Contents

Two-Dimensional Arrays (Solutions)

Multiple Choices

1. What is the correct way of creating a 2-dimensional rectangular array of `int` with 5 rows and 2 columns named `myMatrix`?

   ☐ `int[][] myMatrix = new int[5][2];`
   ☐ `int[][] myMatrix = new int[2][5];`
   ☐ `int[,] myMatrix = new int[2, 5];`
   ☒ `int[,] myMatrix = new int[5, 2];`

2. Consider the following code:

   ```
   int[,] grades = {{10, 20}, {30, 40}};
   Console.WriteLine(grades[1,0]);
   ```

   What will it display?

   ☐ Nothing
   ☐ 10
   ☐ 20
   ☐ grades
   ☒ 30
   ☐ grades(1,0)
   ☐ 40

Exercises

1. Write a statement that creates a 2-dimensional rectangular array of `int` with 5 rows and 3 columns.

   Solution

   ```
   int[,] matrix = new int[5, 3];
   ```

2. Write a statement that creates a 2-dimensional jagged array of `int` with 2 rows. The first row should contain an array containing 1, the second row should contain an array containing 2, 3.

   Solution

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

3. Write a declaration for a 2-dimensional rectangular array of `int` containing the following data:

10	20	30
40	50	60
70	80	90

Solution

```
int[,] matrix =
    {
        {10, 20, 30},
        {40, 50, 60},
        {70, 80, 90}
    };
```

4. Write a declaration for a 2-dimensional ragged array of `int` containing the following data:

10	20	
40		
70	80	90

Solution

```
int[][] data = new int[3][];
data[0] = new int[3] { 10, 20, 30 };
data[1] = new int[1] { 40 };
data[2] = new int[3] { 70, 80, 90 };
```

5. Suppose we have a 2-dimensional rectangular array named `temp` that has been declared and initialized. How can we know the number of rows in this array?

Solution

By using the `GetLength` method: `temp.GetLength(0)` is the number of rows in the `temp` array.

6. Suppose we have a 2-dimensional rectangular array named `temp` that has been declared and initialized. How can we know the number of elements in this array?

   Solution

   By using the `Length` field: `temp.Length` is the number of elements in the `temp` array. We can also compute the product of `temp.GetLength(0)` and `temp.GetLength(1)`.

7. Write a `Display` static method that takes as an argument a 2-dimensional array and displays it at the screen.

   Solution

```
static void Display(int[,] matP)
{
  for (int row = 0; row < matP.GetLength(0); row++)
  {
    for (int col = 0; col < matP.GetLength(1);
     ↪  col++)
    {
      Console.Write(
        String.Format("|{0,4} ", matP[row, col])
      );
    }
    Console.WriteLine(" |");
  }
}
```

8. Write a program that display "Every row contains its own index" if the 2-dimensional rectangular array of `int` `matrix` is such that its first row contains the value 0, its second row contains the value 1, etc.

   Solution

```
bool foundIndex = true;
for (int row = 0; row < matrix.GetLength(0); row++)
{
    if(row == 0 || foundIndex)
    {
        foundIndex = false;
        for (int col = 0; col < matrix.GetLength(1);
         ↪  col++)
        {
            if (matrix[row, col] == row) {
                foundIndex = true;
            }
        }
```

```
        }
    }

    if(foundIndex){
        Console.WriteLine("Every row contains its own
    ↪  index");
    }
```

9. Write a program that display the average of each row of a 2-dimensional jagged array of `int jArray`.

   Solution

```
double sum;
for(int i = 0; i < jArray.Length; i++)
{
    sum = 0;
    for (int j = 0; j < jArray[i].Length; j++)
    {
        sum += jArray[i][j];
    }
    Console.WriteLine("Average for row #" + i
        + " is " + sum / jArray[i].Length);
}
```

10. Write a program that display the sum of the values on the diagonal of a 2-dimensional rectangular array of `int jArray`.

