CSC161 2011S *Imperative Problem Solving*

Class 41: Pointers to Functions

Held: Monday, 18 April 2011

Summary: We consider pointers to functions in C.

Related Pages:

- EBoard.
- Lab: Function Pointers.
- Reading: K&R 5.10-5.11.

Notes:

- I have not yet responded to your current set of reading questions. Sorry.
- Reading for tonight: K&R 7.5-7.7.
- Since some of you asked: There will be a new assignment due next week. Then I'll distribute the 3rd exam. Then there will be one final assignment.
- Due to prereg and College committee work, I need folks to sign up for appointments. Times at http://www.cs.grinnell.edu/~rebelsky/Schedule/signup.2011S.txt. Signup via email.
- EC for any of the Activism Week activities, e.g., Sarah Smith on Wednesday. [academic]
- EC for Ginsburg convo, Thursday. [academic]
- EC for open discussion with Jane Ginsburg, Thursday, 9:00-10:30 in Mears Cottage. [academic]
- EC for open discussion with Jane Ginsburg, Thursday, 2:30-4:00 in Mears Cottage. [academic]
- EC for Benji's films event Saturday at 4. [peer]
- EC for Spring Bling on Friday. [peer]
- EC for Grinnell Monologues on Friday/Saturday. [peer]

Overview:

- Function Pointers.
- Echo.
- Quicksort.
- Lab.

Function Pointers

- In C, "pointer to function" is a valid type.
- We declare that a parameter has that type with

```
RETURN_TYPE (*NAME)(PARAM_TYPES)
```

- You can assign, pass, and call function pointers.
- How do function pointers in C seem to differ from function parameters in Scheme?

echo

- The section on `echo` isn't relevant to function pointers, but there were some questions, so ...

```
/**
 * echo arg1 ... argn
 *   Print out the command line arguments.
 */
int
main (int argc, char *argv[])
{
  while (--argc > 0)
    printf ("%s%s", *++argv, (argc > 1) ? " " ; "");
  printf ("\n");
  return 0;
} // main
```

Quicksort

- Quicksort is one of the better sorting algorithms available.
  - It doesn't use extra memory.
  - It's *usually* fast (as fast as merge sort).
- Quicksort was invented (discovered?) by C. A. R. Hoare.
- Quicksort is based on two key design principles.
  - Divide-and-conquer.
  - Unpredicatability is your friend.
- Like Merge Sort, Quicksort divides an array into two halves and recurses on the two halves.
- In contrast to Merge Sort, Quicksort divides the array into "small values" and "large values".
- How do we divide it?
  - We pick some "random" value (the "pivot") as the boundary between small and large.
  - We rearrange *in place*.
- How do we merge it?
  - We don't have to. Since we've rearranged, after the recursive sorts are done.
- Overall strategy

```
Quicksort (stuff)
  if zero or one elements
    done!
  pick a pivot
  rearrange into small and large elements
  sort the two halves
```

- The details are slightly more complicated.

Lab

- Do The lab.
