Лабораторная работа по С# №1

Работа с массивами. Методы

Задание 1

//Console.WriteLine("20.Вывести на экран все локальные минимумы.");

int size, min, max;

Random rand = new Random();

while (true)

{

Console.Write("Размер массива [1..100]");

if (int.TryParse(Console.ReadLine(), out size) && size > 0 && size <= 100)

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("min: ");

if (int.TryParse(Console.ReadLine(), out min))

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("max: ");

if (int.TryParse(Console.ReadLine(), out max) && max>min)

break;

Console.WriteLine("неверное значение");

}

int[] m=new int[size];

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

{

m[i] = rand.Next(min, max+1);

Console.Write($"{m[i]}\t");

}

Console.WriteLine();

//Вариант 20

//for (int i = 1; i < size-1; i++) {

// if (m[i] < m[i-1] & m[i] < m[i+1])

// Console.WriteLine($"m[{i}] = {m[i]}");

//}

//Вариант 1 Вывести на экран индексы всех отрицательных элементов.

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

{

if (m[i] < 0) Console.Write($"{i}\t");

}

//Вариант 2 Заменить все отрицательные элементы нулями.

//for (int i = 0; i < size; i++)

// {

// if (m[i] < 0) m[i]=0;

// Console.Write($"{m[i]}\t");

// }

//Вариант 3 Заменить все отрицательные элементы на противоположные.

//for (int i = 0; i < size; i++)

// {

// if (m[i] < 0) m[i]\*=-1;

// Console.Write($"{m[i]}\t");

// }

//Вариант 4 Поменять местами максимальный и минимальный элементы (если таких элементов

//несколько – взять первые из них).

//int maxi = 0;

//int mini = 0;

//int maxar = m[maxi];

//int minar = m[mini];

//for (int i = 0; i < size; i++)

// {

// if (m[i] < minar) { minar = m[i];mini = i; }

// if (m[i] > maxar) { maxar = m[i];maxi = i; }

// //Console.Write($"{m[i]}\t");

// }

//int t;

//t = m[maxi];

//m[maxi] = m[mini];

//m[mini] = t;

//for (int i = 0; i < size; i++)

//{

// Console.Write($"{m[i]}\t");

//}

//Вариант 5 Найти индекс первого минимального элемента.

//int mini = 0;

//int minar = m[mini];

//for (int i = 0; i < size; i++)

// {

// if (m[i] < minar) { minar = m[i];mini = i; }

// }

//Console.WriteLine(mini);

//Вариант 6 Найти индекс последнего максимального элемента.

//int maxi = 0;

//int maxar = m[maxi];

//for (int i = 0; i < size; i++)

// {

// if (m[i] >= maxar) { maxar = m[i];maxi = i; }

// }

//Console.WriteLine(maxi);

//Вариант 7 Подсчитать сумму элементов, расположенных между максимальным и минимальным

//элементами(если таких элементов несколько – взять первые из них).

//int maxi = 0;

//int mini = 0;

//int maxar = m[maxi];

//int minar = m[mini];

//for (int i = 0; i < size; i++)

//{

// if (m[i] < minar) { minar = m[i]; mini = i; }

// if (m[i] > maxar) { maxar = m[i]; maxi = i; }

// //Console.Write($"{m[i]}\t");

//}

//if(maxi>mini) Console.WriteLine(maxi-mini - 1);

//else Console.WriteLine(mini - maxi - 1);

//Вариант 8 Найти индекс первого максимального элемента.

//int maxi = 0;

//int maxar = m[maxi];

//for (int i = 0; i < size; i++)

//{

// if (m[i] > maxar) { maxar = m[i]; maxi = i; }

// //Console.Write($"{m[i]}\t");

//}

//Console.WriteLine(maxi);

//Вариант 9 Найти индекс последнего минимального элемента.

//int mini = 0;

//int minar = m[mini];

//for (int i = 0; i < size; i++)

//{

// if (m[i] <= minar) { minar = m[i]; mini = i; }

//}

//Console.WriteLine(mini);

//Вариант 10 Подсчитать сумму элементов, расположенных между первым максимальным и последним

//минимальными элементами.