    Solution

```
int sum = 0;
for (int i = 0; i < matrix.GetLength(0); i++)
{
    sum += matrix[i, i];
}
Console.WriteLine(sum);
```

11. Write a program that "rotate" a 2-dimensional array 90° clockwise. For example, the array

```
int[,] matrix =
{
  { 1, 2, 3 },
  { 4, 5, 6 },
  { 7, 8, 9 },
  { 10, 11, 12 },
};
```

    would become

```
|  10 |   7 |   4 |   1  |
|  11 |   8 |   5 |   2  |
|  12 |   9 |   6 |   3  |
```

Solution

```
static void Rotate(ref int[,] matP)
{
  int[,] tmp = new int[
    matP.GetLength(1),
    matP.GetLength(0)
  ];
  for (int row = 0; row < tmp.GetLength(0); row++)
  {
    for (int col = 0; col < tmp.GetLength(1); col++)
    {
      tmp[row, col] = matP[
        tmp.GetLength(1) - col - 1,
        row
      ];
    }
  }
  matP = tmp;
}
```

Wrapping-up Problem

1. Answer the following questions using a 2-dimensional jagged array **and** a 2-dimensional rectangular array.

   (a) Write statements that declare a 2-dimensional array with at least 2 rows containing values 1 through 6.

   (b) Write statements that display the values stored in a 2-dimensional array called `arTest`.

   (c) Write statements displaying "Ranked" if the sum of the values stored in each row is greater than the sum of the values in the preceding row in the `arTest` array. For example, the following rectangular and jagged arrays are both ranked, as 4 < 5 < 6.

   ```
   2 1 1
   3 1 1
   5 1 0
   ```

   ```
   4 0
   3 1 1
   2 2 1 1
   ```

Solution

```
using System;

class Program
{
  static void Main(string[] args)
  {
    // Rectangular Arrays
    int[,] testR1 =
    {
      { 1, 2, 3 },
      { 4, 5, 6 },
    };
    int[,] testR2 =
    {
      { 1, 2, 3 },
      { 4, 5, 6 },
      { 0, 2, 10 },
    };

    // Code for DisplayR and RankedR is below, in
     ↪  their own methods.
    DisplayR(testR1);
    DisplayR(testR2);
    Console.WriteLine("Ranked:" + RankedR(testR2));

    // Jagged Arrays
    int[][] testJ1 = new int[2][];
    testJ1[0] = new int[4] { 1, 2, 3, 4 };
    testJ1[1] = new int[2] { 5, 6 };

    int[][] testJ2 = new int[3][];
    testJ2[0] = new int[4] { 1, 2, 3, 4 };
    testJ2[1] = new int[2] { 5, 6 };
    testJ2[2] = new int[3] { 0, 2, 10 };

    // Code for DisplayJ and RankedJ is below, in
     ↪  their own methods.
    DisplayJ(testJ1);
    DisplayJ(testJ2);

    Console.WriteLine("Ranked:" + RankedJ(testJ2));
  }

  /* Rectangular arrays */
```

```
static void DisplayR(int[,] a)
{
  for (int row = 0; row < a.GetLength(0); row++)
  {
    for (int col = 0; col < a.GetLength(1); col++)
    {
      Console.Write(a[row, col] + " ");
    }
    Console.WriteLine();
  }
}

static bool RankedR(int[,] a)
{
  bool rsf = true;
  int countp = 0;
  int count = 0;
  for (int row = 0; row < a.GetLength(0); row++)
  {
    for (int col = 0; col < a.GetLength(1); col++)
    {
      count += a[row, col];
    }
    // Helper line:
    Console.WriteLine(
      $"Previous: {countp} Current: {count}"
    );
    if (row == 0)
    {
      countp = count;
    }
    else if (countp >= count)
      rsf = false;
    countp = count;
    count = 0;
  }
  return rsf;
}

/* Jagged Arrays */
static void DisplayJ(int[][] a)
{
  for (int row = 0; row < a.Length; row++)
  {
    for (int col = 0; col < a[row].Length; col++)
    {
```

```
        Console.Write(a[row][col] + " ");
      }
      Console.WriteLine();
    }
  }

  static bool RankedJ(int[][] a)
  {
    bool rsf = true;
    int countp = 0;
    int count = 0;
    for (int row = 0; row < a.Length; row++)
    {
      for (int col = 0; col < a[row].Length; col++)
      {
        count += a[row][col];
      }
      // Helper line:
      Console.WriteLine(
        $"Previous: {countp} Current: {count}"
      );
      if (row == 0)
      {
        countp = count;
      }
      else if (countp >= count)
        rsf = false;
      countp = count;
      count = 0;
    }
    return rsf;
  }
}
```

