Class 20 + 21

2D Arrays

(Modified slightly from Rebecca Schley's class)
2D Arrays

• An array is like a row of boxes

• A 2D array is like rows of boxes stacked on top of each other
2D Arrays

• Think of 2D arrays as an “array of arrays”
• A 2D array can be considered as a table, with rows and columns
• All elements in a 2D array must be of the same type, just as with 1D arrays
Notation

• Declaration
  • data_type array_name[rows][columns]

• Initialization
  • data_type array_name[rows][columns] = {{initialize row_1}, {initialize row_2}, ...
  ..., {initialize row_n}}

• Reference an individual array element
  • array_name[row_no][col_no]
Process 2D arrays

• For 1D arrays, process elements using loop
• For 2D arrays, process elements using nested loops
• Sometimes we process row by row, sometimes column by column

<table>
<thead>
<tr>
<th>row-by-row processing</th>
<th>column-by-column processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>for (int r = 0; r &lt; ROW_CAPACITY; r++) {</td>
<td>for (int c = 0; c &lt; COL_CAPACITY; c++) {</td>
</tr>
<tr>
<td>for (int c = 0; c &lt; COL_CAPACITY; c++) {</td>
<td>for (int r = 0; r &lt; ROW_CAPACITY; r++) {</td>
</tr>
<tr>
<td>PROCESS ARRAY_NAME [ r ] [ c ];</td>
<td>PROCESS ARRAY_NAME [ r ] [ c ];</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>
Example 1

• Declare and initialize 2D array
• Determine which row has the largest sum
• Plan:
  • Declare variables for maxSum and maxRow
  • Initial value for maxSum should be a sum of one of the rows
  • Iterate through 2D array row by row, summing the elements in each row
  • After summing a row, compare that row’s sum to maxSum
  • If maxSum < rowSum, update maxSum and maxRow to current sum and row values
Example 2

• Compute average value stored in each row of a 2D array

• Plan:
  • Declare 1D array called average with same number of boxes as rows in the 2D array
  • Iterate through each row of the 2D array, adding that row’s elements into a sum variable
  • After summing the row, divide the sum by the number of elements in the row and store the average for that row in average[r]
Example 3

• Determine which column of a 2D array has the largest value

• Plan:
  • Declare variables for maxSum and maxCol
  • Initial value for maxSum should be a sum of one of the columns
  • Iterate through 2D array column by column, summing the elements in each column
  • After summing a column, compare that column’s sum to maxSum
  • If maxSum < colSum, update maxSum and maxCol to current sum and col values
Example 4

- 3 students take 4 quizzes
- Read the scores, quiz by quiz
- Print the averages per student
1D arrays v 2D arrays

<table>
<thead>
<tr>
<th></th>
<th>1D example</th>
<th>2D example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>int nums[5];</td>
<td>int grades[2][3];</td>
</tr>
<tr>
<td>Initialization</td>
<td>int nums[5] = {1, 2, 3, 4, 5};</td>
<td>int grades[2][3] = {{1, 2, 3}, {4, 5, 6}};</td>
</tr>
<tr>
<td>Assign value to specific element</td>
<td>nums[2] = 15; // stores 15 in nums at index 2</td>
<td>grades[0][2] = 10; // stores 10 at row index 0, column index 2</td>
</tr>
<tr>
<td>Reference a specific element</td>
<td>cout &lt;&lt; nums[1];</td>
<td>cout &lt;&lt; grades[1][1];</td>
</tr>
<tr>
<td>Process array</td>
<td>Single for loop with counter initialized to 0</td>
<td>Nested for loops, with row and column counters initialized to 0</td>
</tr>
</tbody>
</table>