CS1003/1004: Intro to CS, Spring 2004

Lecture #10: Data structures II

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW#4 due next Tuesday
- I'll be in Seattle next week; Suhit will lecture in place of me
  - He's eminently qualified for the next topic
- 1 point on midterm problem 3…

Custom data types

- Wouldn't it be nice for HW#3 to have a single "entity" to refer to bank account, so we can have an array of *bank accounts* instead of two separate arrays?
- We can declare such a *structure* (C) or *object* (Java)
  - We'll set it up so that it contains a String and a double
  - We then access *components* of that "bank account"
- You should be learning language-specific skills for this now

How complicated?

- Data structures & types can be almost as complicated as you want
- You can nest complex data structures
  - For example, a bank account can contain an array of dependents
  - You can have an array of bank accounts in a Branch
  - You can have an array of Branches in a BankInstitution
  - And so on…
- How can we organize all this stuff!?
  - Take CS3134, and you'll learn all the details. Here's a few.
  - You won't have to worry about the implementation details – we're focusing only on the basic concepts

"List" data abstraction

- The most common way to organize things is in a list
  - An array is one type of a list – it's *static* sizewise; "contiguous list"
- What are basic *conceptual* operations on a list?
- How do these conceptual operations work with an array?
- Can we organize lists in any different fashion?

Linked List

- Idea: instead of allocating *one* block of memory and dividing it into individual cells, create lots of individual scattered cells and connect them together in one long chain
- Advantages:
  - Infinite-length – just allocate another block
  - Easy to insert or remove an element in the middle
- Disadvantages:
  - Lots of memory management

Stacks and Queues

- Variation on lists to support specific problems
- Stacks follow a *LIFO* policy (last-in, first-out)
  - "Push" and "pop" operations
- Queues follow a *FIFO* policy (first-in, first-out)
  - Enqueue, dequeue
- Both have numerous applications in computing
  - Stacks used to keep track of procedure calls
  - Queues used for print queues

Trees

- Instead of just a linear data structure, why can't we have something more flexible?
- Trees are called such because they have nodes that are arranged into a hierarchy with a *root*, *leaves*, and *children*
- Most popular kind of tree is a *binary* tree, where every node has two children
- Binary *search* trees provide faster ways to search of information: O(log n) for insert, remove, search

Yes, this is a whirlwind tour

- Data Structures, W3134, covers all of these in much greater detail, including implementation
- Just make sure you understand the concepts and the basic algorithms involved with them
- Brookshear has a decent discussion of these

Next time

- Suhit will teach you guys the basics of a computer (i.e., computer architecture)