static int[] Mult(int[,] arrDim, int[] arr)

if (arrDim.GetLength(1) != arr.Length)

return null;

int[] ergenbis = new int[arrDim.GetLength(0)];

for (int i = 0; i < arrDim.GetLength(0); i++)

{

int result = 0;

for (int j = 0; j < arrDim.GetLength(1); j++)

{ result += arrDim[i, j] \* arr[j]; }

ergenbis[i] = result;

}

return ergenbis;

int[,] A = { { 3, 0, 2 }, { 1, 2, 0, } };

int[] x = { 1, 2, 3 };

int[] y = Mult(A, x); // 3\*1 + 0\*2 + 2\*3 за всяка колона

--------------------------------------------------------------------------

int[,] arrDim = {

{1, 2},

{3, 4},

{5, 6},

{7, 8}

};

// && and, || or, else if

// out int name

// name = 0;

int rows = arrDim.GetLength(0);

// Броя на редовете (zeile) (4)

int cols = arrDim.GetLength(1);

// Броя на колоните (spalten) (2)

--------------------------------------------------------------------------

static bool Contains(string s1, string s2)

for (int i = 0; i <= s1.Length - s2.Length; i++)

{

bool found = true;

for (int j = 0; j < s2.Length; j++)

{

if (s1[i + j] != s2[j])

{

found = false;

break;

}

}

if (found)

{

return true;

}

}

return false;

}

Console.WriteLine(Contains("Grünes Gras", "Gras"));

static int BinarySearch(int[] array, int target)

{

int left = 0;

int right = array.Length - 1;

while (left <= right)

{

int mid = left + (right - left) / 2;

if (array[mid] == target)

{

return mid;

}

else if (array[mid] < target)

{

left = mid + 1;

}

else

{

right = mid - 1;

}

}

return -1;

}

int[] numbers = { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 };

int target = 21;

int result = BinarySearch(numbers, target);

static void SortArr(int[] arr)

for (int i = 0; i < arr.Length - 1; i++)

{

int aktuelle = arr[i];

for (int j = i + 1; j < arr.Length; j++)

{

if (arr[j] < arr[i])

{

arr[i] = arr[j];

arr[j] = aktuelle;

}

}

}

int[] arr = { 3, 1, 4, 2 };

Sort(arr); // 1 2 3 4

static int[,] Transponieren(int[,] f)

int[,] f2 = new int[f.GetLength(1),f.GetLength(0)];

for(int i = 0; i < f.GetLength(0); i++)

{

for (int j = 0; j < f.GetLength(1); j++)

{

f2[j, i] = f[i, j];

}

}

return f2;

int[,] f = { { 0, 1, 2, 3 }, { 4, 5, 6, 7 } };

int[,] f2 = Transponieren(f);

// int[,] f2 = {{0,4}, {1,5} …}

// int[колона, елементId] f2[1,1] = 5

static void ReverseArray(int[] arr)

for(int i = 0; i < arr.Length/2; i++)

{

int element = arr[i];

arr[i] = arr[(arr.Length - 1) - i];

arr[(arr.Length - 1) - i] = element;

}

static int[] MergeSortedArrays(int[] arr1, int[] arr2)

{

int[] result = new int[arr1.Length + arr2.Length];

int i = 0;

int j = 0;

for (int k = 0; k < result.Length; k++)

{

if (i < arr1.Length && (j >= arr2.Length || arr1[i] < arr2[j]))

{

result[k] = arr1[i++];

}

else

{

result[k] = arr2[j++];

}

}

return result;

}

int[] arr1 = { 1, 3, 5, 7, 8, 8, 9 };

int[] arr2 = { 2, 4, 6, 8 ,8};

int[] mergedArray = MergeSortedArrays(arr1, arr2);

// 1 2 3 4 5 6 7 8 8 8 8 9

static bool Teilmenge(int[] a, int[] b)

for (int i = 0; i < b.Length; i++)

{

bool contain = false;

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

{

if (b[i] == a[j])

{

contain = true;

break;

}

}

if (!contain)

return false;

}

return true;

int[] a = { 1, 2, 3, 4, 5 }; int[] b = { 2, 3, 4 }; Console.WriteLine(Teilmenge1(a, b)); //true

static bool Enthaelt(int zahl, int ziffer, bool genau1 = false)

int anzahl = 0;

while (zahl > 0)

{

int letzteZiffer = zahl % 10;

if (letzteZiffer == ziffer)

anzahl++;

zahl /= 10;

}

if (genau1)

return anzahl == 1;

else

return anzahl > 0;

// Enthaelt(70, 7, true) – true Enthaelt(77, 7, true) – false

// Enthaelt(70, 7, false) – true Enthaelt(77, 7, false) – true

static bool PrimzahlTest(int number)

if (number < 2) return false;

for (int i = 2; i \* i <= number; i++)

{

if (number % i == 0)

{

return false;

}

}

return true;

static int Twist(int zahl)

int twist = 0;

while (zahl > 0)

{

int ziffer = zahl % 10;

twist = twist \* 10 + ziffer; // 0 \* 10 + 7 = 7; 7 \* 10 + 6 = 76

zahl /= 10;

}

return twist;

Twist(4567) // 7654

switch (nameTypeInt) switch (nameTypeString)

{ {

case 1: case “baba”:

return sum + 4.9; return “result”;

case 2: case “dada”:

return sum + 6.5; return “result”;

case 3: defaut:

return sum + 12; return “result”;

default: }

return sum = -1;

}

static int[] Compact(int[] input)

int[] result = new int[input.Length];

int index = 0;

for (int i = 0; i < input.Length; i++)

