**Topic 3: Two-Dimensional Arrays and Jagged Arrays**

Top of Form

Bottom of Form

**Content**

**Reading**

[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

2d Arrays

* + Microsoft:docs.microsoft.com
    - [Multidimensional Arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays)
  + GeeksForGeeks: geeksforgeeks.com
    - [C# Multidimensional Indexer](https://www.geeksforgeeks.org/c-sharp-multidimensional-indexers/)
  + TutorialsPoint: tutorialspoint.com
    - [C# - Multidimenstional Arrays](https://www.tutorialspoint.com/csharp/pdf/csharp_multi_dimensional_arrays.pdf)

Jagged Arrays

* + Microsoft:docs.microsoft.com
    - [Jagged Arrays](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/jagged-arrays)
  + GeeksForGeeks: geeksforgeeks.com
    - [C# Jagged Arrays](https://www.geeksforgeeks.org/c-sharp-jagged-arrays/)
  + TutorialsPoint: tutorialspoint.com
    - [C#- Jagged Arrays](https://www.tutorialspoint.com/csharp/csharp_jagged_arrays.htm)

**2D Array Basics**

[2D Array Basics](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

https://youtu.be/EiNaSsSFigo

using System;

namespace Module6

{

class Program

{

static void Main(string[] args)

{

const int DAYS = 5;

const int HOURLY\_RATES = 3;

// Prices per hour

int[,] rates = new int[DAYS, HOURLY\_RATES] {{ 10, 10, 10 },

{ 10, 10, 15 },

{ 10, 15, 20 },

{ 10, 15, 20 },

{ 15, 20, 25 }};

rates[0, 0] = 5;

// print array

Console.WriteLine("\nPrinting the 2D array:");

for (int i = 0; i < DAYS; i++)

{

for (int j = 0; j <= HOURLY\_RATES; j++)

{

Console.Write(rates[i, j] + "\t");

}

Console.WriteLine();

}

}

}

}

**Declare, Instantiate and Initialize Two-Dimenstional Arrays**

[Declare, Instantiate and Initialize Two-Dimenstional Arrays](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

To declare a two-dimensional (2d) array, you need a number of rows and number of columns.

Below is an example 2d array with three rows and four columns:

double[ , ] scores = new double[3, 4];

Below is an example 2d array with five rows and ten columns, using constants for declaration and assigning Row 2, column 1 to 95.

const int ROWS = 5

const int COLS = 10;

int[ , ] values = new int[ROWS, COLS];

values[2,1] = 95;

Recall arrays begin at zero, the same is true for 2d arrays, (0,0).

Below the example 2d array with three rows and three columns, using constants for declaration and assigning row 0, column 0 to 1.

const int ROWS = 3;

const int COLS = 3;

int[,] values = new int[ROWS, COLS];

values[0,0] = 1; ​

**Two-Dimensional Array as a Table**

[Two-Dimensional Array as a Table](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

Note in the example of a 2D array below, you still want to use constants for sizes!

const int ROWS = 5;

const int COLS = 3;

string[,] petSitting = new string[ROWS, COLS];

You can think of the 2D array as table, as diplayed below:

petSitting:

|  |  |  |
| --- | --- | --- |
| Leo | Luna | Tex |
| Dani | Angel | Finn |
| Bowie | Tex | Leo |
| Bear | Monkey | Lexie |
| Bowie | Tex | Leo |

To access the 2D array called petSitting:

<tdpetSitting[2,2]

|  |  |  |
| --- | --- | --- |
| petSitting[0,0] | petSitting[0,1] | petSitting[0,2] |
| petSitting[1,0] | petSitting[1,1] | petSitting[1,2] |
| petSitting[2,0] | petSitting[2,1] | petSitting[2,2] |
| petSitting[3,0] | petSitting[3,1] | petSitting[3,2] |
| petSitting[4,0] | petSitting[4,1] | petSitting[4,2] |

Even though you do not have the rest of the code, you can guess is this likely a dog sitting schedule. The rows are likely Days of the week and the columns are likely times. This is an example of self-documenting code. Using meaningful and descriptive variables names can tell some of the story of the code.

**Jagged Arrays Basics**

[Jagged Arrays Basics](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

https://youtu.be/FMEjXZbSbAQ

const int NUMBER\_OF\_STUDENTS = 5;

// Declare and Initialize Jagged Arrays

int[][] scores = new int[NUMBER\_OF\_STUDENTS][];

scores[0] = new int[3] { 23, 43, 54 };

scores[1] = new int[4] { 53, 32, 21, 33 };

scores[2] = new int[4] { 42, 61, 44, 88 };

scores[3] = new int[3] { 450, 21, 32 };

scores[4] = new int[5] { 45, 56, 34, 23, 34 };

// Get user input for scores

// using nested loops to print array

Console.WriteLine("\nPrinting the jagged arrays:");

for (int i = 0; i < NUMBER\_OF\_STUDENTS; i++)

