

Contents

Manipulating Rectangular Arrays	1
Summing the values row per row	1
Computing Magic Square	2

Manipulating Rectangular Arrays

We present below some simple algorithms to manipulate 2-dimensional (rectangular) arrays. The code for this lecture is available in this archive.

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 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.

The following is an example of a magic square:

```
int[,] arrayP1 =  
{  
    { 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:

```
static class MagicSquare  
{  
    public static 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;
}
}

```