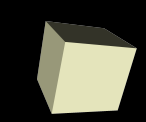
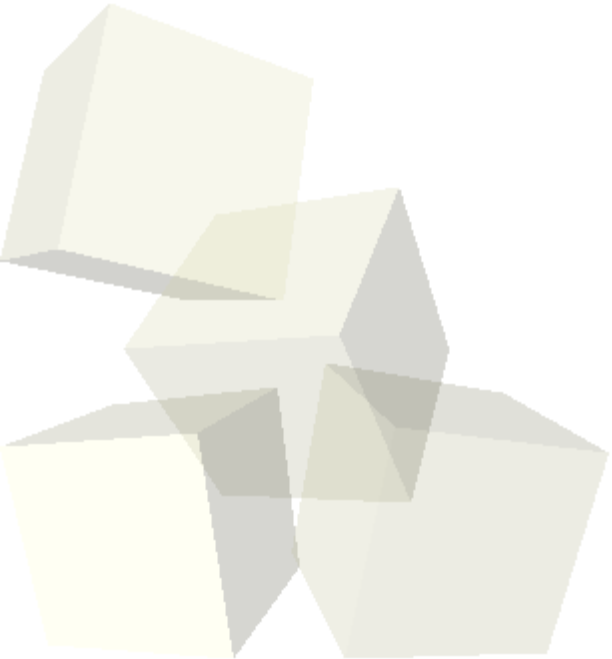


1-D Arrays and Sorting

10-5-2006



Opening Discussion

- What did we talk about in the class before the test?
- Do you have any questions about the test?
- What does !<Bash Command> do in vi?
- There is an ACM meeting in HAS 329 this afternoon. If you have an interest in CS I strongly encourage you to attend.

A Standard Problem

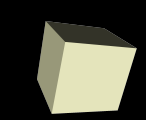
- Imagine a program where I wanted to be able to give it a set of number then ask the user what types of calculations to do on the numbers. Those calculations might include the sum, the standard deviation, the min, the max, the median, the mode.
- How would you do this for a few numbers? How would you do it for a user specified number of numbers?
- The problem is that the only way we can get memory to store things is by declaring individual variables and the only way we can refer to those variables is by name.

Arrays

- The solution to our problem is to use arrays. An array is a group of memory cells all of a particular type that can be indexed by an integer value.
- That is to say that we can declare one variable that represents many values and we refer to each value by combining the name with a number.
- What does the syntax for arrays look like in C? How do we declare an array? How do we use an array?
- Let's write a little program that will read values into an array.

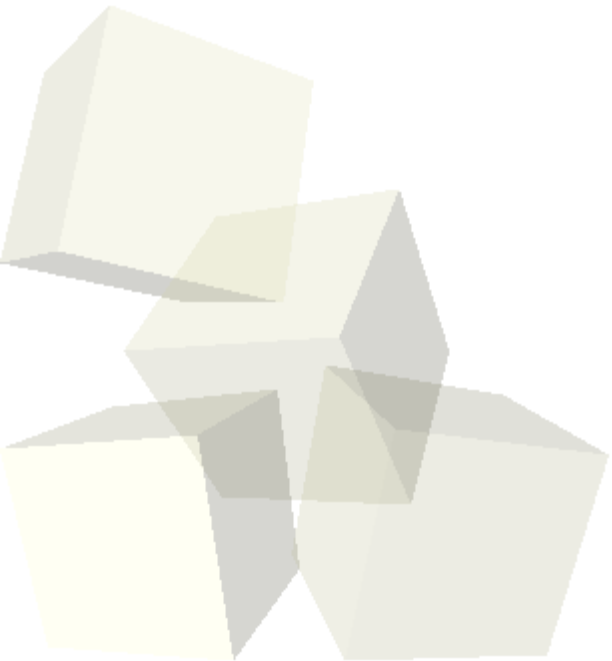
Strings

- We don't officially talk about strings for another 4 weeks, but they are so useful I want to briefly introduce them.
- We have worked with string literals where we put characters between double quotes. This actually gets translated to an array of characters that ends with a 0 (ASCII value 0 or character '\0'). We call them null terminated strings.
- You can declare an array of characters and use %s with scanf or printf to interact with it.
- Let's look at a simple example of this.