//int maxi = 0;

//int mini = 0;

//int maxar = m[maxi];

//int minar = m[mini];

//for (int i = 0; i < size; i++)

//{

// if (m[i] <= minar) { minar = m[i]; mini = i; }

// if (m[i] > maxar) { maxar = m[i]; maxi = i; }

//}

//int sum = 0;

//if (maxi < mini)

//{

// for (int i = maxi+1; i < mini; i++)

// {

// sum += m[i];

// }

//}

//else

//{

// for (int i = mini + 1; i < maxi; i++)

// {

// sum += m[i];

// }

//}

//Console.WriteLine(sum);

//Вариант 11 Поменять местами первый минимальный и последний максимальный элементы.

//int maxi = 0;

//int mini = 0;

//int maxar = m[maxi];

//int minar = m[mini];

//for (int i = 0; i < size; i++)

//{

// if (m[i] < minar) { minar = m[i]; mini = i; }

// if (m[i] >= maxar) { maxar = m[i]; maxi = i; }

//}

//int t;

//t = m[maxi];

//m[maxi] = m[mini];

//m[mini] = t;

//for (int i = 0; i < size; i++)

//{

// Console.Write($"{m[i]}\t");

//}

//Вариант 12 Найти максимум из отрицательных элементов.

//int maxi = 0;

//int maxar = m[maxi];

//for (int i = 0; i < size; i++)

//{

// if (m[i]<0 && m[i] > maxar) { maxar = m[i]; maxi = i; }

//}

//Console.WriteLine(maxar);

//Вариант 13 Найти минимум из положительных элементов.

//int mini = 0;

//int minar = m[mini];

//for (int i = 0; i < size; i++)

//{

// if (m[i] > 0 && m[i] < minar) { minar = m[i]; mini = i; }

//}

//Console.WriteLine(minar);

//Вариант 14 Найти максимум из модулей элементов.

//int maxi = 0;

//int maxar = m[maxi];

//for (int i = 0; i < size; i++)

//{

// if ((m[i]) <0) { m[i] \*= -1; }

// if (m[i] > maxar) { maxar = m[i]; }

//}

//Console.WriteLine(maxar);

//Вариант 15 Найти количество пар соседних элементов, разность между которыми равна заданному

//числу.

//int a= Convert.ToInt32(Console.ReadLine());

//int c = 0;

//for (int i = 0;i < size-1; i++)

//{

// if (a == m[i] - m[i+1] || a == m[i + 1] - m[i])

// {

// c++;

// }

//}

//Console.WriteLine(c);

//Вариант 16 Подсчитать количество элементов, значения которых больше значения предыдущего

//элемента.

//int c = 0;

//for (int i = 1; i < size; i++)

//{

// if (m[i-1] < m[i])

// {

// c++;

// }

//}

//Console.WriteLine(c);

//Вариант 17 Найти количество пар соседних элементов, в которых предыдущий элемент кратен

//последующему.

//int c = 0;

//for (int i = 1; i < size; i++)

//{

// if (m[i - 1] % m[i]==0)

// {

// c++;

// }

//}

//Console.WriteLine(c);

//Вариант 18 Найти длину максимальной возрастающей последовательности элементов.

//int c = 0;

//int cmax = 0;

//for (int i = 1; i < size; i++)

//{

// if (m[i - 1] < m[i])

// {

// c++;

// }

// else { c = 0;}

// if (c >= cmax) cmax = c+1;

//}

//Console.WriteLine(cmax);

//Вариант 19 Найти длину максимальной неубывающей последовательности элементов.

//int c = 0;

//int cmax = 0;

//for (int i = 1; i < size; i++)

//{

// if (m[i - 1] <= m[i])

// {

// c++;

// }

// else { c = 0; }

// if (c >= cmax) cmax = c + 1;

//}

//Console.WriteLine(cmax);

Console.ReadKey();

Задание 2

Console.WriteLine("Определить, есть ли в данном массиве квадрат 2х2, состоящий только из нулевых эллементов.");

int size, min, max;

Random rand = new Random();

while (true)

{

Console.Write("Размер массива [1..100]");

if (int.TryParse(Console.ReadLine(), out size) && size > 0 && size <= 100)

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("min: ");

if (int.TryParse(Console.ReadLine(), out min))

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("max: ");

if (int.TryParse(Console.ReadLine(), out max) && max > min)

break;

Console.WriteLine("неверное значение");

}

int[,] m = new int[size,size];

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

{

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

{

m[i,j] = rand.Next(min, max + 1);

Console.Write($"{m[i,j]}\t");

}

Console.WriteLine();

}

Console.WriteLine();

//Вариант 20

//bool fl = false;