(Download this code)

Problem: Toward a Crossword Puzzle Solver

The goal of this problem is to work toward the creation of a program that solve crossword puzzles. We will reason in the simple case where the "word" is actually simply a pair of number (so, "1, 2" or "8, 101").

In the following, assume given two `int` variables `first` and `second`, as well as a 2-dimensional rectangular array `values`.

1. Write a program that display "pair found" if `first` and `second` occur next to each other in the same row.

2. Edit your program so that "pair found" is displayed also if `second` occurs before `first` in the same row.
3. Edit your program so that "pair found" is displayed also if `first` occurs "above" `second` (that is, if they are next to each other in the same column),
4. Edit your program so that "pair found" is displayed also if `second` occurs "above" `first`,
5. Edit your program so that "pair found" is displayed also if `first` and `second` occur diagonally,
6. Edit your program so that "pair found" is displayed also if `first` and `first` occur anti-diagonally.

Test your program thoroughly, possibly bundling it in a **`static`** class to ease testing and debugging.

Solution

A possible implementation, as a static class, is as follows:

```
using System; // required to use String.Format

public static class Crossword
{
  public static string Display(int[,] arrP)
  {
    string ret = "";
    for (int row = 0; row < arrP.GetLength(0); row++)
    {
      ret += "|";
      for (int col = 0; col < arrP.GetLength(1); col++)
      {
        ret += String.Format("{0,4}|", arrP[row, col]);
      }
      ret += "\n";
    }
    return ret;
  }

  public static bool Pair(
    int[,] arrP,
    int first,
    int second
  )
  {
    return PairRow(arrP, first, second)
      || PairRowInverse(arrP, first, second)
      || PairCol(arrP, first, second)
      || PairColInverse(arrP, first, second)
```

```
    || PairDiag(arrP, first, second)
    || PairDiagInverse(arrP, first, second);
}

public static bool PairRow(
  int[,] arrP,
  int first,
  int second
)
{
  bool foundPair = false;
  for (int row = 0; row < arrP.GetLength(0); row++)
  {
   for (int col = 0; col + 1 < arrP.GetLength(1); col++)
   {
     if (
       arrP[row, col] == first
       && arrP[row, col + 1] == second
     )
     {
       foundPair = true;
     }
   }
  }
  return foundPair;
}

public static bool PairRowInverse(
  int[,] arrP,
  int first,
  int second
)
{
  return PairRow(arrP, second, first);
}

public static bool PairCol(
  int[,] arrP,
  int first,
  int second
)
{
  bool foundPair = false;
  for (int row = 0; row + 1 < arrP.GetLength(0); row++)
  {
    for (int col = 0; col < arrP.GetLength(1); col++)
```

```
    {
      if (
        arrP[row, col] == first
        && arrP[row + 1, col] == second
      )
      {
        foundPair = true;
      }
    }
  }
  return foundPair;
}

public static bool PairColInverse(
  int[,] arrP,
  int first,
  int second
)
{
  return PairCol(arrP, second, first);
}

public static bool PairDiag(
  int[,] arrP,
  int first,
  int second
)
{
  bool foundPair = false;
  for (int row = 0; row + 1 < arrP.GetLength(0); row++)
  {
   for (int col = 0; col + 1 < arrP.GetLength(1); col++)
    {
      if (
        arrP[row, col] == first
        && arrP[row + 1, col + 1] == second
      )
      {
        foundPair = true;
      }
    }
  }
  return foundPair;
}

public static bool PairDiagInverse(
```

```
    int[,] arrP,
    int first,
    int second
  )
  {
    return PairDiag(arrP, second, first);
  }
}
```

(Download this code)