(9-1) Arrays IV – Parallel and H&K Chapter 7

Instructor – Beiyu Lin

CptS 121 (May 29th, 2019)

Washington State University

Parallel Arrays (1)

- What is Paralle Array:
- The components of those arrays have related information.

- E.g. int student_id[50];
  double gpa[50];

ID	GPA
111111111	4.0
111111112	3.9
111111113	3.9
111111114	3.8

Parallel Arrays (1)

- Often, we'd like to associate the values in one array with those in another array
  - A list of student numbers, together with their class standings, for example
- We can declare *parallel arrays* to accomplish this:

```
#define NUM_STUDENTS 100
typedef enum {freshman, sophomore, junior, senior}class_t;
int id[NUM_STUDENTS];
class_t class[NUM_STUDENTS];
```


Parallel Arrays (2)

- The parallel arrays of student numbers and class standings might look something like this:

id[0]	id[1]	id[2]	id[3]	id[4]	id[5]
1230	3124	6534	7842	3523	9785

class[0]	class[1]	class[2]	class[3]	class[4]	class[5]
freshman	senior	junior	sopho-more	junior	freshman

Multidimensional Arrays (1)

- Thus far, we've focused on *single dimensional* arrays
  - We declare them as follows:
    ```
    int my_array[6];
    ```
  - And we visualize them as follows:

10	12	0	89	0	91

  - Essentially, they are a single row of values
- Many applications, however, call for not just a single row, but a two-dimensional matrix of values
  - Examples: A tic-tac-toe board, A table of financial data, a grid of train connections
  - We can use *two-dimensional arrays* to represent such objects

Graph is from: https://saudi.souq.com/sa-en/foldable-chess-board-12-inch-9971928/i/

Multidimensional Arrays (2)

- Declaring a multidimensional array
  - The following code declares a 3 × 3 array that could be used to represent a tic-tac-toe board:

```
char tic_tac_toe_board[3][3];
```

  - We'll represent the three possible values on a board as characters: 'B' = blank, 'X' = x player, and 'O' = o player.
  - A sample board:

		Column 0	Column 1	Column 2
Row	0	X	B	O
	1	B	O	X
	2	X	B	B

Multidimensional Arrays (3)

- Referencing array cells
  - We use double bracket notation to reference a cell
  - For example, assuming the previous `board`:

Row \ Column	0	1	2
0	X	B	O
1	B	O	X
2	X	B	B

The following are true:

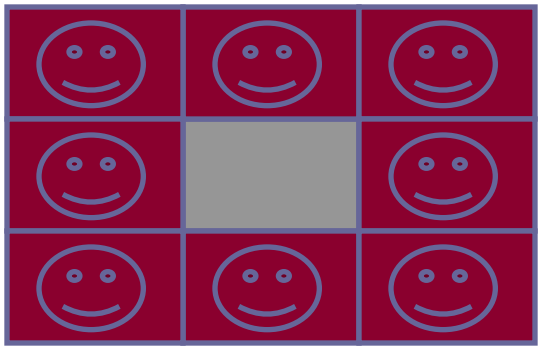
```
board[0][0] == 'X'          board[0][1] == 'B'          board[0][2] == 'O'
board[1][0] == 'B'          board[1][1] == 'O'          board[1][2] == 'X'
board[2][0] == 'X'          board[2][1] == 'B'          board[2][2] == 'B'
```

Multidimensional Arrays (4)

- Example 2D Array Application: Implementation of “Game of Life”
  - “World” is an n x n grid
  - The “game” is to determine who lives and who dies from generation to generation
  - A "live" cell will be alive in the next generation if and only if 2 or 3 of its 8 “neighbors” are living
  - A "dead" cell will come to life in the next generation if and only if exactly 2 of its 8 neighbors are living
  - See http://www.math.com/students/wonders/life/life.html for the interesting history of this application, and a really cool applet

Multidimensional Arrays (5)

- Here are the neighbors of a given cell:



- Problem statement:

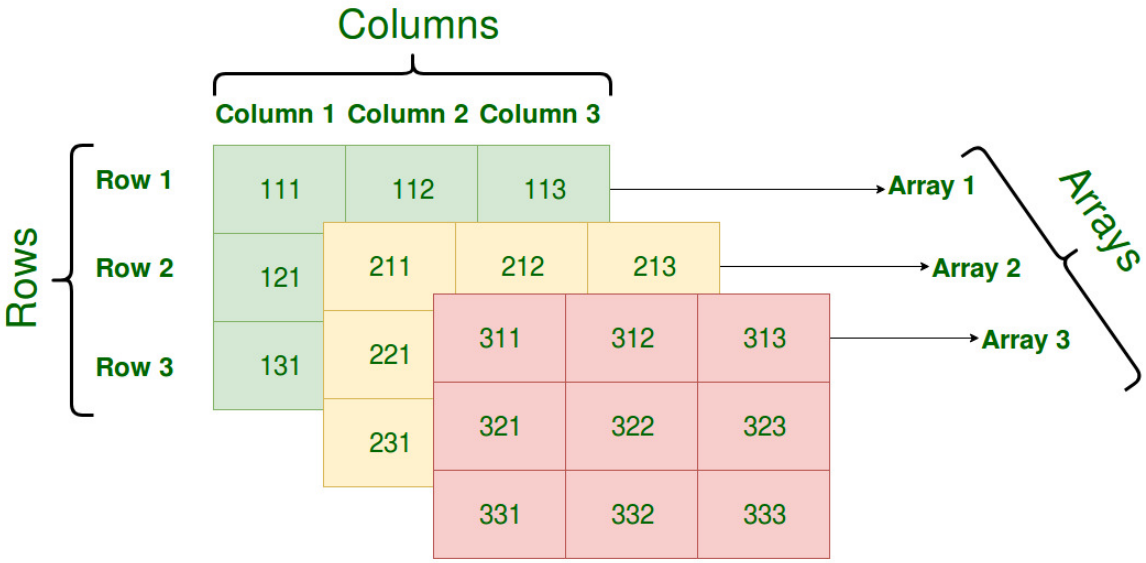
  Implement an application that models the Game of Life by choosing an initial distribution of life and then carrying out the game from one generation to the next. Display the generations as they are produced

Multidimensional Arrays (6)

- Top-down design:
  - Key data structure:
    - `world` – a 2D array of ints (0 == dead, 1 == alive)
  - Functions
    - `get_life_probability` – prompts the user for the probability of life in a given square, and returns that value (an int between 0 and 100)
    - `initialize_world` – Given the probability of life in a given square, randomly generates the world by filling the world array with 1's and 0's
    - `get_next_generation` – Generates the next generation of life by applying the rules outlined on the previous slide.
    - `life_is_instinct` – determines whether a world is completely devoid of life
    - `living_neighbors` – Given a cell in the world, determines how many of the cell's neighbors are alive
    - `is_alive` – Given a cell in the world, determines whether the cell is alive. Gracefully handles the boundary cases (e.g., a neighbor of cell 0,0 is -1,-1)
    - `display_world` – pretty-prints the world to the screen

Multidimensional Arrays (7)

- Note: Higher dimensional arrays (e.g., 3D) are possible. However, we won't cover them in this class
- Examples:



Graph is from: https://www.geeksforgeeks.org/multidimensional-arrays-in-php/

References

- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (8th Ed.)*, Addison-Wesley, 2016
- P.J. Deitel & H.M. Deitel, *C How to Program (7th Ed.)*, Pearson Education , Inc., 2013.

Collaborators

- Chris Hundhausen
- Andrew O'Fallon