//for (int i = 0; i < size-1; i++)

//{

// for (int j = 0; j < size-1; j++)

// {

// if (m[i, j] == 0 && m[i, j + 1] == 0 && m[i + 1, j] == 0 && m[i + 1, j + 1] == 0) {

// fl = true;

// goto m1;

// }

// }

//}

//m1:

//if (fl)

//{

// Console.WriteLine("yes");

//}

//else {

// Console.WriteLine("no");

//}

//Вариант 1 Подсчитать среднее арифметическое четных элементов, расположенных ниже главной

//диагонали(элементу лежащему ниже побочной диагонали соответствует выражение i + j

//+ 1 > n).

double sum = 0;

double c = 0;

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

{

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

{

if (j < i && m[i, j] % 2 == 0)

{

sum += m[i, j];

c++;

}

}

}

Console.WriteLine($"{sum / c:#.##}");

//Вариант 2 Подсчитать сумму элементов, расположенных на побочной диагонали.

//int sum = 0;

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (j ==size - i - 1)

// {

// sum += m[i, j];

// }

// }

//}

//Console.WriteLine($"{sum}");

//Вариант 3 Подсчитать среднее арифметическое ненулевых элементов, расположенных над

//побочной диагональю.

//double sum = 0;

//double c = 0;

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (j+i+1<size && m[i,j]!=0)

// {

// sum += m[i, j];

// c++;

// }

// }

//}

//Console.WriteLine($"{sum/c:#.##}");

//Вариант 4 Подсчитать среднее арифметическое элементов, расположенных под побочной

//диагональю.

//double sum = 0;

//double c = 0;

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (j + i + 1 > size )

// {

// sum += m[i, j];

// c++;

// }

// }

//}

//Console.WriteLine($"{sum / c:#.##}");

//Вариант 5 Поменять местами столбцы по правилу: первый с последним, второй с предпоследним

//и т.д.

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size/2; j++)

// {

// int t;

// t= m[i,j];

// m[i, j] = m[i, size - j - 1];

// m[i, size - j - 1] = t;

// }

// }

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 6 Поменять местами две средних строки, если количество строк четное, и первую со

//средней строкой, если количество строк нечетное.

//if (size % 2 == 0) {

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (i == size / 2-1)

// {

// int t;

// t = m[i, j];

// m[i, j] = m[i+1,j];

// m[i+1,j] = t;

// }

// }

// }

//}

//else

//{

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (i == size / 2)

// {

// int t;

// t = m[0, j];

// m[0, j] = m[i, j];

// m[i,j] = t;

// }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 7 Поменять местами два средних столбца, если количество столбцов четное, и первый

//со средним столбцом, если количество столбцов нечетное.

//if (size % 2 == 0)

//{

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (j == size / 2 - 1)

// {

// int t;

// t = m[i, j];

// m[i, j] = m[i, j+1];

// m[i, j+1] = t;

// }

// }

// }

//}

//else

//{

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (j == size / 2)

// {

// int t;

// t = m[i, 0];

// m[i, 0] = m[i, j];

// m[i, j] = t;

// }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 8 Если количество строк в массиве четное, то поменять строки местами по правилу:

//первую строку со второй, третью – с четвертой и т.д. Если количество строк в массиве

//нечетное, то оставить массив без изменений.

//if (size % 2 == 0)

//{

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (i%2==0)

// {

// int t;

// t = m[i, j];

// m[i, j] = m[i+1, j];

// m[i+1, j] = t;

// }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 9 Если количество столбцов в массиве четное, то поменять столбцы местами по

//правилу: первый столбец со вторым, третий – с четвертым и т.д. Если количество

//столбцов в массиве нечетное, то оставить массив без изменений.

//if (size % 2 == 0)

//{

// for (int i = 0; i < size; i++)

// {

// for (int j = 0; j < size; j++)

// {

// if (j % 2 == 0)

// {

// int t;

// t = m[i, j];

// m[i, j] = m[i, j+1];

// m[i, j+1] = t;

// }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 11 Подсчитать норму матрицы по формуле  =

//i j j i a A , max (сумма максимумов строк).

//int A = 0;

//for (int i = 0; i < size; i++)

//{

// int maxa = m[i, 0];

// for (int j = 0; j < size; j++)

// {

// if (m[i, j] > maxa) maxa = m[i, j];

// }

// A += maxa;

//}

//Console.WriteLine(A);

//Вариант 12 Подсчитать норму матрицы по формуле  =

//j i j i a A , max (сумма максимумов столбцов).

