Contents

Manipulating Rectangular Arrays

We present below some simple algorithms to manipulate 2-dimensional (rectangular) arrays.

Summing the values row per row

The following code sum the values contained in a 2-dimensional array row per row, and display the result each time before moving on to the next row:

```
int[,] numbers =
    {
        {1, 2, 3, 4},
        {5, 6, 7, 8}
    };

int acc;
for (int row = 0; row < numbers.GetLength(0); row++)
    {
    acc = 0;
    for (int col = 0; col < numbers.GetLength(1); col++)
    {
        acc += numbers[row, col];
    }
    Console.WriteLine("Total for row #" + row +
        " is " + acc + ".");
}
```

This code can easily be adapted to compute the sums *column per column* if needed.

Computing Magic Square

A magic square[1] is a square matrix where the sums of the numbers in each row, each column, and both the diagonal and the anti-diagonal are the same.

[1] https://en.wikipedia.org/wiki/Magic_square

The following is an example of a magic square:

```
int[,] magicSquare = {
    { 4, 9, 2 },
    { 3, 5, 7 },
    { 8, 1, 6 }
};
```

as we have, diagonally,

$$4 + 5 + 6 = 15$$

and anti-diagonally,

$$2 + 5 + 8 = 15$$

and on the rows,

$$4 + 9 + 2 = 15$$
$$3 + 5 + 7 = 15$$
$$8 + 1 + 6 = 15$$

and finally on the columns

$$4 + 3 + 8 = 15$$
$$9 + 5 + 1 = 15$$
$$2 + 7 + 6 = 15$$

A method to return **true** if the 2d-matrix of `int` passed as an argument is a magic square is as follows:

```
bool isMagic(int[,] arrayP)
{
    bool magicSoFar = true;
    if (arrayP.GetLength(0) == arrayP.GetLength(1))
    { // The array is a square.
        int magicConstant = 0;
        for (int i = 0; i < arrayP.GetLength(1); i++)
        {
            magicConstant += arrayP[i, i];
        }
        int testedValue = 0;
        for (int i = 0; i < arrayP.GetLength(1); i++)
        {
            testedValue += arrayP[i, arrayP.GetLength(1) -
 ↪  i - 1];
        }
```

```
    if (testedValue == magicConstant)
    {// The diagonal and anti-diagonal have the same
     ↪  sums.
        // We test the rows.
        for (int row = 0; row < arrayP.GetLength(0);
         ↪  row++)
        {
            testedValue = 0;
            for (int col = 0; col <
             ↪  arrayP.GetLength(1); col++)
            {
                testedValue += arrayP[row, col];

            }

            if (testedValue != magicConstant)
            {
                magicSoFar = false;
            }
        }
        // We test the columns.
        for (int col = 0; col < arrayP.GetLength(1);
         ↪  col++)
        {
            testedValue = 0;
            for (int row = 0; row <
             ↪  arrayP.GetLength(0); row++)
            {
                testedValue += arrayP[row, col];
            }

            if (testedValue != magicConstant)
            {
                magicSoFar = false;
            }
        }
    }
    else
    {// The diagonal and anti-diagonal have different
     ↪  same sums.
        magicSoFar = false;
    }
}
else
{ // The array is not a square.
    magicSoFar = false;
```

```
    }

    return magicSoFar;
}
```

That code can be tested using for example this code[2].

[2] https:/princomp.github.io/code/snippets/magicSquare_test.cs