Slide 1

Administrivia

- Homework 5 on the Web. Due next Friday.
- Instructions for using “mail files” script corrected.
- Notice that the programs linked from the course “sample programs” are slightly-cleaned-up versions of what we write in class. In class I don’t take time to write adequate comments or error messages, which is fine in context but not something you should emulate!

  One more version of “get integer” program added, reworking a bit to avoid what I thought was ugly duplication in the function to collect digits.

Slide 2

More Administrivia

- If you get one of those mailed-every-Saturday messages about disk space usage on your account, you probably should pay attention: At least two of you have 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.

Slide 3

Minute Essay From Last Lecture

- Possible uses mentioned for repetition/looping include Riemann sums and Taylor series, getting inputs from sensors and discarding until one of interest is found, determining position-in-space by repeatedly getting information from a satellite.

Slide 4

Loops — Recap/Review

- Loops, like recursion, are a way to repeat some operation. Useful in applying the same operation to all elements of some collection or in repeating an operation until some condition is met.
- What all these ways of repetition have in common:
  - A starting point (initial condition, first element of a collection).
  - The operation to repeat.
  - How to move from one iteration to the next.
  - When to stop (though the syntax often is such that what you actually say is when *not* to stop).
- Last time we looked at basic syntax for `for`, `while`, and `do while` loops. When to use which one? “it depends”, and sometimes a matter of style, but if you know how many times you want to repeat something, a `for`

Slide 5

loop is probably more idiomatic, while if you don't, a `while` or `do while` is probably better.

Slide 6

Loops Versus Recursion

- As noted in class, recursive functions can be simple to write but potentially inefficient (though in some cases a sufficiently smart compiler can reduce or eliminate the inefficiency — look up "tail recursion" to find out more).
- For other problems, a loop is simpler to write — loop versions of many of the in-class examples of recursion are as simple or simpler, and that program to get an integer from input without using `scanf` would have been simpler. So it may seem that loops are better.
- But there problems for which recursive solutions are much simpler to write and get right, while non-recursive solutions are decidedly not simple — anything involving "trees", plus at least two widely-used algorithms for "sorting" (putting things in order).

Loops — More Examples

- First let's modify the "sum of integers" program to compute an average. Both programs are examples of what one might call a "running total" pattern.
- Now let's rewrite that "get an integer" program to use a loop or loops.
- Other examples . . .

Slide 7

Numerical Computation

- 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. 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 . . . (next time).

Slide 8

Minute Essay

- Make your best guess in the few minutes available at writing a C function to print the squares of numbers from 1 through n:

```
void print_squares(int n);
```

- Any questions about loops (or anything else)?

Minute Essay Answer

- Here is one solution:

```
void print_squares(int n) {
    for (int i = 1; i <= n; ++i) {
        printf("%d\n", i*i);
    }
}
```