Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М.Машерова

**Отчет**

по практическому заданию

«Модуль 1.3. Статические методы (методы классов)»

по учебной практике по программированию

Выполнила учащаяся Жданова С.О.

группы 3ПОИС23 29.09.2025

Проверил Алейников М.А.

29.09.2025

Орша, 2025

**Цель:** применить на практике ранее полученные знания о статических методах и методах класса в языке программирования C#.

**Задания**

**Задание 1.**

Определить функцию (статический метод) для вычисления наибольшего общего делителя двух целых натуральных чисел (Greatest Common Measure). В основной программе, используя функцию, сократить неотрицательную обыкновенную дробь. Дробь вводится с клавиатуры в виде неотрицательного числителя и положительного знаменателя.

**Задание 2.**

Присваивая последовательным элементам массива случайные значения от 1 до 9, создать массив с минимальным количеством элементов, сумма которых не превышает заданного пользователем числа.

**Задание 3.**

Сформировав квадратную целочисленную матрицу со случайными значениями элементов, упорядочить ее строки по возрастанию сумм их элементов. Пусть элементы принимают значения в диапазоне от –50 до +50.

**Ход выполнения работы**

**Выполнение практических заданий**

**Задание 1.**

Определить функцию (статический метод) для вычисления наибольшего общего делителя двух целых натуральных чисел (Greatest Common Measure). В основной программе, используя функцию, сократить неотрицательную обыкновенную дробь. Дробь вводится с клавиатуры в виде неотрицательного числителя и положительного знаменателя.

Ниже представлен листинг 3.1 с кодом программы для вычисления наибольшего общего делителя двух целых натуральных чисел.

Листинг 3.1. Код программы для вычисления наибольшего общего делителя двух целых натуральных чисел.

using System;

namespace FractionReducer

{

class Program

{

static void Main(string[] args)

{

// Выводим приветственное сообщение

Console.WriteLine("Программа сокращает обыкновенную дробь, используя НОД.");

// Запрашиваем числитель

Console.Write("Введите неотрицательный числитель: ");

string inputNumerator = Console.ReadLine();

int numerator;

if (!int.TryParse(inputNumerator, out numerator) || numerator < 0)

{

Console.WriteLine("Ошибка: числитель должен быть неотрицательным целым числом!"); // Сообщение об ошибке

return; // Завершаем программу при некорректном вводе

}

// Запрашиваем знаменатель

Console.Write("Введите положительный знаменатель: ");

string inputDenominator = Console.ReadLine();

int denominator;

if (!int.TryParse(inputDenominator, out denominator) || denominator <= 0)

{

Console.WriteLine("Ошибка: знаменатель должен быть положительным целым числом!"); // Сообщение об ошибке

return; // Завершаем программу при некорректном вводе

}

// Выводим исходную дробь

Console.WriteLine($"\nИсходная дробь: {numerator}/{denominator}");

// Вычисляем НОД числителя и знаменателя

int gcd = MathHelper.GreatestCommonDivisor(numerator, denominator);

// Сокращаем дробь

int reducedNumerator = numerator / gcd;

int reducedDenominator = denominator / gcd;

// Выводим сокращённую дробь

Console.WriteLine($"Сокращённая дробь: {reducedNumerator}/{reducedDenominator}");

}

}

// Класс для статических математических методов

static class MathHelper

{

// Статический метод для вычисления НОД двух чисел (алгоритм Евклида)

public static int GreatestCommonDivisor(int a, int b)

{

// Приводим числа к неотрицательным значениям

a = Math.Abs(a);

b = Math.Abs(b);

// Если одно из чисел равно 0, возвращаем другое (или 1, если оба 0)

if (a == 0 && b == 0)

{

return 1; // Соглашение: НОД(0,0) = 1, чтобы избежать деления на 0

}

if (a == 0)

{

return b;

}

if (b == 0)

{

return a;

}

// Алгоритм Евклида: НОД(a,b) = НОД(b, a mod b)

while (b != 0)

{

int temp = b;

b = a % b; // Остаток от деления

a = temp;

}

return a; // Возвращаем НОД

}

}

}

Далее запускаем написанный ранее код в среде разработке для получения результата, результат представлен на рисунке 1.3.1.

