Министерство образования Республики Беларусь

Оршанский колледж учреждения образования

«Витебский государственный университет имени П. М. Машерова»

Специальность 5-04-0612-02

«Разработка и сопровождение программного обеспечения информационных систем»

Группа 3ПОИС23

УЧЕБНАЯ ПРАКТИКА

ПО ПРОГРАММИРОВАНИЮ

с 29 сентября 2025 года по 25 октября 2025 года

ОТЧЕТ

Выполнил Трубочкин И.А.

Руководитель практики Алейников М.А.

2025

**СОДЕРЖАНИЕ**

## Модуль 1.1 Основы C#

## Модуль 2. Объектно-ориентированное программирование (ООП)

## Модуль 3. Задания по делегатам

## Модуль 4. Задания по интерфейсам

## Модуль 5. Графический интерфейс

## Модуль 6. Базы данных

## Модуль 7. Проектная работа

## Модуль 8. Реализация проекта

## Постановка задачи

## Описание предметной области

## Характеристика решаемой задачи

## Разработка моделей решаемой задачи

## Проектирование программного модуля

## Сбор исходных материалов

## Проектирование информационной модели

## Описание входных и выходных данных

## Реализация программного модуля

## Описание диаграммы классов разрабатываемого проекта

## Описание структуры разрабатываемого проекта

## Проектирование и реализация интерфейса программы

## Модуль 9. Совершенствование навыков ООП

## ЗАКЛЮЧЕНИЕ

## Список использованных источников

## Приложение А Текст программы модуля 8

## Модуль 1.1 Основы C#

Вариант 1

Выполнение:

Задание 1. Реализуйте приложение для вычисления факториала числа.

Листинг 1. Код программы на C#

using System;

class SimpleFactorial

{

static void Main()

{

while (true)

{

Console.Write("\nВведите число: ");

string input = Console.ReadLine();

if (input == "exit")

break;

if (int.TryParse(input, out int number))

{

if (number < 0)

{

Console.WriteLine("Факториал отрицательного числа не существует!");

}

else if (number > 20)

{

Console.WriteLine("Попробуйте число до 20.");

}

else

{

long factorial = 1;

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

{

factorial \*= i;

}

Console.WriteLine($"Факториал {number} = {factorial}");

}

}

else

{

Console.WriteLine("Пожалуйста, введите целое число!");

}

}

}

}

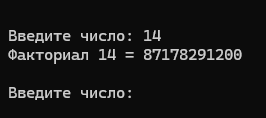


Рисунок 1.1.1 – Результат выполнения программы листинга 1

Задание 2. Напишите программу, которая запрашивает у пользователя два целых числа и выводит их сумму.

Листинг 2. Код программы на C#

using System;

class Program

{

static void Main()

{

Console.Write("Введите первое число: ");

string input1 = Console.ReadLine();

Console.Write("Введите второе число: ");

string input2 = Console.ReadLine();

if (int.TryParse(input1, out int number1) && int.TryParse(input2, out int number2))

{

int sum = number1 + number2;

Console.WriteLine($"\nРезультат: {number1} + {number2} = {sum}");

}

else

{

Console.WriteLine("\nОшибка: Пожалуйста, введите корректные целые числа!");

}

Console.WriteLine("\nНажмите любую клавишу для выхода...");

Console.ReadKey();

}

}

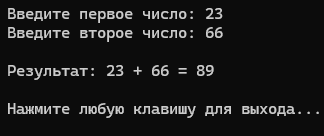


Рисунок 1.1.2 – Результат выполнения программы листинга 2

Задание 3. Напишите программу, которая принимает на вход строку и выводит ее в обратном порядке.

