CS 111

2D arrays

Model declaration for array

BASE_TYPE ARRAY_NAME [ROW_CAPACITY] [COL_CAPACITY]

- This creates a table with ROW_CAPACITY rows and COL_CAPACITY columns, containing variables with BASE_TYPE

Model declaration for array

BASE_TYPE ARRAY_NAME [ROW_CAPACITY] [COL_CAPACITY]

- To process all elements in the 2D array we follow a nested loop
- Sometimes this should go row by row and sometimes column by column
- Before coding we should always decide which is more convenient

Model for row by row processing

```
for (int r = 0; r < ROW_CAPACITY; r++) {
    for (int c = 0; c < COL_CAPACITY; c++) {
        PROCESS ARRAY_NAME [ r ] [ c ];
    }
}
```

Model for column by column processing

```
for (int c = 0; c < COL_CAPACITY; c++) {
    for (int r = 0; r < ROW_CAPACITY; r++) {
        PROCESS ARRAY_NAME [ r ] [ c ];
    }
}
```

Sample questions

1. The array seatingChart has 5 rows, each of which has 10 columns. Each entry is a string. How do we declare it?

2. The array digits has 2 rows, each containing 5 columns of numbers. The first row is 0, 1, 2, 3, 4 and the second is 5, 6, 7, 8, 9. How do we declare and initialize it?

3. What's the output of the following code snippet?

```
int data[2][4] = { {3, 1,4, 1}, {2, 7, 1, 8} };
cout << data[1][1];
```

Sample exercises

- 2D array int data[2][4] = { {3, 1,4, 1}, {2, 7, 1, 8} };
- What is the for loop control to move through its elements to print it column by column?
- If we our goal is to print the sums of the rows of the array, do we still need a nested loop?
- If so, should it be row by row or column by column?

Print the sums of each row

```
int data[2][4] = { {3, 1, 4, 1}, {2, 7, 1, 8} };
int sum = 0;
for(int r = 0; r < 2; r++){
    sum = 0;
    for(int c = 0; c < 4; c++){
        sum += data[r][c];
        cout << data[r][c] << " ";
    }
    cout << "sum of row " << r << " is " << sum << endl;
}
```