Multi-dimensional Arrays

- The elements of an array can be any type
  - Including an array type
- So
  - int 2D[][]; declares an array of arrays of int
- Two dimensional arrays are useful for representing tables of data with rows and columns
- Three dimensional arrays are useful for representing volume data with three coordinates x, y, and z

Two-Dimensional Arrays

- A 2 dimensional array is declared and allocated like this:
  - int a[][] = new int[3][4];
  
  - a is an array of 3 arrays of int
  - a[0] is an array of 4 ints
  - a[1] is an array of 4 ints
  - a[2] is an array of 4 ints

Two-Dimensional Array Indexing

- Array elements are access by providing a value for all subscripts.

- The first subscript represents the row
- The second subscript represents the column

Two-Dimensional Array Example

// Division Table
class Divide {
    public static void main(String[] args) {
        double[][] data = new double[10][5];
        for(int i = 0; i < data.length; i++) {
            for(int j = 0; j < data[i].length; j++) {
                data[i][j] = (double)i / j;
            }
        }
        for(int i = 0; i < data.length; i++) {
            for(int j = 0; j < data[i].length; j++) {
                System.out.print(data[i][j] + " ");
            }
            System.out.println();
        }
    }
}
Two-Dimensional Array Initializer

- Remember, we can initialize a single dimensional array
  ```java
  int temperature[] = {32, 35, 38, 36, 38, 35, 40};
  ```

- We can also initialize two-dimensional arrays
  ```java
  int temperature[][] = {{32, 58}, {35, 60},
                         {38, 59}, {36, 63}, {38, 61},
                         {35, 65}, {40, 68}};
  ```

Irregular Arrays

- 2-dimensional arrays don't have to be 'rectangular'
  ```java
  int a[][] = {{2, 5}, {3, 4, 0}, {3}, {4, 5}};
  a[0] array of 2 ints
  a[1] array of 3 ints
  a[2] array of 1 int
  a[3] array of 2 ints
  ```

Conway's Game of Life

- This game was originally developed to simulate certain kinds of growth
- It is "played" on a rectangular grid, like a checkerboard
- Each cell of the board is either alive or dead
- Live cells are marked with an 'X'

Rules for the Game of Life

- Each cell is either empty or alive
- Each cell is at the center of a 3x3 grid of cells, which contains the cell's 8 neighbors
- A cell that is empty at time $t$ becomes alive at time $t+1$ if and only if exactly 3 of its neighbors were alive at time $t$.
- A cell that is alive at time $t$ remains alive at time $t+1$ if and only if 2 or 3 of its neighbors were alive at time $t$. Otherwise, it dies for lack of company (<2) or overcrowding (>3)
- The simulation is conducted, in principle, on an infinite grid.
Requirements

• Use a finite grid instead of an infinite one
• Treat borders as lifeless zones
• User will specify grid size, which will always be square
• The user will specify the number of steps to simulate
• Get the initial configuration by reading *'s and dots from the console
• The program will echo the initial configuration and then display each subsequent configuration.

Top Level PsuedoCode

Read in the size of the grid
Read in the initial generation
Read in the number of generations to simulate
Print the current generation
For the specified number of generations
  Advance one generation
  Print the generation
end for loop

Advance One Generation

For each cell
  compute the number of neighbors
  apply the game rules to decide if the cell should be alive or dead in the next generation
end for loop

Compute the number of neighbors

visit each cell in the 3x3 subarray around the cell
keep a count of how many of those are alive
if the center cell is alive
  then return 1 less than the count found
else return the count found
Refined Top Level PsuedoCode

Read in the size of the grid
Create 2 arrays, currentGen and nextGen
Read in the initial generation into currentGen
Read in the number of generations to simulate
Print the current generation
For the specified number of generations
   Update nextGen using currentGen
   Swap nextGen and currentGen
   Print currentGen
end for loop

Refined Advance One Generation

input: currentGen and nextGen
output: updated nextGen

For each cell in currentGen
   compute the number of neighbors
   apply the game rules to decide if the cell should be alive or dead in the next generation
   store the result, alive or dead, in the corresponding cell in nextGen
end for loop