//int A = 0;

//for (int i = 0; i < size; i++)

//{

// int maxa = m[0, i];

// for (int j = 0; j < size; j++)

// {

// if (m[j, i] > maxa) maxa = m[j, i];

// }

// A += maxa;

//}

//Console.WriteLine(A);

//Вариант 13 Вывести элементы матрицы в следующем порядке:

//for (int i = 0; i < size; i++)

//{

// if (i % 2 == 0)

// {

// for (int j = 0; j < size; j++)

// {

// {

// Console.Write($"{m[i,j]}\t");

// }

// }

// Console.WriteLine();

// }

// else

// {

// for (int j = size-1; j >=0; j--)

// {

// {

// Console.Write($"{m[i, j]}\t");

// }

// }

// Console.WriteLine();

// }

//}

//Вариант 14 Выяснить, является ли матрица симметричной

//относительно главной диагонали.

//bool fl = false;

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (m[i, j] == m[j, i]) { fl = true; }

// else { fl = false;break; }

// }

//}

//Console.WriteLine(fl);

//Вариант 15 Вывести элементы матрицы в следующем порядке:

//for (int i = 0; i < size; i++)

//{

// if (i % 2 == 0)

// {

// for (int j = 0; j < size; j++)

// {

// {

// Console.Write($"{m[j, i]}\t");

// }

// }

// Console.WriteLine();

// }

// else

// {

// for (int j = size - 1; j >= 0; j--)

// {

// {

// Console.Write($"{m[j, i]}\t");

// }

// }

// Console.WriteLine();

// }

//}

//Вариант 16 Определить, есть ли в данном массиве строка, состоящая только из положительных

//элементов.

//bool fl = false;

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 0; j < size; j++)

// {

// if (m[i, j] >0) { c++; }

// if (c == size) fl = true;

// }

//}

//Console.WriteLine(fl);

//Вариант 17 Определить, есть ли в данном массиве столбец, состоящий только из отрицательных

//элементов.

//bool fl = false;

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 0; j < size; j++)

// {

// if (m[j, i] < 0) { c++; }

// if (c == size) fl = true;

// }

//}

//Console.WriteLine(fl);

//Вариант 18 В каждой строке найти максимум

//и заменить его на противоположный элемент.

//for (int i = 0; i < size; i++)

//{

// int maxi = i;

// int maxj = 0;

// int maxa = m[maxi, maxj];

// for (int j = 0; j < size; j++)

// {

// if (m[i, j] > maxa) {maxi = i; maxj = j; maxa = m[i, j]; }

// }

// m[maxi, maxj] \*= (-1);

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 19 В каждом столбце найти минимум и заменить его нулем.

//for (int i = 0; i < size; i++)

//{

// int mini = 0;

// int minj = i;

// int mina = m[mini, minj];

// for (int j = 0; j < size; j++)

// {

// if (m[j, i] < mina) { mini = j; minj = i; mina = m[j, i]; }

// }

// m[mini, minj] = 0;

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// Console.Write($"{m[i, j]}\t");

// }

// Console.WriteLine();

//}

Console.ReadKey();

Задание 3

//Console.WriteLine("20.Для каждой строки вычислить произведение ее длины на минимальный элемент." +

// "Данные записать в новый массив.");

int size, min, max;

Random rand = new Random();

while (true)

{

Console.Write("Размер массива [1..100]");

if (int.TryParse(Console.ReadLine(), out size) && size > 0 && size <= 100)

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("min: ");

if (int.TryParse(Console.ReadLine(), out min))

break;

Console.WriteLine("неверное значение");

}

while (true)

{

Console.Write("max: ");

if (int.TryParse(Console.ReadLine(), out max) && max > min)

break;

Console.WriteLine("неверное значение");

}

int[][] m = new int[size][];

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

{

m[i] = new int[rand.Next(1,size+1)];

for (int j = 0; j < m[i].Length; j++)

{

m[i][j] = rand.Next(min, max + 1);

Console.Write($"{m[i][j]}\t");

}

Console.WriteLine();

}

Console.WriteLine();

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int minstr = m[i][0];

// for (int j = 1; j < m[i].Length; j++)

// {

// if (m[i][j] < minstr)minstr = m[i][j];

// }

// result[i] = minstr \* m[i].Length;

//}

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 1 Найти минимальный элемент в каждом столбце и записать данные в новый массив. Если

//в строке данный столбец отсутствует, элемент не учитывать.

int[] result = new int[size];

int[,] m1 = new int[size, size];

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

{

for (int j = 0; j < m[i].Length; j++)

{

m1[i, j] = m[i][j];

}

}

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

{

int minst = m1[0, i];

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

{

if (m1[j, i] < minst) minst = m1[j, i];

}

result[i] = minst;

}

Console.WriteLine();

foreach (int i in result)

{

Console.Write($"{i}\t");

}

//Вариант 2 Четные строки таблицы заменить на вектор [1, 0, 1, 0…].