{

if (input[i] >= 0)

{

result[index++] = input[i];

}

}

for (int i = index; i < result.Length; i++)

{

result[i] = -1; // modify

}

return result;

int[] inputArray1 = { 1, 2, 3, -3, -4, 4, -5 };

int[] resultArray1 = Compact(inputArray1); // 1 2 3 4 -1 -1 -1

static int CountNumbersDivisibleOrContainingDigit(int digit)

int count = 0;

for (int num = 100; num <= 999; num++)

{

if (num % digit == 0)

count++;

else

{

// Разделяме числото на отделни цифри

int hundreds = num / 100; // Първата цифра

int tens = (num / 10) % 10; // Втората цифра

int ones = num % 10; // Третата цифра

if (hundreds == digit || tens == digit || ones == digit)

{

count++;

}

}

static int SumRowMinusColumn(int[,] m, int k)

int rows = m.GetLength(0);

int cols = m.GetLength(1);

if (k >= rows || k >= cols)

{

return 0;

}

int rowSum = 0;

int colSum = 0;

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

{

rowSum += m[k, i];

}

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

{

colSum += m[i, k];

}

return rowSum - colSum;

static int[,] ZeileLöschen(int[,] m, int znr)

{

int rows = m.GetLength(0);

int cols = m.GetLength(1);

int[,] result = new int[rows - 1, cols];

int resultRow = 0;

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

{

if (i != znr)

{

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

{

result[resultRow, j] = m[i, j];

} int[,] matrix = {

resultRow++; { 1, 2, 3 },

} { 4, 5, 6 },

} { 7, 8, 9 }

return result; };

int[,] result = ZeileLöschen(matrix, 1);

int[,] result = { { 1, 2, 3 },{ 7, 8, 9 } };

static void Minima(int[] array) // проверка масив > 3

// Намиране на локалните минимуми

for (int i = 1; i < array.Length - 1; i++)

{

if (array[i] < array[i - 1] && array[i] < array[i + 1])

{

Console.WriteLine(array[i]);

}

}

int[] array = { 10, 1, 20, 5, 4, 11, 3, 7, 6 }; // 1 4 3

static string Addiere(string num1, string num2)

{

int length = num1.Length;

int carry = 0;

string result = "";

for (int i = length - 1; i >= 0; i--)

{

int digit1 = num1[i] - '0';

int digit2 = num2[i] - '0';

int sum = digit1 + digit2 + carry;

carry = sum / 10;

result = (sum % 10) + result;

}

if (carry > 0)

{

result = carry + result;

}

return result;

static Eigenschaft Gerade(int[] feld)

bool hasEven = false;

bool hasOdd = false;

foreach (int num in feld)

{

if (num % 2 == 0)

hasEven = true;

else

hasOdd = true;

}

if (hasEven && hasOdd)

return Eigenschaft.gemischt;

int[] feld3 = { 1, 2, 3, 4 }; // смесени

static int AlternierendeQuersumme(int zahl)

{

int summe = 0;

bool addieren = false;

while (zahl != 0)

{

int ziffer = zahl % 10;

if (addieren)

{

summe += ziffer;

}

else

{

summe -= ziffer;

}

addieren = !addieren;

zahl /= 10;

}

return summe;

//Eingabe: 1048576

//Alt. Quersumme: 6 – 7 + 5 – 8 + 4 – 0 + 1 = 1

static double[] Mittelwerte(double[,] feld)

double[] ergebnis = new double[feld.GetLength(0)];

for (int i = 0; i < feld.GetLength(0); i++)

{

double summe = 0;

for (int j = 0; j < feld.GetLength(1); j++)

{

summe += feld[i, j];

}

ergebnis[i] = summe / feld.GetLength(1);

}

return ergebnis;

static string RemoveBlanks(string input)

string result = "";

bool lastWasSpace = false;

foreach (char c in input)

{

if (c == ' ')

{

if (!lastWasSpace)

{

result += c;

lastWasSpace = true;

}

}

else

{

result += c;

lastWasSpace = false;

}

}

return result;

static int Wortzahl(string input)

int wordCount = 0;

bool inWord = false;

foreach (char c in input)

{

if (c != ' ')

{

if (!inWord)

{

wordCount++;

inWord = true;

}

}

else

{

inWord = false;

}

return wordCount; // da asd asdw -> 3

static int Randsumme(int[,] feld)

int summe = 0;

int rows = feld.GetLength(0);

int cols = feld.GetLength(1);

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

{

summe += feld[0, i];

}

if (rows > 1)

{

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

{

summe += feld[rows - 1, i];

}

}

for (int i = 1; i < rows - 1; i++)

{

summe += feld[i, 0]; // Първата колона

summe += feld[i, cols - 1]; // Последната колона

}

return summe;

static int MaxDifferenz(int[] feld, out int startIndex, out int zielIndex)

{

int maxDifferenz = 0;

startIndex = 0;

zielIndex = 0;

for (int i = 0; i < feld.Length; i++)

{

for (int j = i + 1; j < feld.Length; j++)

{

int differenz = feld[j] - feld[i];

if (differenz > maxDifferenz)

{

maxDifferenz = differenz;

startIndex = i;

zielIndex = j;

}

}

}

return maxDifferenz;

static int Randsumme(int[,] feld)

{

int summe = 0;

int rows = feld.GetLength(0);

int cols = feld.GetLength(1);

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

{

summe += feld[0, i];

}

if (rows > 1)

{

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

{

summe += feld[rows - 1, i];

}

}

for (int i = 1; i < rows - 1; i++)

{

summe += feld[i, 0];

summe += feld[i, cols - 1];

}

return summe;

}