GameOfLife: main part 1

//GameOfLife.java - Conway's Game of Life
import tio.*;
class GameOfLife {
   public static void main(String[] args) {
      // read in the size of the grid and create arrays
      int size = 10; // fix at 10 for testing
      boolean[][] currentGeneration, nextGeneration;
      currentGeneration = new boolean[size][size];
      nextGeneration = new boolean[size][size];
      readInitialGeneration(currentGeneration);

      // read in the number of generations to simulate
      int cycles = 4; // fix at 4 for testing
      printState(currentGeneration);
      for (int i = 0; i < cycles; i++) {
         System.out.println("Cycle = " + i + "\n");
         advanceOneGen(currentGeneration,
                        nextGeneration);
         printState(nextGeneration);
         // swap current and next generations
         boolean[][] temp = nextGeneration;
         nextGeneration = currentGeneration;
         currentGeneration = temp;
      }
   }
}

GameOfLife: main part 2

// read in the size of the grid and create arrays
int size = 10; // fix at 10 for testing
boolean[][] currentGeneration, nextGeneration;
currentGeneration = new boolean[size][size];
readInitialGeneration(currentGeneration);

// read in the number of generations to simulate
int cycles = 4; // fix at 4 for testing
printState(currentGeneration);
for (int i = 0; i < cycles; i++) {
   System.out.println("Cycle = " + i + "\n");
   advanceOneGen(currentGeneration,
                  nextGeneration);
   printState(nextGeneration);
   // swap current and next generations
   boolean[][] temp = nextGeneration;
   nextGeneration = currentGeneration;
   currentGeneration = temp;
}
readInitialGeneration()

// read the initial generation from the input
// a dot means empty and a * means alive
// any other characters are ignored
// the border cells are all set to empty
// the method assumes the array is square
static void readInitialGeneration(boolean[][] w) {
    for (int i = 0; i < w.length; i++)
        for (int j = 0; j < w[i].length; j++) {
            char c = (char)Console.in.readChar();
            //skip illegal characters
            while (c != '.' && c != '*')
                c = (char)Console.in.readChar();
            if (c == '.')
                w[i][j] = EMPTY;
            else
                w[i][j] = ALIVE;
        }
}

readInitialGeneration() part 2

// set border cells to be empty
static void printState(boolean[][] w) {
    for (int i = 0; i < w.length; i++)
        System.out.println();
        for (int j = 0; j < w[i].length; j++)
            if (w[i][j] == ALIVE)
                System.out.print('X');
            else
                System.out.print('.');
        System.out.println();
    }

neighbors()

// compute the number of alive neighbors of a cell
static int neighbors(int row, int column, boolean[][] w) {
    int neighborCount = 0;
    for (int i = -1; i <= 1; i++)
        for (int j = -1; j <= 1; j++)
            if (w[row + i][column + j] == ALIVE)
                neighborCount = neighborCount + 1;
    return neighborCount;
}
**advanceOneGen()**

```java
static void advanceOneGen(boolean[][] wOld, boolean[][] wNew) {
    int neighborCount;
    for (int i = 1; i < wOld.length - 1; i++)
        for (int j = 1; j < wOld[i].length - 1; j++) {
            neighborCount = neighbors(i, j, wOld);
            if (neighborCount == 3)
                wNew[i][j] = ALIVE;
            else if (wOld[i][j] == ALIVE &&
                neighborCount == 2)
                wNew[i][j] = ALIVE;
            else
                wNew[i][j] = EMPTY;
        }
}
```

**Cool Web Site**

- Good description of the Game of Life
- Neat Java applet that runs simulations with different starting patterns

**Distributed Pair Programming**

- Pair Programming has been shown to provide many benefits
- However, there is at least one major drawback
  - You have to be collocated
- I am developing a tool to support *distributed* pair programming
  - Supports pair programming from 2 separate locations

**Seeking Volunteers**

- I am looking for some students who would like to try to use this tool
  - You can use it on your homework assignments
- What do I want?
  - Your honest feedback regarding your experience using the tool
    - Did it work?
    - How did distributed pairing compare to collocated pairing?
    - What would you do to improve the tool?
Requirements