//for (int i = 0; i < size; i++)

//{

// if (i % 2 == 0)

// {

// for (int j = 0; j < m[i].Length; j++)

// {

// if(j%2 == 0) { m[i][j] = 1; }

// else { m[i][j] = 0; }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// Console.Write($"{m[i][j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 3 Нечетные строки таблицы заменить на вектор [1, 0, 1, 0…].

//for (int i = 0; i < size; i++)

//{

// if (i % 2!= 0)

// {

// for (int j = 0; j < m[i].Length; j++)

// {

// if (j % 2 == 0) { m[i][j] = 1; }

// else { m[i][j] = 0; }

// }

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// Console.Write($"{m[i][j]}\t");

// }

// Console.WriteLine();

//}

//Вариант 4 Для каждой строки подсчитать количество

//элементов и записать данные в новый массив.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// result[i] = m[i].Length;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 5 Для каждой строки подсчитать количество

//положительных элементов и записать данные в

//новый массив.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 0; j < m[i].Length; j++)

// {

// if (m[i][j] > 0) c++;

// }

// result[i] = c;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 6 Для каждого столбца подсчитать сумму

//отрицательных элементов и записать данные в

//новый массив. Если в строке данный столбец отсутствует,

//считать элемент равным 0.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int sum = 0;

// for (int j = 0; j < size; j++)

// {

// if (m1[j, i] < 0) sum+= m1[j, i];

// }

// result[i] = sum;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 7 Для каждого столбца подсчитать

//сумму четных положительных элементов и записать

//данные в новый массив. Если в строке данный

//столбец отсутствует, считать элемент равным 0.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int sum = 0;

// for (int j = 0; j < size; j++)

// {

// if (m1[j, i] > 0 && m1[j,i]%2==0) sum += m1[j, i];

// }

// result[i] = sum;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 8 Для каждой строки подсчитать количество

//элементов, больших заданного числа, и

//записать данные в новый массив.

//int n;

//while (true)

//{

// Console.Write("n: ");

// if (int.TryParse(Console.ReadLine(), out n))

// break;

// Console.WriteLine("неверное значение");

//}

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 0; j < m[i].Length; j++)

// {

// if (m[i][j] > n) c++;

// }

// result[i] = c;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 9 Для каждого столбца найти первый

//положительный элемент и записать данные в новый

//массив.Если в строке данный столбец

//отсутствует, считать элемент равным 0.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 0; j < size; j++)

// {

// if (m1[j, i] > 0 && c == 0) {result[i] = m1[j, i]; c++; }

// }

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 10 Для каждой строки найти последний четный элемент

//и записать данные в новый массив.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (m1[i,j]!=0 && m1[i, j]%2==0) { result[i] = m1[i, j];}

// }

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 11 Для каждого столбца найти номер последнего

//нечетного элемента и записать данные в

//новый массив. Если в строке данный столбец отсутствует,

//считать элемент равным 0.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// if (m1[j, i]%2!=0) { result[i] =j;}

// }

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 12 Для каждой строки найти номер первого отрицательного элемента и записать данные в

//новый массив.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// int k = 0;

// for (int j = 0; j < m[i].Length; j++)

// {

// if(m[i][j]<0 && c == 0) { result[i] = j;c++;}

// if (m[i][j] >= 0) k++;

// }

// if (k == m[i].Length) result[i] = -1;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 13 Для каждой строки найти сумму элементов с номерами

//от k1 до k2 и записать данные в

//новый массив. Если в строке данный столбец отсутствует,

//считать элемент равным 0.

