Introduction to Computer Science and Programming in C

Session 20: November 13, 2008
Columbia University
Announcements

- Homework 4 is out, due last day of class: December 4 before class
- Final Exam: Tuesday, 12/16, 1:10 pm - 4:00 pm
  Mudd 233 (our normal room)
Review

- Modular Programming (almost object oriented)
  - Think of programs as modules/objects interacting
- Makefiles
  - Use “make” to automate compilation
Today

- Homework 3 solutions
- Revisiting pointers
  - Pointers to pointers
  - Pointers to functions
Homework 3
Solutions

http://www.cs.columbia.edu/~bert/courses/1003/homework3_soln.txt
Pointers to pointers

- Recall that C arrays and pointers are basically the same:
  ```c
  int A[10];
  int *A_ptr = A;
  ```

- How does C store 2d arrays?
  ```c
  int B[10][10];
  ```

- **B** is a pointer to an array of pointers
Pointers to pointers

B

int **

B[0]

B[1]

B[2]

B[3]

B[4]

int *

B[0][0] B[0][1] B[0][2] B[0][3] B[0][4]


int

7
We can dynamically allocate multi-dimensional arrays

```c
int **C;
C = (int**) malloc(N*sizeof(int*));

for (i=0; i<N; i++) {
    C[i] = (int*)malloc(N*sizeof(int));
}
```
Pointers to functions

- It is occasionally useful to use pointers to functions

- Since functions are stored in memory, we can reason about their addresses too

- This allows us to say, “run the function at address _____ on these arguments”

- Useful for being truly general, e.g. stdlib qsort
Function Pointer
Syntax

- `int (*f_ptr)();`
  /* pointer to function that returns an int */

- Parentheses are important. Without parentheses, `f_ptr` looks like it returns a pointer to an int.

- `int (*f_ptr)(int, int);`
  /* function takes 2 ints as arguments */

- `int greater_than(int a, int b);
  f_ptr = greater_than;`
qsort example

- Stdlib’s qsort function is a general sorting function.

- Sort an array of any type, using any comparison criterion

- Define that comparison as a function pointer

  void qsort(void *base, size_t n, size_t size, int (*cmp)(const void *, const void *));
qsort example

- Compare function should take two entries A and B,
  - return +1 if A>B
  - return -1 if A<B
  - return 0 if A==B
```c
int greater_than(const void *x, const void *y)
{
    float *a = (float*)x, *b = (float*)y;

    if (*a>*b)
        return 1;
    if (*a<*b)
        return -1;
    return 0;
}

void mySort(float A[], int N)
{
    int (*f_ptr)(const void *, const void *)
        = greater_than;
    qsort((void*)A, N, sizeof(float), f_ptr);
}
```
Reading

- The C Programming Language. Chapter 5 (describes pointers in great detail)