Slide 11

Converting Between Bases

- Converting from another base to base 10 is easy if tedious (just use definition).
- Converting from base 10 to another base? Let's try to develop an "algorithm" (procedure) for that . . .

Slide 12

Decimal to Binary, Take 1

- One way is to first find the highest power of 2 smaller than or equal to the number, write that down, subtract it from the number, and continue:
  1. If $n = 0$, stop.
  2. Find largest p such that $2^p \leq n$.
  3. Write a 1 in the p-th output position.
  4. Subtract 2^p from n.
  5. Go back to first step.
- Is this okay? What's not quite right about it? (We don't say what to put in the positions that don't have ones in them.)
- (Example.)

Slide 13

Decimal to Binary, Take 2

- Another way produces the answer from right to left rather than left to right, repeatedly dividing by 2 (again n will be the number we want to convert):
  1. If $n = 0$, stop.
  2. Divide n by 2, giving quotient q and remainder r.
  3. Write down r.
  4. Set n equal to q.
  5. Go back to first step.
- Is this okay? What's not quite right about it? (We don't say to write down the remainders from right to left.)
- (Example.)

Slide 14

Recap

- Key ideas here — break problem down into a sequence of steps that we hope don't require much intelligence, just an ability to calculate, with some decision-making and repeating.
- Before moving back to programming and C, a little more about different number bases and how binary numbers are used to represent data . . .

Slide 15

Minute Essay

- What is 1011_2 in base 10?
- What's the (base 10) value of the largest number you can represent with 4 bits? (E.g., the largest number you can represent with 2 bits is 11_2, or 3_{10}.)

Slide 16

Minute Essay Answer

- 1011_2 is 11_{10}.
- 15 (1111_2).