{

Console.Write("Student #{0}:\t", i + 1); // array start 0

// The current row is an array of any size, use Length property

for (int j = 0; j < scores[i].Length; j++)

{

Console.Write(scores[i][j] + "\t");

}

Console.WriteLine();

}

https://youtu.be/bvHvTKTddsY

// using nested loops to print array

Console.WriteLine("\nPrinting the jagged arrays:");

int index = 0;

foreach (int[] scoreRow in scores)

{

Console.Write("Student #{0}:\t", ++index); // increment before printing

// The current row is an array of any size, use Length property

for (int j = 0; j < scoreRow.Length; j++)

{

Console.Write(scoreRow[j] + "\t");

}

Console.WriteLine();

}

**Initialize Jagged Arrays from User Input**

[Initialize Jagged Arrays from User Input](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

https://youtu.be/g6CH3qI8ars

[Program.cs](https://dmacc.blackboard.com/bbcswebdav/pid-7286867-dt-content-rid-101466158_1/xid-101466158_1)

**Jagged Arrays vs. 2D arrays**

[Jagged Arrays vs. 2D arrays](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286838_1)

A two-dimensional array where the rows in can have different lengths is called a jagged array.

The array is declared the way, but instantiated with only the first index size (row size). (line 2)

Then each row is instantiated separately (lines 6, 7, 8). Note they are also initialized in the same statement.

// Create an array of 3 int arrays.

int[ ][ ] jaggedArray = new int[3][ ];

// Create each array that is an element of the jagged array.

jaggedArray[0] = new int[4] { 1, 2, 3, 4 };

jaggedArray[1] = new int[3] { 5, 6, 7 };

jaggedArray[2] = new int[5] { 8, 9, 10, 11, 12 };

Jagged arrays are not available in all programming languages. There are few advantages. In languages without jagged arrays, you can use 2-d arrays and have empty, null entries or even repeated values. This takes the up unnecessary storage memory and is less efficient. For example, in a mileage table, you only need to store one entry, instead of repeated (and memory wasting) entries.

For 2-d Array:

Graphical user interface, table

Description automatically generated

Can store in Jagged Array:

Table

Description automatically generated

[**Max and Min in Jagged Arrays**](https://dmacc.blackboard.com/webapps/assignment/uploadAssignment?content_id=_7286871_1&course_id=_102593_1&group_id=&mode=view)

Congrats, you landed a job that allows you to live in Manchester! You are ready to plan a mini-holiday getaway and a longer holiday. You have free rooms from your hotel rewards program, but only at certain destintations and certain times. (It was in the fine print). You want to write a program to find the closest available city with a hotel so that you can maximize your time-off with minimal travel time. You also want to find the fartherest available city for a longer stay. The list of destinations for your holidays are below in the mileage table.

Table

Description automatically generated

In a file named JaggedArrays.cs

* + Write a method Min() that finds and returns the minimum value of integer array (do not use Array class methods)
    - Argument list is an array of type integer, pass by value
  + Write a method Max() that finds and returns the maximum value of integer array (do not use Array class methods)
    - Argument list is an array of type integer, pass by value
  + Main()
    - To make sense of the table, assign Edinburgh to 0, Birmingham to 1, etc.
    - string[] cityList = { "Edinburgh", "Birmingham", "Cardiff",
    - "Dover", "Leeds", "Liverpool", "London", "Manchester",

"NewCastle", "York" };

* + - Declare a jagged array from the mileage table, called mileageTable
      * Why type is your array?
      * What is the size of your jagged array?
      * The first jagged array at index 0 will have the entry 0. It represents distance from Edinburgh to Edingburgh. It's not listed in table
      * The second jagged array at index 1 will have entry of 290, It represents distance from Edinburgh to Birmingham.
      * Create the array for each, with index 2 representing Cardiff, index 3 Dover, etc...
      * Notice each jagged array increases in size by one.
    - Print the table using a nested for loop.
      * This can be tricky. Reach out on the discussion board or via email if you need an extra hint.
      * Below a "\t" is used for spacing.
      * 0
      * 290
      * 373 102
      * 496 185 228
      * 193 110 208 257
      * 214 90 165 270 73
      * 412 118 150 81 191 198
      * 222 86 173 285 41 34 201
      * 112 2017 301 360 94 155 288 141

186 129 231 264 25 97 194 66 82

* + - Now you can use the table to determine closest mileage to Manchester
      * You will not need to send the entire jagged array in the calls to Min() and Max(), you will need to select the array representing Manchester
    - Call Min() and print the closest mileage (no need to print the city, the table is printed for reference)

The closest is 34 miles

* + - Call Max() and print the farthest mileage (no need to print the city, the table is printed for reference)

The farthest is 285 miles

* + - Extra Credit: Print the name of the city from Min() and Max()
    - The closest is 34 to Liverpool

The farthest is 285 to Dover

Submit your JaggedArrays.cs file with the Academic Honesty Header included. Be sure to use good naming conventions and variables instead of values.

This is 20 points.