**6 kyu**

**Pascal's Triangle**

3784989% of8321,093 of 7,306[boatmeme](https://www.codewars.com/users/boatmeme)

C#

* [TRAIN AGAIN](https://www.codewars.com/kata/pascals-triangle/train/csharp)
* [NEXT KATA](https://www.codewars.com/trainer/csharp)

Details

[Solutions](https://www.codewars.com/kata/pascals-triangle/solutions/csharp)

[Forks (4)](https://www.codewars.com/kata/pascals-triangle/forks/csharp)

[Discourse (108)](https://www.codewars.com/kata/pascals-triangle/discuss/csharp)

* Add to Collection
* |
* Share this kata:

#Pascal's Triangle



Wikipedia article on Pascal's Triangle: <http://en.wikipedia.org/wiki/Pascal's_triangle>

Write a function that, given a depth (n), returns a single-dimensional array/list representing Pascal's Triangle to the n-th level.

For example:

Kata.PascalsTriangle(4) == new List<int> {1,1,1,1,2,1,1,3,3,1}

<https://www.codewars.com/kata/pascals-triangle/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

public static List<int> PascalsTriangle(int n)

{

int[,] matriz = new int[n, n];

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

{

matriz[i, 0] = 1;

matriz[0, i] = 1;

}

for (int i = 1; i < n; i++)

for (int j = 1; j < n; j++)

matriz[i, j] = matriz[i, j - 1] + matriz[i - 1, j];

List<int> ans = new List<int>();

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

{

int fila = i;

int col = 0;

while(fila >=0 && col < n)

{

//Console.Write(matriz[fila, col] + " ");

ans.Add(matriz[fila, col]);

fila--;

col++;

}

}

return ans;

}

public static List<int> PascalsTriangle(int n)

{

List<int> pt = new List<int>();

pt.Add(1);

for (int i = 1; i < n; i++)

{

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

{

if (j == 0 || j == i)

pt.Add(1);

else

{

pt.Add(pt[pt.Count - i] + pt[pt.Count - (i + 1)]);

}

}

}

return pt;

}

static void Main(string[] args)

{

foreach(int item in PascalsTriangle(15))

{

Console.Write(item + " ");

}

Console.ReadLine();

}

}

}