Arrays (2)

Multidimensional arrays
By the end of this lesson you will be able to:

- Understand and use multidimensional arrays, including:
  - Two+ dimensional arrays of basic data types
  - Arrays of strings
  - Command line arguments
Two-dimensional arrays

- Many problems are well served by representation in two dimensions (e.g. matrices, game boards, etc.)
- Usage of two-dimensional arrays is fairly common and a natural extension of one-dimensional arrays
- You can create multidimensional arrays as follows:

  ```
  datatype identifier[dim1_value][dim2_value];
  ```
Two-dimensional arrays

- **Example:**
  ```java
  int ary[3][2];
  ```

- **Accessible elements are** `ary[0][0]`, `ary[0][1]`, `ary[1][0]`, `ary[1][1]`, `ary[2][0]`, and `ary[2][1]`
  - 6 total elements (3 x 2 = 6)

- **Common to process using nested loops:**
  ```java
  for (int i = 0; i < 3; i++)    // 1st dimension
    for (int j = 0; j < 2; j++)  // 2nd dimension
      // something with `a[i][j]` here
  ```
Two-dimensional arrays

- Initialization:
  ```c
  int a[3][2] = { { 1, 2, 3 }, { 4, 5, 6 } };
  ```

- When passing to functions they are still automatically passed by reference, however in both the prototype and the implementation you must give the second dimension:
  ```c
  void fcn(int [[]][2]); // prototype
  void fcn(int a[][2]) // implementation
  {
    // some code here
  }
  ```

- Call function as such: `fcn(a);` (no change from one-dimensional calling)
Multidimensional arrays

- Rarely, you may want to use more than two dimensions in an array, for example keeping track of a matrix over time

- Example:
  - Declaration: `int matrix[5][10][100];`
  - Access: `matrix[2][4][15];`
  - Process: 3 nested `for` loops (`i: 0 to 4, j: 0 to 9, k: 0 to 100`)
  - Function usage: `void fcn(int [[5][100]];`
Arrays of strings

• Behind the scenes, a string basically contains an array of characters

• Things to note:
  – Passing a one-dimensional array of strings to a function is just like passing any one-dimensional array (no need to put the size in the prototype or first line of implementation)
  – Accessing a string in a one-dimensional array of strings is just like any other one-dimensional array: `string a[5];  a[2] = "Hi!";`
  – Accessing a character in a one-dimensional array of strings is like using a two-dimensional array:
    `cout << a[2][3] << endl; // will output !`
Command line arguments

- We have usually provided data to our programs by hardcoding filenames and by requesting input (a filename, or numerical data)
- An alternate method is to provide data to the program at the same time it is executed via command line arguments
- Values typed after `. /a.out` will enter the program as a two-dimensional array of characters and there is a unique syntax for obtaining these
Command line arguments

- **Main body declaration:**
  ```
  int main(int argc, char **argv)
  OR
  int main(int argc, char *argv[])
  ```
- **argc** is a count of how many values were entered, including `.a.out`
- **argv[0]** through **argv[argc - 1]** is each value
  - Values are broken up separated by spaces
- You can use any valid identifier, but convention says we use **argc** and **argv**
Command line arguments

- Example:
  - ./a.out data1
    - argv[0] contains "./a.out"
    - argv[1] contains "data1"
  - ./a.out Sam 55
    - argv[0] contains "./a.out"
    - argv[1] contains "Sam"
    - argv[2] contains "55"

- Numerical values need to be converted? Use <cstdlib> header and the atoi function (e.g. atoi(argv[2])) in the last example to get the integral value 55. Similarly, atof will convert to double (floating-point)
Exploration

• Write a program that declares a two-dimensional array of size 2 by 4 to represent a matrix. Initialize the values in the array at declaration, pass it to a function that will double each value, then pass it to a function that will output the matrix in a nicely formatted way.

• A file contains no more than 100 numbers. At the command line provide both the file name as well as an integer. Open the file, read all values in, then output the sum of those values multiplied by the integral command line argument you provided.