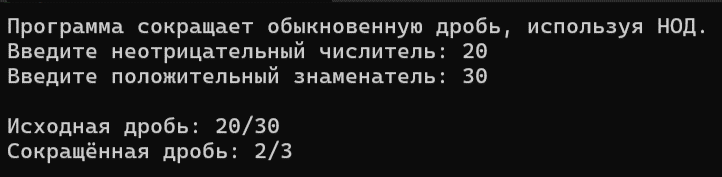


Рисунок 1.3.1 – Результат вычисления наибольшего общего делителя

**Задание 2.**

Присваивая последовательным элементам массива случайные значения от 1 до 9, создать массив с минимальным количеством элементов, сумма которых не превышает заданного пользователем числа.

Ниже представлен листинг 3.2 с кодом программы для создания массива с минимальным количеством элементом.

Листинг 3.2. Код программы для создания массива с минимальным количеством элементов.

using System;

using System.Collections.Generic;

namespace MinimalArraySum

{

class Program

{

static void Main(string[] args)

{

// Выводим приветственное сообщение

Console.WriteLine("Программа создаёт массив с минимальным количеством элементов, сумма которых не превышает заданное число.");

// Запрашиваем максимальную сумму

Console.Write("Введите максимальную сумму (положительное число): ");

string inputSum = Console.ReadLine();

int sumLimit;

if (!int.TryParse(inputSum, out sumLimit) || sumLimit <= 0)

{

Console.WriteLine("Ошибка: введите положительное целое число!"); // Сообщение об ошибке

return; // Завершаем программу при некорректном вводе

}

// Создаём массив с минимальным количеством элементов

int[] array = ArrayHelper.CreateMinimalArray(sumLimit);

// Выводим массив

Console.WriteLine("\nСозданный массив:");

ArrayHelper.PrintArray(array);

// Выводим сумму элементов

int sum = ArrayHelper.CalculateSum(array);

Console.WriteLine($"Сумма элементов: {sum}");

}

}

// Класс для статических методов обработки массивов

static class ArrayHelper

{

// Статический метод для создания массива с минимальным количеством элементов

public static int[] CreateMinimalArray(int sumLimit)

{

Random random = new Random(); // Создаём объект Random

List<int> tempList = new List<int>(); // Временный список для элементов

int remainingSum = sumLimit; // Остаток суммы для распределения

// Определяем начальное количество элементов (максимум 9 элементов для лимита)

int maxElements = Math.Min(9, sumLimit); // Не больше 9, так как числа от 1 до 9

int elementsCount = 1; // Минимальное количество элементов

while (elementsCount <= maxElements)

{

int targetSumPerElement = remainingSum / elementsCount; // Целевая сумма на элемент

if (targetSumPerElement >= 1) // Убеждаемся, что элемент может быть хотя бы 1

{

tempList.Clear(); // Очищаем список для нового расчёта

int currentSum = 0;

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

{

int maxValue = Math.Min(9, remainingSum - currentSum); // Максимальное возможное значение

if (maxValue < 1) break; // Если не можем добавить, прерываем

int value = random.Next(1, maxValue + 1); // Случайное число от 1 до maxValue

tempList.Add(value);

currentSum += value;

}

if (currentSum <= sumLimit) // Если сумма в пределах лимита, используем этот массив

{

remainingSum = sumLimit - currentSum;

if (remainingSum == 0 || elementsCount == maxElements)

{

return tempList.ToArray(); // Возвращаем массив, если больше добавить нельзя

}

}

}

elementsCount++; // Увеличиваем количество элементов, если текущий вариант не подошёл

}

// Если не удалось заполнить, возвращаем минимальный возможный массив

if (tempList.Count == 0 && sumLimit >= 1)

{

tempList.Add(sumLimit); // Если лимит позволяет, добавляем сам лимит

}

return tempList.ToArray();

}

// Статический метод для вывода массива

public static void PrintArray(int[] array)

{

if (array.Length == 0)

{

Console.WriteLine("Массив пустой.");

return;

}

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

{

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

}

Console.WriteLine(); // Переходим на новую строку

}

// Статический метод для вычисления суммы элементов

public static int CalculateSum(int[] array)

{

int sum = 0;

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

{

sum += array[i]; // Суммируем элементы

}

return sum; // Возвращаем сумму

}

}

}

Далее запускаем написанный ранее код в среде разработке для получения результата, результат представлен на рисунке 1.3.2.