//int k1;

//int k2;

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k1) && k1 >= 0 && k1<=size-1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k2) && k2 >= k1 && k2 <= size - 1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int sum = 0;

// for (int j = 0; j < size; j++)

// {

// if (j>=k1 && j<=k2) { sum += m1[i, j]; }

// }

// result[i] = sum;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 14 Для каждого столбца найти

//произведение элементов с номерами от k1 до k2 и записать

//данные в новый массив. Если в строке данный

//столбец отсутствует, считать элемент равным 1.

//int k1;

//int k2;

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k1) && k1 >= 0 && k1 <= size - 1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k2) && k2 >= k1 && k2 <= size - 1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// m1[i, j] = 1;

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int pr = 1;

// for (int j = 0; j < size; j++)

// {

// if (j >= k1 && j <= k2) { pr\*= m1[j, i]; }

// }

// result[i] = pr;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 15 Для каждой строки подсчитать сумму элементов,

//не попадающих в заданный интервал, и

//записать данные в новый массив.

//int k1;

//int k2;

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k1) && k1 >= 0 && k1 <= size - 1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//while (true)

//{

// if (int.TryParse(Console.ReadLine(), out k2) && k2 >= k1 && k2 <= size - 1)

// break;

// else

// {

// Console.WriteLine("неверное значение");

// }

//}

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int sum = 0;

// for (int j = 0; j < size; j++)

// {

// if (j < k1 || j > k2) { sum += m1[i, j]; }

// }

// result[i] = sum;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 16 Подсчитать сумму элементов каждой строки и

//записать данные в новый массив. Найти

//максимальный элемент нового массива.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int sum = 0;

// for (int j = 0; j < m[i].Length; j++)

// {

// sum+= m[i][j];

// }

// result[i] = sum;

//}

//Console.WriteLine();

//int maxr = result[0];

//for (int i = 0;i < size; i++)

//{

// if (result[i] > maxr) maxr = result[i];

//}

//Console.WriteLine(maxr);

//Вариант 17 Подсчитать произведение элементов каждого

//столбца и записать данные в новый массив.

//Если в строке данный столбец отсутствует,

//считать элемент равным 1.

//int[] result = new int[size];

//int[,] m1 = new int[size, size];

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < size; j++)

// {

// m1[i, j] = 1;

// }

//}

//for (int i = 0; i < size; i++)

//{

// for (int j = 0; j < m[i].Length; j++)

// {

// m1[i, j] = m[i][j];

// }

//}

//for (int i = 0; i < size; i++)

//{

// int pr = 1;

// for (int j = 0; j < size; j++)

// {

// pr \*= m1[j, i];

// }

// result[i] = pr;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 18 Для каждой строки найти номер первой

//пары неравных элементов. Данные записать в

//новый массив.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// for (int j = 1; j < m[i].Length; j++)

// {

// if (m[i][j - 1] != m[i][j] && c == 0) { result[i] = j - 1; c++; }

// }

// if(c == 0) result[i] = -1;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

//Вариант 19 Для каждой строки найти номер первой

//пары одинаковых элементов. Данные записать в

//новый массив.

//int[] result = new int[size];

//for (int i = 0; i < size; i++)

//{

// int c = 0;

// int k = 0;

// for (int j = 1; j < m[i].Length; j++)

// {

// if (m[i][j - 1] == m[i][j] && c == 0) { result[i] = j - 1; c++; }

// if (m[i][j - 1] != m[i][j]) { k++; }

// }

// if (k == m[i].Length - 1) result[i] = -1;

//}

//Console.WriteLine();

//foreach (int i in result)

//{

// Console.Write($"{i}\t");

//}

Console.ReadKey();

Задание 4

// Вариант 20 Console.WriteLine("Метод принимает Index и возвращает соответствующую букву английского алфавита(или \*," +

//"если индекс не корректен.");

// Console.WriteLine(m20(1));

// Console.WriteLine(m20(20));

// Console.WriteLine(m20(26));

// Console.WriteLine(m20(1000));

//var result=m8(true, false);

//Console.WriteLine(result);

//char m20(Index index)

//{

// if (index.Value < 0 || index.Value > 26)

// return '\*';

// return (char)((int)'A'-1 + index.Value);

//}

////Вариант 8 Метод принимает 2 параметра bool и получает рез-т всех булевых

////операций над ними(&,|,^). Возвращение рез-та - на ваше усмотрение

//(bool,bool,bool) m8(bool b1, bool b2)

//{

// return (b1 & b2, b1 | b2, b1 ^ b2);

//}

//Вариант 1 Метод принимает 3 целых числа и возвращает минимальное и максимальное из них

//(используйте кортеж).

