

Class 44: Introduction to Sorting

Held: Friday, April 20, 2007

Summary: Today we visit the problem of *sorting*. When you sort a list or vector, you put the elements in order (e.g., alphabetical, numerical, ...).

Related Pages:

- EBoard.

Due

- HW 15

Notes:

- I have been asked to meet with some visitors to campus today, so I will not be available in class or during office hours. I apologize.

Overview:

- The problem of sorting, revisited.
- Writing sorting algorithms.
- Examples: Insertion, Selection, etc.
- Formalizing the problem.

The Problem of Sorting

- As we saw recently, one problem that seems to crop up a lot in programming (and elsewhere) is that of *sorting*.
- The problem: Given a list, array, vector, sequence, or file of comparable elements, put the elements in order.
 - *In order* typically means that each element is no bigger than the next element. (You can also sort in decreasing order, in which case each element is no smaller than the next element.)
- We'll look at techniques for sorting vectors and lists.

Designing Sorting Algorithms

- I suggest that you think about the development of sorting algorithms in Scheme similarly to the way you think about writing many algorithms.
- Start by thinking about the way you might do it by hand.
 - We may find a few different ways to sort by hand.
 - We'll probably leave the Scheme-ification to the end.

- Postconditions?
 - Here are some postconditions I typically think about:
 - You also need to ensure that all elements in the original list are in the sorted list.
 - You also need to ensure that no other elements are in the list.
-

Copyright © 2007 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.