

ECE 220 Computer Systems & Programming

Lecture 13 – Problem Solving with Pointers and Arrays

October 12, 2017



- Quiz 3 next week
- Midterm 1 regrade due this Friday

Multi-dimensional Arrays Recap

int a [2][3];		Column 1	Column 2	Column 3
	Row 1	a[0][0]	a[0][1]	a[0][2]
	Row 2	a[1][0]	a[1][1]	a[1][2]

In memory

a[0][0]
a[0][1]
a[0][2]
a[1][0]
a[1][1]
a[1][2]

* multi-dimensional array is stored in **row-major** order

Exercise 1: implement a function that interchanges two rows of a 5x5 matrix. The function takes three arguments: pointer to the matrix, row number x and row number y.

```
void row_interchange(int matrix[5][5], int x, int y)
{
```

```
}
```

Exercise 2: implement a function that transpose an $n \times n$ matrix

```
#define ROW 3
#define COL 4
void transpose(int in_matrix[ROW][COL], int out_matrix[COL][ROW]){

}
```

```
#define ROW 3  
#define COL 4  
void transpose2(int *in_matrix, int *out_matrix){
```

```
}
```

1. Pointer Array & Pointer to an Array

```
int a[4];  
int b[5];  
int *ptr_array[2];  
ptr_array[0] = &a[0]; /* ptr_array[0] = a; */  
ptr_array[1] = &b[0]; /* ptr_array[1] = b; */
```

or

```
int a[4];  
int b[5];  
int *ptr_array[2] = {a,b}
```

2. Search Algorithms

Linear Search: search from the beginning of the array until item is found

Binary Search: (for sorted array)

- 1) find the middle of the array and check if it's the search item;
- 2) search the first half if the search item is smaller than the center item, else search the second half;
- 3) repeat step 1 & 2 until search item is found.

If searching for 23 in the 10-element array:

	2	5	8	12	16	23	38	56	72	91
	L					H				
23 > 16, take 2 nd half	2	5	8	12	16	23	38	56	72	91
						L			H	
23 < 56, take 1 st half	2	5	8	12	16	23	38	56	72	91
						L		H		
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91

Exercise: implement a function that performs binary search

This function takes two arguments: a pointer to the sorted array and the search item. If the search item is found, the function returns its index in the array. Otherwise, it returns -1.

```
#define SIZE 8  
int binary_search(int array[], int item)  
{
```

```
}
```


3. Sorting Algorithms

Bubble Sort: 1) compare items next to each other and swap them if needed;
2) repeat this process until the entire array is sorted.

Insertion Sort:

- 1) remove item from array, insert it at the proper location in the sorted part by shifting other items;
- 2) repeat this process until the end of array is reach.

Quick Sort: also called divide-and-conquer

- 1) pick a pivot and partition array into 2 subarrays;
- 2) then sort subarrays using the same method.

Sorting Animation - <http://visualgo.net/sorting>

Exercise: implement a function that performs bubble sort

This function takes one argument: a pointer to the array.

```
#define SIZE 5  
void bubble_sort(int array[])  
{
```

```
}
```

Insertion Sort

Step 1

Assume first item is "sorted"

5	2	6	1	3	9
---	---	---	---	---	---

Step 2

Identify the value to compare

5		6	1	3	9
---	--	---	---	---	---

2

Step 3

Since $5 > 2$, shift 5 over to create space for 2 in the sorted section

	5	6	1	3	9
--	---	---	---	---	---

2

Step 4

Insert 2 into the empty space in the sorted section

2	5	6	1	3	9
---	---	---	---	---	---

Exercise: implement a function that performs insertion sort

This function takes one argument: a pointer to the array.

```
#define SIZE 5  
void insertion_sort(int array[])  
{
```

```
}
```