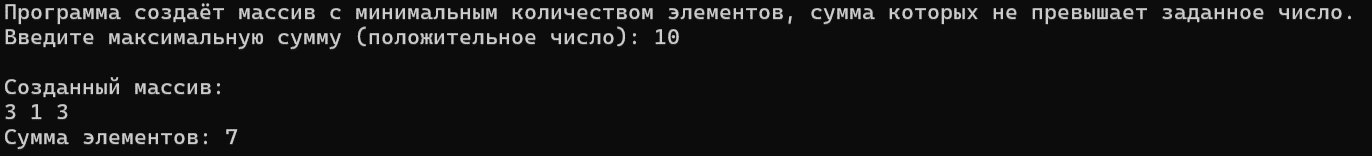


Рисунок 1.3.2 – Результат создания массива

**Задание 3.**

Сформировав квадратную целочисленную матрицу со случайными значениями элементов, упорядочить ее строки по возрастанию сумм их элементов. Пусть элементы принимают значения в диапазоне от –50 до +50.

Ниже представлен листинг 3.3 с кодом программы для создания и дальнейшей работы с квадратной целочисленной матрицей.

Листинг 3.3. Код программы для создания и работы с квадратной целочисленной матрицей.

using System;

namespace MatrixSortBySum

{

class Program

{

static void Main(string[] args)

{

// Выводим приветственное сообщение

Console.WriteLine("Программа формирует прямоугольную матрицу и сортирует её строки по возрастанию сумм элементов.");

// Запрашиваем количество строк

Console.Write("Введите количество строк (положительное число): ");

string inputRows = Console.ReadLine();

int rows;

if (!int.TryParse(inputRows, out rows) || rows <= 0)

{

Console.WriteLine("Ошибка: введите положительное целое число для строк!"); // Сообщение об ошибке

return; // Завершаем программу при некорректном вводе

}

// Запрашиваем количество столбцов

Console.Write("Введите количество столбцов (положительное число): ");

string inputCols = Console.ReadLine();

int cols;

if (!int.TryParse(inputCols, out cols) || cols <= 0)

{

Console.WriteLine("Ошибка: введите положительное целое число для столбцов!"); // Сообщение об ошибке

return; // Завершаем программу при некорректном вводе

}

// Создаём матрицу

int[,] matrix = MatrixHelper.CreateMatrix(rows, cols);

// Выводим исходную матрицу

Console.WriteLine("\nИсходная матрица:");

MatrixHelper.PrintMatrix(matrix);

// Сортируем строки по суммам

MatrixHelper.SortRowsBySum(matrix);

// Выводим отсортированную матрицу

Console.WriteLine("\nМатрица после сортировки строк по суммам:");

MatrixHelper.PrintMatrix(matrix);

}

}

// Класс для статических методов обработки матриц

static class MatrixHelper

{

// Статический метод для создания матрицы со случайными значениями

public static int[,] CreateMatrix(int rows, int cols)

{

Random random = new Random(); // Создаём объект Random

int[,] matrix = new int[rows, cols]; // Создаём матрицу с заданными размерами

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

{

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

{

matrix[i, j] = random.Next(-50, 51); // Генерируем число от -50 до 50

}

}

return matrix; // Возвращаем заполненную матрицу

}

// Статический метод для вывода матрицы

public static void PrintMatrix(int[,] matrix)

{

int rows = matrix.GetLength(0); // Количество строк

int cols = matrix.GetLength(1); // Количество столбцов

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

{

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

{

Console.Write($"{matrix[i, j],4}"); // Выводим с выравниванием (4 символа)

}

Console.WriteLine(); // Переходим на новую строку

}

}

// Статический метод для вычисления суммы строки

private static int CalculateRowSum(int[,] matrix, int row)

