CS 241 — Introduction to Problem Solving and Programming

Applied Topics

Toward a better array, part I:
Linked lists

April 18, 2005

Arrays

What are arrays good for?

What's bad about arrays (even when they are the "right tool" for the job)?

Arrays

Use arrays when your data

- Is uniform
- Is sequenced
- Needs to be treated uniformly
- Needs to be iterated (looped) over
- Has a size unknown until runtime

Problems with arrays

- Arrays cannot grow
- You cannot add to the middle of an array
- The length is not necessarily equal to the number of legitimate elements
- You must do everything (eg, finding a specific element) yourself

Arrays

Solution:

Write your own class that works like an array (and is essentially a wrapper for an array) but allows for shrinking and growing, inserting and deleting, etc.

Write this once for all, and use it forever, never needing to use an array again.

Or better yet, don't write your own class but use the `Vector` class Java already provides

Drawbacks

- `Vectors` take up a lot of memory
- Storing primitives in a `Vector` is a hassle (and re-writing it for primitives would be worse)
- Even though `Vectors` can do it, inserting and removing is grossly inefficient; if you have to do it a lot, don't use a `Vector`

Linked lists

Solution:

Wrap (contain) each item in an object called a node, and let each node have a reference (or link) to the next node.

Write a class to manage this collection of nodes (potentially with an array-like interface), maintaining (at least) a reference to the first node (or head).

A structure like this is called a linked list.

The last node in the list is called the tail.

Lists

Variations

- References to head and tail
- Links to previous as well as next
- Tail links back to head (a ring)
- More than one "next" node (a tree)

Summary

- Node
- Link
- Linked list
- Head
- Tail
- Traversal— iterative and recursive