(int, int) m1(int a, int b, int c)

{

int[] k = new int[3];

k[0] = a; k[1] = b; k[2] = c;

int max = k[0];

int min = k[0];

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

{

if (k[i] > max) { max = k[i]; }

if (k[i] < min) { min = k[i]; }

}

return (max, min);

}

(int, int) d = m1(9, 6, 15);

Console.WriteLine(d);

//Вариант 2 Метод принимает 3 вещественных и

//округляет их до целых (меняются сами

//переменные). Возвращает void.

//void m2(ref double a, ref double b, ref double c)

//{

// a = (int)a;

// b = (int)b;

// c = (int)c;

//}

//double a1 = 3.25;

//double b1 = 5.57;

//double c1 = 8.46;

//m2(ref a1, ref b1, ref c1);

//Console.WriteLine($"{a1} {b1} {c1}");

//Вариант 3 Метод принимает вещественное число и массив.

//Заполняет массив указанным числом.

//void m3(int[] ar, int a)

//{

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

// {

// ar[i]= a;

// }

//}

//int[] k = new int[10];

//m3(k, 4);

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

//{

// Console.Write($"{ k[i]}\t");

//}

//Вариант 4 Метод принимает набор целых чисел

//(используйте params) и формирует из них массив

//подходящего размера. Возвращает размер массива.

//int m4(params int[] a)

//{

// return a.Length;

//}

//int[] b = { 1, 2, 3, 4, 5, 6 };

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

//int[] d = { 1, 2 };

//int b1 = m4(b);

//int c1 = m4(c);

//int d1 = m4(d);

//Console.WriteLine($"{b1} {c1} {d1}");

//Вариант 5 Метод вычисляет ускорение свободного

//падения для любого космического тела. По

//умолчанию – для Земли.

//double m5( double m=5.9736E24, double r=6378E3, double G = 6.6743E-11){

// double g = G \* m/Math.Pow(r,2);

// return g;

//}

//double ml = 7.36E22;

//double rl = 1737.4E3;

//double gz = m5();

//double gl =m5(ml, rl);

//Console.WriteLine($"{gz:#.##} {gl:#.##}");

//Вариант 6 Метод принимает 3 целых переменных и меняет

//их значения по кругу (значение

//первой переменной записывается во вторую,

//второй – в третью, третьей – в первую).

//void m6(ref int a,ref int b,ref int c)

//{

// int t;

// t = b;

// b = a;

// a = c;

// c = t;

//}

//int a1 = 5;

//int b1 = 6;

//int c1 = 7;

//m6(ref a1, ref b1, ref c1);

//Console.WriteLine($"{a1} {b1} {c1}");

//Вариант 7 Метод принимает массив char и 5 переменных char.

//Значения из массива переписываются в переменные.

//Предусмотреть несовпадение длины.

//void m7(ref char a1, ref char a2, ref char a3, ref char a4, ref char a5, params char[] ar)

//{

// a1 = ar[0];

// a2 = ar[1];

// a3 = ar[2];

// a4 = ar[3];

// a5 = ar[4];

//}

//char[] k = {'a','b','c','d','e','r','u'};

//char b1='b';

//char b2='c';

//char b3='m';

//char b4='p';

//char b5='h';

//m7(ref b1, ref b2, ref b3, ref b4, ref b5, k);

//Console.WriteLine($"{b1} {b2} {b3} {b4} {b5}");

//Вариант 9 Метод принимает 2 кортежа (int, double)

//и (char, bool). Возвращает кортеж

//(bool, char, int, double).Исходные кортежи меняться не могут(in).

//(bool,char,int,double) m9(in (int, double) a, in (char, bool) b)

//{

// return (b.Item2,b.Item1,a.Item1,a.Item2);

//}

//(int, double) a1 = (3, 5.25);

//(char, bool) b1 = ('a', true);

//(bool, char, int, double) c =m9(a1,b1);

//Console.WriteLine(c);

//Вариант 10 Метод принимает 2 массива char

//и возвращает тот, который короче.

//char[] m10( char[]a, char[]b)

//{

// int a1 = a.Length;

// int b1 = b.Length;

// if(a1 > b1) { return b; }

// else { return a; }

//}

//char[] a2 = { 'a', 'b', 'c', 'd' };

//char[] b2 = { 'a', 'b' };

//char[] c=m10(a2,b2);

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

//{

// Console.Write($"{c[i]} ");

//}

//Вариант 11 Метод принимает массив и диапазон (Range).