Листинг 3. Код программы на C#

using System;

class Program

{

static void Main()

{

Console.Write("Введите строку: ");

string input = Console.ReadLine();

if (string.IsNullOrEmpty(input))

{

Console.WriteLine("Вы ввели пустую строку!");

return;

}

string reversed = ReverseString(input);

Console.WriteLine($"\nОригинал: {input}");

Console.WriteLine($"Реверс: {reversed}");

}

static string ReverseString(string str)

{

char[] charArray = str.ToCharArray();

Array.Reverse(charArray);

return new string(charArray);

}

}

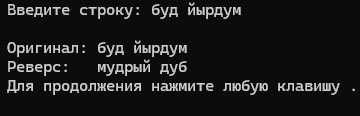


Рисунок 1.1.3 – Результат выполнения программы листинга 3

Задание 4. Напишите программу, которая создает массив из 15 случайных чисел и находит среднее значение всех положительных чисел в массиве.

Листинг 4. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int[] numbers = GenerateRandomArray(15, -100, 100);

PrintArray("Сгенерированный массив", numbers);

int[] positiveNumbers = numbers.Where(n => n > 0).ToArray();

PrintArray("Положительные числа", positiveNumbers);

if (positiveNumbers.Length > 0)

{

double average = positiveNumbers.Average();

Console.WriteLine($"\n Результаты:");

Console.WriteLine($"Количество положительных чисел: {positiveNumbers.Length}");

Console.WriteLine($"Сумма положительных чисел: {positiveNumbers.Sum()}");

Console.WriteLine($"Среднее значение: {average:F2}");

}

else

{

Console.WriteLine("\n В массиве нет положительных чисел!");

}

}

static int[] GenerateRandomArray(int size, int minValue, int maxValue)

{

Random random = new Random();

int[] array = new int[size];

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

{

array[i] = random.Next(minValue, maxValue + 1);

}

return array;

}

static void PrintArray(string title, int[] array)

{

Console.WriteLine($"\n{title}:");

Console.WriteLine(string.Join(", ", array));

}

}

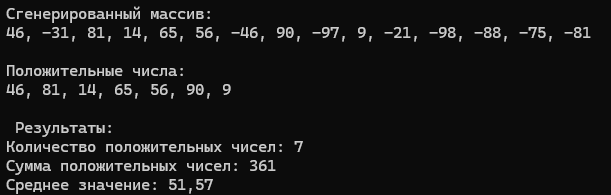


Рисунок 1.1.4 – Результат выполнения программы листинга 4

Задание 5. Напишите программу, которая проверяет, является ли введенное пользователем число простым (не имеет делителей, кроме 1 и самого себя).

Листинг 5. Код программы на C#

using System;

class Program

{

static void Main()

{

while (true)

{

Console.Write("Введите число: ");

string input = Console.ReadLine();

if (input?.ToLower() == "exit")

break;

if (int.TryParse(input, out int number))

{

if (number < 2)

{

Console.WriteLine($" Число {number} не является простым (простые числа ≥ 2)");

}

else if (IsPrime(number))

{

Console.WriteLine($" Число {number} - Простое");

}

else

{

Console.WriteLine($" Число {number} - Составное");

}

}

else

{

Console.WriteLine(" Ошибка: Введите целое число!");

}

Console.WriteLine();

}

}

static bool IsPrime(int n)

{

if (n == 2 || n == 3)

return true;

if (n % 2 == 0 || n % 3 == 0)

return false;

for (int i = 5; i \* i <= n; i += 6)

{

if (n % i == 0 || n % (i + 2) == 0)

return false;

}

return true;

}

}

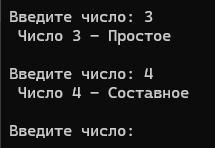


Рисунок 1.1.5 – Результат выполнения программы листинга 5

Модуль 1.2 Массивы, Строки, Переключатели

Задача 1. Ввести размер массива N и значения его элементов. Нормировать элементы массива, разделив их на значение максимального по модулю элемента. Вывести значения элементов измененного массива.