• You must have:
  – A PC running Windows
  – A sound card
  – A high speed Internet connection (DSL, campus network, cable modem)
    • Dial-up is probably not sufficient, although I don’t know for sure
• You also need to have or set up:
  – SSH (secure shell)
  – AOL Instant Messenger

Quiz 4

• Covers topics in chapter 5
• Single-dimensional and two-dimensional arrays
  – Array length
  – Array indexing
• Defining array references, allocating the array, and array initialization
  – operator new
  – default initialization
  – initializer lists

Quiz 4

• Array assignment
• Passing arrays as parameters to methods
  – Distinction between assignments to the array reference and array elements
• Common Errors with arrays
  – indexing from 1 to length
  – forgetting to allocate the array
• Big O notation
  – what it means
  – which orders are more efficient

Quiz 4: Examples

• Write a single statement that declares an array of String, allocates the array, and assigns the array elements the values "one", "two", and "three".
• Write a single statement that declares and allocates a two dimensional array of int, with 10 rows and 5 columns.
Quiz 4: Examples

• Assume that you have a choice of two algorithms that you can use to sort the elements in an array. One of the algorithms performs in $O(n \log n)$ time, while the other performs in $O(n^2)$ time. In general, which algorithm is more efficient?

Quiz 4: Examples

• Write a method `arraySum()` that takes an array of `int` as a parameter and returns the sum of the array elements to the caller. Do not write a program – write a method only.

Quiz 4: Examples

• Assume `a` has the following definition:
  ```java
  int a[][] = {{1,2,3}, {4,5}, {6,7,8,9}};
  ```
  What is the value of the following expressions?
  - `a.length`
  - `a[1].length`
  - `a[1]`
  - `a[1][1]`
  - `a[1][2]`
  - `a[2][0]`

Quiz 4: Examples

• What is printed by the following Java fragment?
  ```java
  int firstArray[] = {1,2,3};
  int secondArray[] = {5,10,15,20,25};
  firstArray = secondArray;
  secondArray[0] = 15;
  System.out.println( firstArray[0] );
  ```
Quiz 4: Examples

• What is printed by the following Java program?

```java
class Q4Example1 {
    public static void main( String[] args ) {
        int a[] = {1,2};
        int b[] = {3,4};
        mystery(a, b);
        System.out.println( "a[0] = " + a[0] );
        System.out.println( "b[0] = " + b[0] );
    }
}

static void mystery( int[] s, int[]t ) {
    s = t;
    System.out.println("s[0] = " + s[0] );
    System.out.println("t[0] = " + t[0] );
}
```

Quiz 4: Examples

• The following Java program contains an error. Correct it.

```java
class Q4Example2 {
    public static void main( String[] args ) {
        int a[] = {1,2,3,4};
        mystery( a );
    }
}

static void mystery( int[] a ) {
    for (int i = 0; i <= a.length; i++ )
        System.out.println(a[i] + " ");
}
```

Quiz 4: Examples

• What is printed by the following program?

```java
class Q4Example3 {
    public static void main( String[] args ) {
        String names[][] = { {"Brian", "Dave"}, {"Amy", "Karen"}, {"Sanjit", "Anne", "William"} };
        mystery( names );
    }
}

static void mystery( String[][] s ) {
    for (int i = 0; i < s.length; i++ ) {
        for (int j = 0; j < s[i].length; j++)
            System.out.print( s[i][j] + " " );
        System.out.println();
    }
}
```

Quiz 4: Examples

• What is printed by the following program?

```java
class Q4Example4 {
    public static void main( String[] args ) {
        int values[] = {10, -5, 3, -2, -1, 12, -7};
        mystery( values );
        for ( int i = 0; i < values.length; i++ )
            System.out.print( values[i] + " ");
    }
}

static void mystery( int[] a ) {
    for (int i = 0; i < a.length; i++ ) {
        if ( a[i] < 0 )
            a[i] = -a[i];
    }
}
```