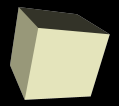
Sorting

- If I give you a stack of folders and tell you to alphabetize them, how would you do it? Can you express what you would do as an algorithm we could use on an array?



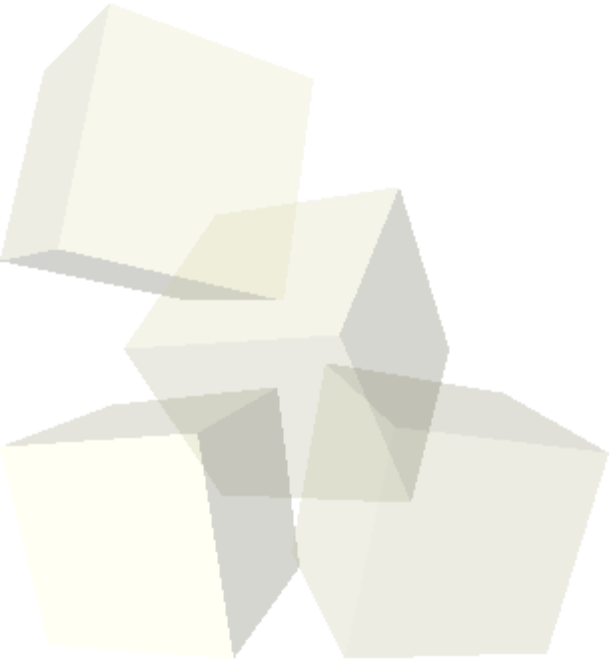
“Simple” Sorting Algorithms

- These are easy to code, $O(n^2)$ sorting algorithms.
- Bubble sort
  - Go through the array comparing adjacent elements. Swap them if they are out of order. Repeat this until done.
- Selection sort
  - Can be a min or max sort. Go through the array and find the minimum. SWAP it into place. Repeat until done.
- Insertion sort
  - Probably what you do with the folders. Look at each element in turn and swap it forward until you get in the right position.
- Let's write these.



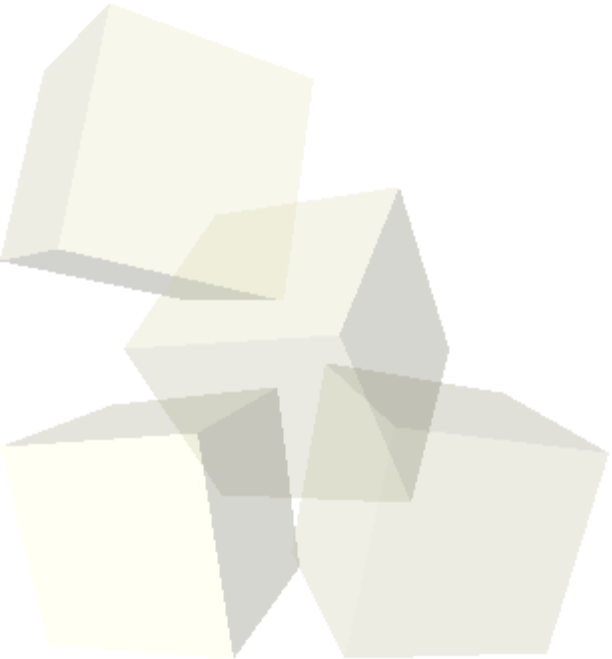
Faster Sorts

- We can make faster sorting algorithms that operate in O(n log n) time. They are more complex to code but well worth it if you are going to sort a lot of data.
- Merge sort and quicksort both use recursion to sort things efficiently.



Searching/Multidimensional Arrays

- The primary reason for sorting things is so that you can find what you want more quickly.
- Next time we will talk about algorithms for searching through sorted and unsorted data.
- We will also look at how to make arrays with more than one dimension and what we would use them for.



Minute Essay

- Assume you have the array {7,4,1,6,2,3} and you run a bubble sort on it. What does the array look like at the end of each iteration of the outer loop?
- There is no class next week. I'll try to get some options for assignment #4 posted so you can work on that next week.
- ACM meeting this afternoon here in Halsell.