Выполнение:

Листинг 1. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int n;

while (true)

{

Console.Write("Введите размер массива N: ");

if (int.TryParse(Console.ReadLine(), out n) && n > 0)

{

break;

}

Console.WriteLine("Ошибка: введите целое положительное число!");

}

double[] array = new double[n];

Console.WriteLine($"Введите {n} элементов массива:");

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

{

while (true)

{

Console.Write($"Элемент [{i}]: ");

if (double.TryParse(Console.ReadLine(), out array[i]))

{

break;

}

Console.WriteLine("Ошибка: введите число!");

}

}

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

PrintArray(array);

double[] normalizedArray = NormalizeArray(array);

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

PrintArray(normalizedArray);

Console.WriteLine($"\nМаксимальный по модулю элемент: {FindMaxAbsolute(array)}");

}

static double[] NormalizeArray(double[] array)

{

double maxAbsolute = FindMaxAbsolute(array);

if (maxAbsolute == 0)

return (double[])array.Clone();

double[] result = new double[array.Length];

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

{

result[i] = array[i] / maxAbsolute;

}

return result;

}

static double FindMaxAbsolute(double[] array)

{

double max = 0;

foreach (double element in array)

{

double absolute = Math.Abs(element);

if (absolute > max)

max = absolute;

}

return max;

}

static void PrintArray(double[] array)

{

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

{

Console.WriteLine($"[{i}] = {array[i]:F6}");

}

}

}

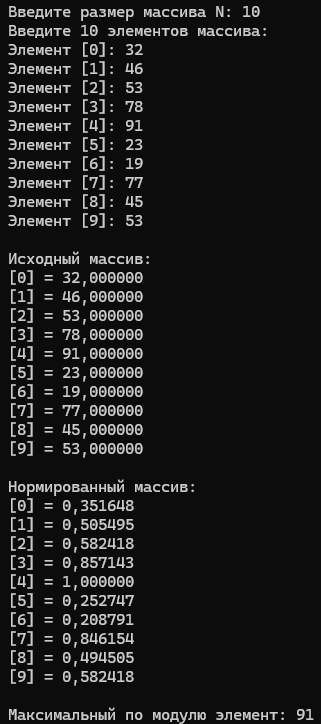


Рисунок 1.2.1 – Результат выполнения программы листинга 1

Задание 2. Определить и инициализировать целочисленный массив из 10-ти элементов. Ввести целое число и заменить им значение максимального элемента в массиве.

Выполнение:

Листинг 2. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int[] array = { 15, 7, 23, 9, 42, 18, 31, 5, 27, 12 };

Console.WriteLine("Исходный массив:");

PrintArray(array);

int maxValue = array.Max();

int maxIndex = Array.IndexOf(array, maxValue);

Console.WriteLine($"\nМаксимальный элемент: array[{maxIndex}] = {maxValue}");

int newNumber;

while (true)

{

Console.Write("\nВведите целое число для замены: ");

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

{

break;

}

Console.WriteLine("Ошибка: введите целое число!");

}

array[maxIndex] = newNumber;

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

PrintArray(array);

Console.WriteLine($"\nМаксимальный элемент заменен на: {newNumber}");

}

static void PrintArray(int[] array)

{

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

{

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

}

}

}

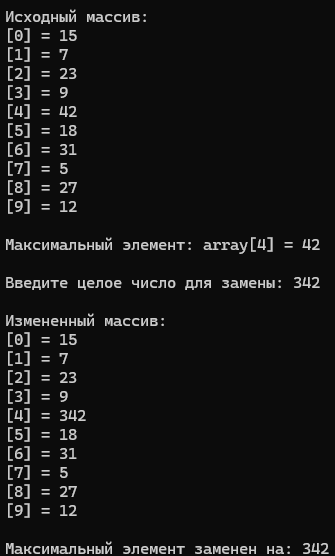


Рисунок 1.2.2 – Результат выполнения программы листинга 2

Задание 3. Вычислить К простых чисел. Значение К ввести с клавиатуры. Вывести значения чисел, размещая их по 10 на строке.

