CSE 142
Programming I

Multidimensional Arrays

Arrays So Far

- Arrays let us collect many pieces of information in one place
- Limited—we can only store a “list” of items
- What other kinds of tables might we wish to keep track of? (Think homework!)

2-Dimensional Arrays

- An ordered collection of data—just like the 1-D arrays that we’ve seen already
- Now they’ll be ordered in TWO directions
- All of the items still must be of the same type

2-D Array Picture:

score

test #

student	0	1	2
0	22	22	22
1	50	48	49
2	10	0	0
3	45	37	41
4	39	28	31
5	18	22	35
6	16	50	3
7	42	29	37

score[3][1]

Declaring a 2-D Array

```
#define NUM_STDNTS   8
#define NUM_TSTS     3

int score[NUM_STDNTS][NUM_TSTS];
```

- This declares the previous array

2 Views of 2D Arrays

- We can either think of scores as an 8 by 3 array of scores...
- ...or as an array of 8 one dimensional arrays of size 3

Accessing Array Elements

- We access the elements of the array just like we did for 1-D arrays, but now with 2 subscripts:

```
scores[5][1];
```

- As before, each element of the array behaves exactly like a regular variable

How To Read in A 2-D Array

- With 1-D arrays we used loops
- With 2-D arrays we'll use…?

Reading in an Array

- Read in test scores for 8 students:

```
int i, j, scores[8][3];
for (i=0; i<8; j++){
   for (j=0; j<3; i++){
        scanf("%d", &scores[i][j]);
   }
}
```


Printing the Array

- Same idea: loops!

```
for (i=0; i<8; j++){
   for (j=0; j<3; i++){
        scanf("%d", &scores[i][j]);
   }
}
```


Problem: Averaging

- How could we write a program to average all of the scores in our array?

2-D Arrays and Functions

- We can use 2-D arrays as parameters to functions as well
- We cannot return them from functions (just like regular arrays)

Swap Two Rows

```
void swap(int a[][NUM_TSTS], int
  row1, int row2){

  int i;
  for (i=0; i<NUM_TSTS; i++)
     swap(&a[row1][i], &a[row2][i]);

  return;
}
```


Representation of Arrays

- We think:

22	22	22
50	48	49

...

16	50	3
42	29	37

- Actually:

22	22	22	50	48	49	...

More than 2-D

- We can make arrays with as many dimensions as we wish
- If we use them as parameters, we must always give the function each subscript *except for the first one*
  - ➤Why? Think about the representation!