Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М. Машерова

Отчет

По модулю №1.3

«Статические методы (методы классов)»

по учебной практике по программированию

Выполнил учащийся Войтеховская К.В.

группа 3ПОИС23 29.09.2025 г.

Проверил Алейников М.А.

30.09.2025 г.

Орша, 2025

**Ход работы.**

**Выполнение практических заданий.**

**Выполнение задания 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

В листинге 1 для решения задачи создан статический метод NOD с параметрами a и b для вычисления наибольшего общего делителя. В методе использован алгоритм с циклом while и оператором % для вычисления остатка от деления. Использованы операторы if для проверки корректности введенных данных (неотрицательный числитель и положительный знаменатель). Для сокращения дроби использовано целочисленное деление числителя и знаменателя на вычисленный НОД. Применены операторы if-else для вывода соответствующего сообщения в зависимости от того, была ли дробь сокращена.

Листинг 1. Код программы для поиска НОД

using System;

class Program

{

// Статический метод для вычисления НОД

static int NOD(int a, int b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

static void Main()

{

Console.WriteLine("Введите неотрицательную обыкновенную дробь:");

Console.Write("Числитель: ");

int numerator = int.Parse(Console.ReadLine());

Console.Write("Знаменатель: ");

int denominator = int.Parse(Console.ReadLine());

if (denominator <= 0)

{

Console.WriteLine("Ошибка: знаменатель должен быть положительным числом!");

return;

}

if (numerator < 0)

{

Console.WriteLine("Ошибка: числитель должен быть неотрицательным!");

return;

}

// Вычисляем НОД числителя и знаменателя

int gcd = NOD(numerator, denominator);

// Сокращаем дробь

int reducedNumerator = numerator / gcd;

int reducedDenominator = denominator / gcd;

// Выводим результат

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

Console.WriteLine($"НОД({numerator}, {denominator}) = {gcd}");

if (gcd == 1)

{

Console.WriteLine("Дробь уже несократима");

}

else

{

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

}

}

}

Ссылка на GitHub: https://github.com/Ksenia1912/yi.san.yi

На рисунке 1 представлен результат выполнения программы.

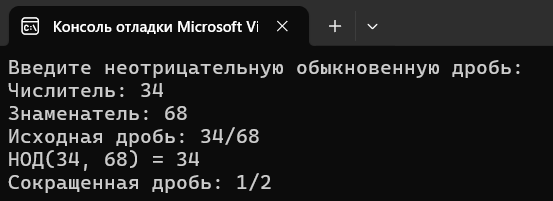


Рисунок 1 – Результат сокращении дроби

**Выполнение задания 2.** Присваивая последовательным элементам массива случайные значения от 1 до 9, создать массив с минимальным количеством элементов, сумма которых не превышает заданного пользователем числа.

В листинге 2 для решения задачи был использован список List<int> для динамического формирования массива. Применён цикл while с условием currentSum <= maxSum для добавления элементов. В цикле использована переменная potentialSum для проверки превышения максимальной суммы и оператор if с break для выхода из цикла. Для вывода результата применён цикл for с условным оператором if для корректного отображения знаков "+" между элементами. Выведена итоговая сумма созданного массива.

Листинг 2. Код программы для создания массива

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

Console.Write("Введите максимальную сумму элементов: ");

int maxSum = int.Parse(Console.ReadLine());

Random random = new Random();

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

int currentSum = 0;

// Добавление элементов, пока сумма не превысит максимальную

while (currentSum <= maxSum)

{

int newElement = random.Next(1, 10); // случайное число от 1 до 9

int potentialSum = currentSum + newElement;

// Если добавление нового элемента превысит максимальную сумму, остановка

if (potentialSum > maxSum)

break;

array.Add(newElement);

currentSum = potentialSum;

}

Console.Write("Созданный массив: ");

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

{

Console.Write(array[i]);

if (i < array.Count - 1)

Console.Write(" + ");

}

Console.WriteLine($" = {currentSum}");

}

Ссылка:https://github.com/Ksenia1912/yi.san.er

На рисунке 2 представлен результат выполнения программы.

