CSC161 2010F *Imperative Problem Solving*

Class 45: Hash Tables, Continued

Held: Friday, 19 November 2010

Summary: We continue our exploration of hash tables by writing code for one.

Related Pages:

- EBoard.
- Reading: K&R 6.6.

Notes:

- A Kington story.
- Assignment 9 is now ready.

Overview:

- Review.
- Collisions.
- A Hash Functions.
- Building Code.

Review

- *Dictionaries* are a common ADT. Goal: Provide "indexed" access where the index is a string, rather than a number.
  - We call the index the *key*.
  - We call the associated value the *value*.
- Two basic operations: insert and find.
- Lots of other possible operations.
- We often implement dictionaries with *hash tables*.
- In a hash table, we use a hash function to convert the key to an integer index.
- To store a value, we add a key/value pair to that index.
- To retrieve a value by key, we look in that index and make sure that we have the matching key.

A Problem: Collisions

- There's a big potential problem with hash tables: What happens when two keys hash to the same index?
- There are two possible solutions:
  - Systematically look elsewhere in the hash table
  - Store a list of key/value pairs in the array

- K&R use the second technique

A Hash Function

- I've given you the K&R hash function in slightly better form.
- We'll explore it a bit.

Building a Hash Table Library

- Okay, let's get to work building a hash table library.
  - We'll mix tasks with issues in C that we need to figure out.
- First, we'll specify the data type and the functions.
- Next, we'll build a test progrma or two.
  - One interactive
  - One unit test
- Then we'll work on the details of the implementation, stealing from K&R as we need to.
