Slide 1

Administrivia

- Reminder: Quiz 3 Friday. Likely topic is loops.
- Reminder: Homework 5 due Friday.
- Review sheet for midterm on the Web. Mostly about exam format but also has list of topics.

  Aside: In general my idea is that students who have kept up reasonably well with reading and homeworks won't have to spend a lot of time preparing for exams. The goal is to test whether you understand the material rather than whether you can memorize!

Slide 2

Minute Essay From Last Lecture

- Several people mentioned working with different kinds of series and sequences.
- A couple of people mentioned creating data to be plotted. We'll do something like that in a homework later.
- Possibly I should ask this question again later?

Slide 3

More Administrivia

- If you get one of those mailed-every-Saturday messages about disk space usage on your account, you probably should pay attention: More than one of you has already run into problems related to being over quota.
- The message has a link to a "FAQ" page with more information, which I've just updated a bit. If you don't know what the message means or what to do about it, read this FAQ first and then feel free to ask.
- "TL;DR" summary: In a terminal window, first type `quota` to see current usage. Then type `sorted-disk-usage .` to see where the space is being used. If the last (largest) things shown are `.mozilla` or `.cache` (*highly* likely), next try `clear-browser-caches` and then check whether that helped with `quota` again. If that doesn't fix things, read that FAQ or ask.

Slide 4

Numerical Computation — Review

- A big use of computers is in solving (exactly or approximately) mathematical problems — "numerical computation" or "numerical analysis". Matlab is one tool for this, and/or you can write your own programs in a general-purpose programming language (such as C!). Often (maybe always?) these involve various forms of repetition.
- An example is "numerical integration", in which you approximate a definite integral (area under a curve) by computing areas of rectangles and adding them up. As an example . . .

Slide 5

Numerical Integration — Approximating π

- An exact value of π can be obtained by evaluating

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

  (If you don't remember, or never learned, what this means, no worries. For purposes of this class all that matters is how we do the approximation.)

- So we could approximate π by approximating the area under this curve.
- (Aside: This turns out to be a good introductory example of "parallel programming" because it lends itself to solutions involving multiple processing elements. !)
- How does this look in C. . .

Slide 6

Another Loop Example — Loop Until "Convergence"

- It's not atypical to want to repeat something until some computation "converges".
- As an example, we could revise the example we just wrote to do the computation repeatedly until some condition is reached.
- We could repeat until previous and current computed values are close.
- Or we could repeat until computed value is close to best-available library value for π. (Surprisingly, there's a library constant `M_PI`, but it isn't standard, so use `acos()` to compute.)

Slide 7

A Little About "Random" Numbers

- Among the C library functions discussed briefly in the textbook chapter on functions are `srand()` and `rand()`. A few words about what they do ...
- First, what we mean by "random" is (I think!) an interesting question with no obvious answer. What's often wanted is something that can't be predicted, and it's not clear we can get that with a system that's deterministic. Further, even if we could, we might not want that, since we often want to be able to repeat a test.
- So, often what we really want is a "pseudo-random number generator" — something that generates a sequence of numbers that looks random but is repeatable given some reproducible starting point.

Slide 8

Pseudo-Random Number Generators

- Mathematically interesting topic; classic reference is in one of the volumes of Donald Knuth's *The Art of Computer Programming*.

  (Aside: Some of you may know Knuth as the inventor of the typesetting system TEX. It's an extreme example of a "side project" that turned into something much more?)
- Early researchers apparently thought more-complex algorithms would give better results, but — not necessarily. Very simple algorithms can give quite good results. For example, one reasonable one (not the best, but good) computes each element of the sequence in terms of the previous one:

  $x_{n+1} = (ax_n + b) \bmod M$

  for carefully selected values of a, b, and M.

Slide 9

Pseudo-Random Number Generators, Continued

- Uses in programming include simulating various things in the physical world. Textbook examples often involve simulating rolling dice, shuffling cards, etc.
- (Example next time.)

Slide 10

Minute Essay

- Do you compile with just `gcc`, or `gcc -Wall`, or do you use `make`?