{

int sum = 0;

int cols = matrix.GetLength(1); // Количество столбцов

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

{

sum += matrix[row, j]; // Суммируем элементы строки

}

return sum; // Возвращаем сумму

}

// Статический метод для сортировки строк по суммам

public static void SortRowsBySum(int[,] matrix)

{

int rows = matrix.GetLength(0); // Количество строк

int cols = matrix.GetLength(1); // Количество столбцов

int[] sums = new int[rows]; // Массив для хранения сумм строк

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

{

sums[i] = CalculateRowSum(matrix, i); // Вычисляем сумму для каждой строки

}

// Сортируем строки с помощью пузырьковой сортировки

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

{

for (int j = 0; j < rows - 1 - i; j++)

{

if (sums[j] > sums[j + 1])

{

// Меняем суммы

int tempSum = sums[j];

sums[j] = sums[j + 1];

sums[j + 1] = tempSum;

// Меняем строки матрицы

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

{

int tempValue = matrix[j, k];

matrix[j, k] = matrix[j + 1, k];

matrix[j + 1, k] = tempValue;

}

}

}

}

}

}

}

Далее запускаем написанный ранее код в среде разработке для получения результата, результат представлен на рисунке 1.3.3.

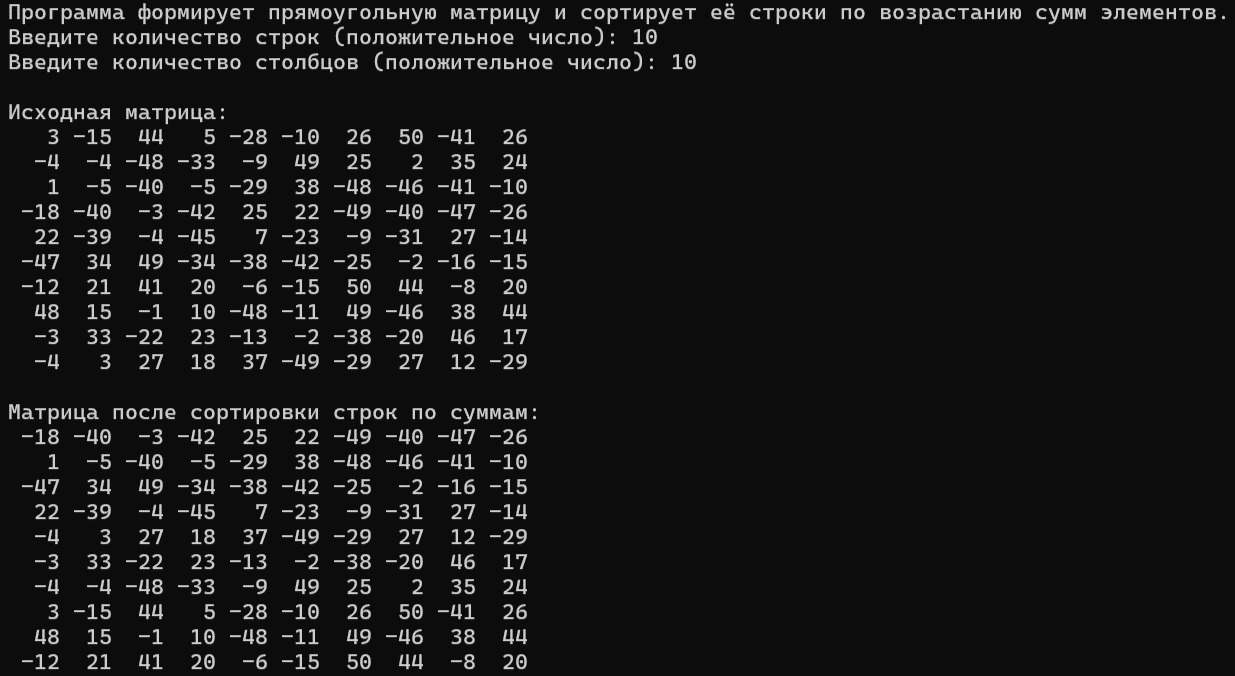


Рисунок 1.3.3 – Результат создания и упорядочения элементов матрицы

**Вывод:** в ходе выполнения практического задания были применены на практике ранее полученные знания о статических методах и методах класса в языке программирования C#.