Выполнение:

Листинг 3. Код программы на C#

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

int k;

while (true)

{

Console.Write("Введите количество простых чисел K: ");

if (int.TryParse(Console.ReadLine(), out k) && k > 0)

{

break;

}

Console.WriteLine("Ошибка: введите целое положительное число!");

}

List<int> primes = GeneratePrimes(k);

Console.WriteLine($"\nПервые {k} простых чисел:");

PrintPrimesFormatted(primes);

Console.WriteLine($"\n Статистика:");

Console.WriteLine($"Последнее простое число: {primes[primes.Count - 1]}");

Console.WriteLine($"Сумма всех чисел: {SumPrimes(primes):N0}");

}

static List<int> GeneratePrimes(int k)

{

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

int number = 2;

while (primes.Count < k)

{

if (IsPrime(number))

{

primes.Add(number);

}

number++;

}

return primes;

}

static bool IsPrime(int n)

{

if (n < 2) return false;

if (n == 2) return true;

if (n % 2 == 0) return false;

for (int i = 3; i \* i <= n; i += 2)

{

if (n % i == 0)

return false;

}

return true;

}

static void PrintPrimesFormatted(List<int> primes)

{

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

{

Console.Write($"{primes[i],6}");

if ((i + 1) % 10 == 0)

Console.WriteLine();

}

if (primes.Count % 10 != 0)

Console.WriteLine();

}

static long SumPrimes(List<int> primes)

{

long sum = 0;

foreach (int prime in primes)

{

sum += prime;

}

return sum;

}

}

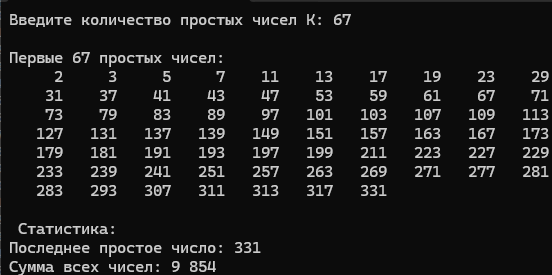


Рисунок 1.2.3 – Результат выполнения программы листинга 3

Задание 4. Определить целочисленный массив из К элементов. Присвоить элементам случайные значения из диапазона [А,В). Найти индексы минимального и максимального элементов массива. Вывести значения элементов, расположенных между найденными (включая найденные)

Выполнение:

Листинг 4. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int k = GetPositiveInteger("Введите размер массива K: ");

int a = GetInteger("Введите начало диапазона A: ");

int b = GetInteger("Введите конец диапазона B: ");

if (b <= a)

{

Console.WriteLine("Ошибка: B должно быть больше A!");

return;

}

int[] array = GenerateRandomArray(k, a, b);

Console.WriteLine("\n Исходный массив:");

PrintArray(array);

int minIndex = FindMinIndex(array);

int maxIndex = FindMaxIndex(array);

Console.WriteLine($"\n Минимальный элемент: array[{minIndex}] = {array[minIndex]}");

Console.WriteLine($" Максимальный элемент: array[{maxIndex}] = {array[maxIndex]}");

PrintElementsBetween(array, minIndex, maxIndex);

}

static int GetPositiveInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int n) && n > 0)

return n;

Console.WriteLine(" Ошибка: введите целое положительное число!");

}

}

static int GetInteger(string message)

{

while (true)

{

Console.Write(message);

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

return n;

Console.WriteLine(" Ошибка: введите целое число!");

}

}

static int[] GenerateRandomArray(int size, int min, int max)

{

Random random = new Random();

int[] array = new int[size];

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

{

array[i] = random.Next(min, max);

}

return array;

}

static int FindMinIndex(int[] array)

{

int minIndex = 0;

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

{

if (array[i] < array[minIndex])

minIndex = i;

}

return minIndex;

}

static int FindMaxIndex(int[] array)

{

int maxIndex = 0;

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

{

if (array[i] > array[maxIndex])

maxIndex = i;

}

return maxIndex;

}

static void PrintArray(int[] array)

{

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

{

Console.WriteLine($"[{i,2}] = {array[i],4}");

}

}

static void PrintElementsBetween(int[] array, int index1, int index2)

{

int start = Math.Min(index1, index2);

int end = Math.Max(index1, index2);

Console.WriteLine($"\n Элементы между индексами {start} и {end}:");

for (int i = start; i <= end; i++)

{

Console.WriteLine($"[{i,2}] = {array[i],4}");

}

Console.WriteLine($"\n Количество элементов: {end - start + 1}");

