Two Dimensional Arrays

Topic 14

Objectives

- This topic should allow students to use two-dimensional arrays.
- This topic is covered in Chapter 9 of the textbook, after sorting and strings.

Tabular Data

- One dimensional arrays allow you to store lists of values of the same type.
- For tabular data, you need two-dimensional arrays.
- Consider a table of student exam marks per question.
  - Each row will represent one student script.
  - Each column represents an exam question.

Exam marks table

<table>
<thead>
<tr>
<th>Student</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>11</td>
<td>20</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>18</td>
<td>23</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>14</td>
<td>0</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>17</td>
<td>15</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>0</td>
<td>15</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Each cell contains a grade out of 25. 0 indicates the question not attempted.
Working with tabular data

- If we want to maintain this data so that we can calculate grades, averages, average per question, etc, we need to use a 2D array.
- As with 1D arrays, counting starts at 0.

As an array

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13</td>
<td>11</td>
<td>20</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>18</td>
<td>23</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>14</td>
<td>0</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>17</td>
<td>15</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>0</td>
<td>13</td>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

Declaring and using a 2D array

- Declaration:
  - double grades [numStudents] [NUMQs];
  - The first number is the number of rows.
- Accessing a slot:
  - grades [s][q]
  - refers to the s row's q column
  - cin >> grades [2][4];
  - tot = tot + grades [s][q];
Processing 2D Arrays
- Process the whole array using nested for loops
- Process rows of the array as though each was a one-D array.
- Process columns of the array as though each was a one-D array.

Passing 2D Arrays to functions
- When declaring a 2D array as a formal parameter you must tell the function how many columns the array has.
- This allows the compiler to work out where each row begins.
- The number of rows can be passed as a separate parameter.
- void print_grades (double grades [], int numStudents);
  Because of this use columns for fixed dimension (e.g. num qs) and rows for variable dimension (num students).

Sample functions
- print_array
- sum_a_row, sum_a_column
  - These are given the row/col number and return the sum
- sum_rows_into_array
  - Given a one-D array puts the row sums into the corresponding slot.

Parallel arrays
- The full program to work with exam data will need other arrays:
  - We’ll keep a separate one-D array to map row numbers to StudentIDs:
  - We might also need an array to store totals per student.
  - These are “parallel” arrays
    - Corresponding slots refer to the same entity.
    - In fact each slot corresponds to a row of the 2D array.

<table>
<thead>
<tr>
<th>ID</th>
<th>tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>9987</td>
<td>62</td>
</tr>
<tr>
<td>9768</td>
<td>41</td>
</tr>
<tr>
<td>9720</td>
<td>17</td>
</tr>
<tr>
<td>9901</td>
<td>44</td>
</tr>
<tr>
<td>9705</td>
<td>31</td>
</tr>
<tr>
<td>9862</td>
<td>72</td>
</tr>
<tr>
<td>9743</td>
<td>59</td>
</tr>
</tbody>
</table>
Grade processing example

- Write a program to
  - Input student ID and grades on each question one student at a time.
  - Check that one grade is 0
  - Compute and store each student’s total.
  - Compute and store the average for each question (omit 0s, since these were not attempted).
  - Print results, studentID, mark per Q, total, and then the averages.

Need for 2D arrays

- Note that there is nothing in this problem that requires the use of a 2D array,
  - unless input and output are to and from keyboard/screen
  - Generally don’t over complicate things
  - Don’t use a 2D array unless you need one.

Lab exercise: Game of Life

- Conway’s Game of Life simulates “life” on a 2D grid.
- 4 basic rules
- Immense complexity possible.