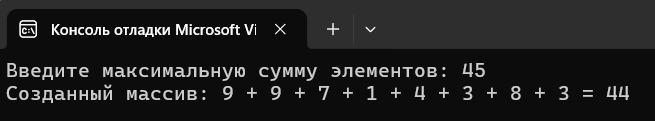


Рисунок 2 – Вывод массива сумма которого не превышает введённое число

**Выполнение задания 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.

В листинге 3 для решения задачи для вычисления сумм строк создан массив rowSums. Использованы вложенные циклы for для подсчёта сумм элементов каждой строки. Применён алгоритм пузырьковой сортировки с вложенными циклами for для упорядочивания строк по возрастанию сумм. В процессе сортировки использован оператор if для сравнения сумм строк rowSums[j] > rowSums[j + 1]. Применён дополнительный цикл for для обмена элементами строк матрицы. Для вывода результатов использованы вложенные циклы for с методом PadLeft(4) для форматированного отображения чисел.

Листинг 3. Код программы для сортировки массива по суммам строк

using System;

class Program

{

static void Main()

{

Console.Write("Введите размер квадратной матрицы: ");

int size = int.Parse(Console.ReadLine());

if (size <= 0)

{

Console.WriteLine("Размер матрицы должен быть положительным!");

return;

}

Random random = new Random();

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

// Заполнение матрицы случайными числами от -50 до 50

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

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

{

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

{

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

Console.Write(matrix[i, j].ToString().PadLeft(4));

}

Console.WriteLine();

}

// Вычисление сумм строк

int[] rowSums = new int[size];

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

{

int sum = 0;

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

{

sum += matrix[i, j];

}

rowSums[i] = sum; }

// Вывод сумм строк

Console.WriteLine("\nСуммы строк:");

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

{

Console.WriteLine($"Строка {i}: {rowSums[i]}");

}

// Сортировка строк по возрастанию суммы (пузырьковая сортировка)

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

{

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

{

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

{

// Обмен сумм

int tempSum = rowSums[j];

rowSums[j] = rowSums[j + 1];

rowSums[j + 1] = tempSum;

// Обмен строк в матрице

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

{

int temp = matrix[j, k];

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

matrix[j + 1, k] = temp;

}

}

}

}

// Вывод отсортированной матрицы

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

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

{

int currentSum = 0;

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

{

Console.Write(matrix[i, j].ToString().PadLeft(4));

currentSum += matrix[i, j];

}

Console.WriteLine($" ");

}

}

}

Ссылка: https://github.com/Ksenia1912/yi.er.san

На рисунке 3 представлен результат выполнения программы.

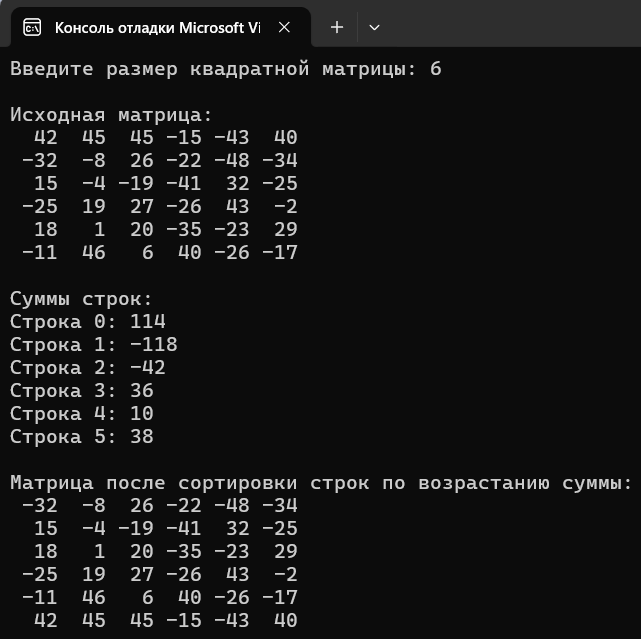


Рисунок 3 – Вывод массива с отсортированными строками