Console.WriteLine($" Диапазон значений: от {array[start]} до {array[end]}");

}

}

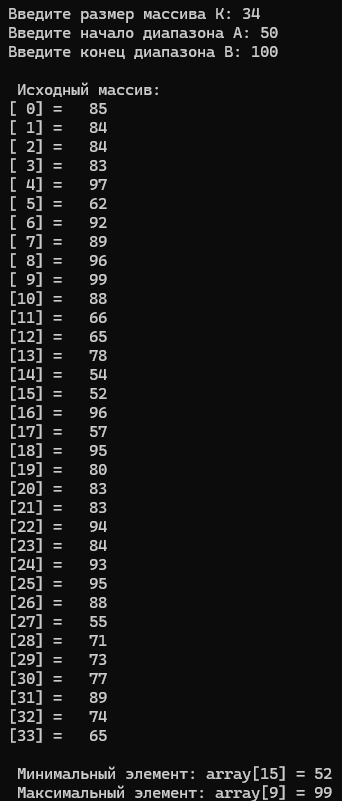


Рисунок 1.2.5 – Результат выполнения программы листинга 4

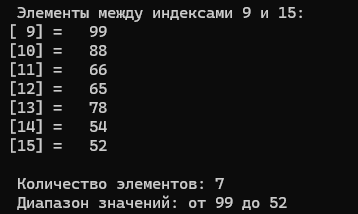


Рисунок 1.2.6 – Результат выполнения программы листинга 4

Задание 5. Определить символьный массив из К элементов. Присвоить элементам случайные значения букв русского алфавита. Создать новый массив, поместив в него только согласные буквы из первого массива. Значение К ввести с клавиатуры. Вывести элементы обоих массивов.

Выполнение:

Листинг 5. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int k = GetPositiveInteger("Введите размер массива K: ");

char[] originalArray = GRussianArray(k);

char[] consonantsArray = FilterConsonants(originalArray);

Console.WriteLine("\n Исходный массив:");

PrintCharArray(originalArray);

Console.WriteLine("\n Массив согласных букв:");

PrintCharArray(consonantsArray);

Console.WriteLine($"\n Статистика:");

Console.WriteLine($"Всего элементов: {originalArray.Length}");

Console.WriteLine($"Согласных букв: {consonantsArray.Length}");

Console.WriteLine($"Гласных букв: {CountVowels(originalArray)}");

}

static int GetPositiveInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int n) && n > 0)

return n;

Console.WriteLine(" Ошибка: введите целое положительное число!");

}

}

static char[] GRussianArray(int size)

{

char[] russianAlphabet = {

'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М',

'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ',

'Ы', 'Ь', 'Э', 'Ю', 'Я',

'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',

'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ',

'ы', 'ь', 'э', 'ю', 'я'

};

Random random = new Random();

char[] array = new char[size];

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

{

array[i] = russianAlphabet[random.Next(russianAlphabet.Length)];

}

return array;

}

static char[] FilterConsonants(char[] array)

{

char[] vowels = { 'А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я',

'а', 'е', 'ё', 'и', 'о', 'у', 'ы', 'э', 'ю', 'я' };

return array.Where(c => !vowels.Contains(c)).ToArray();

}

static void PrintCharArray(char[] array)

{

if (array.Length == 0)

{

Console.WriteLine("Массив пуст");

return;

}

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

{

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

if ((i + 1) % 15 == 0)

Console.WriteLine();

}

if (array.Length % 15 != 0)

Console.WriteLine();

}

static int CountVowels(char[] array)

{

char[] vowels = { 'А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я',

'а', 'е', 'ё', 'и', 'о', 'у', 'ы', 'э', 'ю', 'я' };

return array.Count(c => vowels.Contains(c));

}

}

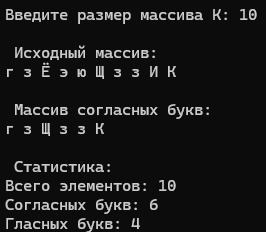


Рисунок 1.2.7 – Результат выполнения программы листинга 5

Задание 6. Определить вещественный массив из 10-ти элементов. Присвоить элементам случайные значения из диапазона [–10, 10). Сформировать массив индексов, которые нумеруют элементы первого массива в порядке возрастания их значений.