//Оставляет в массиве только указанный

//диапазон(замена массива).

//void m11(ref int[] a, Range range)

//{

// a = a[range];

//}

//int[] b = { 1, 2, 3, 4, 5, 6, 7 };

//Range c = 1..4;

//m11(ref b, c);

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

//{

// Console.Write($"{b[i]} ");

//}

//Вариант 12 Метод принимает массив и 2 индекса (Index).

//Оставляет в массиве только элементы

//между индексами(замена массива).

//void m12(ref int[]a, Index start, Index end) {

// Range range = new Range(start, end);

// a = a[range];

//}

//int[] b = { 1, 2, 3, 4, 5, 6, 7 };

//Index c1 = 1;

//Index c2 = 4;

//m12(ref b, c1, c2);

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

//{

// Console.Write($"{b[i]} ");

//}

//Вариант 13 Метод принимает массив и диапазон (Range).

//Возвращает массив только указанного диапазона.

//int[] m13(in int[]ar,Range range)

//{

// int[] b = ar[range];

// return b;

//}

//int[] b = { 1, 2, 3, 4, 5, 6, 7 };

//Range c = 1..5;

//int[]d=m13(b, c);

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

//{

// Console.Write($"{d[i]} ");

//}

//Вариант 14 Метод принимает массив и 2 индекса (Index).

//Возвращает массив из элементов между индексами.

//int[] m14(in int[] a, Index start, Index end)

//{

// Range range = new Range(start, end);

// int[] b = a[range];

// return b;

//}

//int[] b = { 1, 2, 3, 4, 5, 6, 7 };

//Index c1 = 1;

//Index c2 = 4;

//int[] d= m14(b, c1, c2);

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

//{

// Console.Write($"{d[i]} ");

//}

//Вариант 15 Метод принимает массив и все

//отрицательные элементы заменяет на 0.

//void m15(ref int[] a)

//{

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

// {

// if (a[i] < 0) a[i]= 0;

// }

//}

//int[] b = { 1, 2, -3, 4, -5, -6, 7 };

//m15(ref b);

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

//{

// Console.Write($"{b[i]} ");

//}

//Вариант 16 Метод принимает массив и удаляет

//все отрицательные элементы (массив станет

//короче).

//void m16(ref int[] a)

//{

// int c = 0;

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

// {

// if (a[i] < 0) c++;

// }

// int[]b = new int[a.Length-c];

// int j = 0;

// for (int i = 0;i < a.Length;i++) {

// if (a[i] < 0) continue;

// else { b[j] = a[i]; j++; }

// }

// a= b;

//}

//int[] b = { 1, 2, -3, 4, -5, -6, 7 };

//m16(ref b);

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

//{

// Console.Write($"{b[i]} ");

//}

//Вариант 17 Метод принимает 2 параметра int и получает

//результат всех арифметических операций

//над ними(+, - , \*, /). Возвращение результата – на ваше усмотрение.

//(double,double,double,double) m17(int a, int b)

//{

// return (a + b, a - b, a \* b, a / b);

//}

//(double, double, double, double) c = m17(12323, 2342);

//Console.WriteLine(c);

//Вариант 18 Метод принимает английскую букву и bool.

//Если второй параметр true – буква

//перекодируется по алгоритму Цезаря(циклически смещается вправо

//на 13 по алфавиту). Если второй параметр false – буква декодируется.

//void m18(ref char a, bool b)

//{

// if (b)

// {

// a =(char)((int)a + 13);

// }

// else

// {

// a = (char)((int)a - 13);

// }

//}

//char b = 'a';

//Console.WriteLine(b);

//m18(ref b, true);

//Console.WriteLine(b);

//m18(ref b, false);

//Console.WriteLine(b);

//Вариант 19 Метод принимает русскую букву и bool.

//Если второй параметр true – буква

//перекодируется по алгоритму Цезаря(циклически

//смещается вправо на 13 по

//алфавиту). Если второй параметр false – буква декодируется.

//void m19(ref char a, bool b)

//{

// if (b)

// {

// a = (char)((int)a + 13);

// }

// else

// {

// a = (char)((int)a - 13);

// }

//}

//char b = 'в';

//Console.WriteLine(b);

//m19(ref b, true);

//Console.WriteLine(b);

//m19(ref b, false);

//Console.WriteLine(b);

Console.ReadKey();