Выполнение:

Листинг 6. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

double[] array = GenerateRandomArray(10, -10, 10);

Console.WriteLine("\n Исходный массив:");

PrintArray(array);

int[] sortedIndices = GetSortedIndices(array);

Console.WriteLine("\n Отсортированные индексы по возрастанию значений:");

PrintSortedResults(array, sortedIndices);

}

static double[] GenerateRandomArray(int size, double min, double max)

{

Random random = new Random();

double[] array = new double[size];

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

{

array[i] = min + (random.NextDouble() \* (max - min));

}

return array;

}

static int[] GetSortedIndices(double[] array)

{

int[] indices = new int[array.Length];

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

{

indices[i] = i;

}

Array.Sort(indices, (a, b) => array[a].CompareTo(array[b]));

return indices;

}

static void PrintArray(double[] array)

{

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

{

Console.WriteLine($"[{i,2}] = {array[i],8:F3}");

}

}

static void PrintSortedResults(double[] array, int[] sortedIndices)

{

Console.WriteLine("Индекс | Значение");

Console.WriteLine("-------------------");

foreach (int index in sortedIndices)

{

Console.WriteLine($"{index,6} | {array[index],8:F3}");

}

}

}

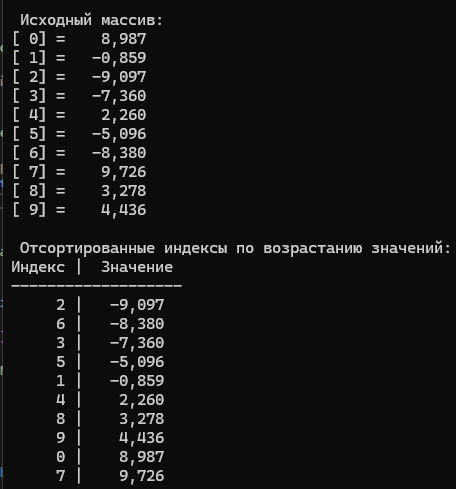


Рисунок 1.2.8 – Результат выполнения программы листинга 6

Модуль 1.3 Статические методы (Методы классов)

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.

Листинг 1. Код программы на C#

using System;

class Program

{

static void Main()

{

int numerator = GetNonNegativeInteger("Введите неотрицательный числитель: ");

int denominator = GetPositiveInteger("Введите положительный знаменатель: ");

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

(int simplifiedNumerator, int simplifiedDenominator) = SimplifyFraction(numerator, denominator);

Console.WriteLine($"Сокращенная дробь: {simplifiedNumerator}/{simplifiedDenominator}");

if (simplifiedNumerator >= simplifiedDenominator && simplifiedDenominator != 1)

{

int wholePart = simplifiedNumerator / simplifiedDenominator;

int remainder = simplifiedNumerator % simplifiedDenominator;

if (remainder != 0)

{

Console.WriteLine($"В виде смешанного числа: {wholePart} {remainder}/{simplifiedDenominator}");

}

else

{

Console.WriteLine($"Целое число: {wholePart}");

}

}

double decimalValue = (double)numerator / denominator;

Console.WriteLine($"Десятичное представление: {decimalValue:F6}");

}

static int GCD(int a, int b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

static (int numerator, int denominator) SimplifyFraction(int numerator, int denominator)

{

if (numerator == 0)

return (0, 1);

int gcd = GCD(numerator, denominator);

int simplifiedNumerator = numerator / gcd;

int simplifiedDenominator = denominator / gcd;

return (simplifiedNumerator, simplifiedDenominator);

}

static int GetNonNegativeInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int number) && number >= 0)

return number;

Console.WriteLine("Ошибка: введите неотрицательное целое число!");

}

}

static int GetPositiveInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int number) && number > 0)

return number;

Console.WriteLine("Ошибка: введите положительное целое число!");

}

}

}

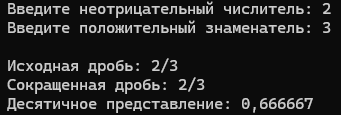


Рисунок 1.3.1 – Результат выполнения программы листинга 1

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.

Листинг 2. Код программы на C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.ConstrainedExecution;

class Program

{

static void Main()

{

int targetSum = GetPositiveInteger("Введите максимальную сумму элементов: ");

var result = GenerateArrayWithSumLimit(targetSum);

int[] array = result.Item1;

int actualSum = result.Item2;

int count = result.Item3;

Console.WriteLine("\nРезультат:");

Console.WriteLine("Количество элементов: " + count);

Console.WriteLine("Сумма элементов: " + actualSum);

Console.WriteLine("Целевая сумма: " + targetSum);

Console.WriteLine("Разница: " + (targetSum - actualSum));

PrintArray(array, count);

}

static Tuple<int[], int, int> GenerateArrayWithSumLimit(int targetSum)

{

Random random = new Random();

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

int currentSum = 0;

int count = 0;

while (currentSum < targetSum)

{

int randomValue = random.Next(1, 10);

int potentialSum = currentSum + randomValue;

if (potentialSum > targetSum)

break;

elements.Add(randomValue);

currentSum = potentialSum;

count++;

}

return Tuple.Create(elements.ToArray(), currentSum, count);

}

static void PrintArray(int[] array, int count)

{

Console.WriteLine("\nМассив элементов:");

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

{

Console.Write(array[i] + " ");

if ((i + 1) % 15 == 0)

Console.WriteLine();

}

Console.WriteLine();

var subArray = new List<int>();

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

{

subArray.Add(array[i]);

}

Console.Write("Сумма: " + string.Join(" + ", subArray) + " = " + subArray.Sum());

Console.WriteLine();

}

static int GetPositiveInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int number) && number > 0)

return number;

Console.WriteLine("Ошибка: введите положительное целое число!");

}

}

}

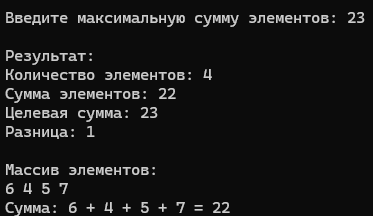


Рисунок 1.3.2 – Результат выполнения программы листинга 2

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.

Листинг 3. Код программы на C#

using System;

using System.Linq;

class Program

{

static void Main()

{

int size = GetPositiveInteger("Введите размер квадратной матрицы: ");

int[,] matrix = GenerateMatrix(size, -50, 50);

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

PrintMatrix(matrix);

int[] rowSums = CalculateRowSums(matrix);

Console.WriteLine("\nСуммы строк исходной матрицы:");

PrintRowSums(rowSums);

int[,] sortedMatrix = SortMatrixByRowSums(matrix);

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

PrintMatrix(sortedMatrix);

int[] sortedRowSums = CalculateRowSums(sortedMatrix);

Console.WriteLine("\nСуммы строк отсортированной матрицы:");

PrintRowSums(sortedRowSums);

PrintStatistics(matrix, sortedMatrix);

}

static int[,] GenerateMatrix(int size, int minValue, int maxValue)

{

Random random = new Random();

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

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

{

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

{

matrix[i, j] = random.Next(minValue, maxValue + 1);

}

}

return matrix;

}

static int[] CalculateRowSums(int[,] matrix)

{

int size = matrix.GetLength(0);

int[] sums = new int[size];

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

{

int sum = 0;

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

{

sum += matrix[i, j];

}

sums[i] = sum;

}

return sums;

}

static int[,] SortMatrixByRowSums(int[,] matrix)

{

int size = matrix.GetLength(0);

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

int[] rowSums = CalculateRowSums(matrix);

int[] indices = new int[size];

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

{

indices[i] = i;

}

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

{

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

{

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

{

int temp = indices[j];

indices[j] = indices[j + 1];

indices[j + 1] = temp;

}

}

}

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

{

int originalRow = indices[i];

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

{

sortedMatrix[i, j] = matrix[originalRow, j];

}

}

return sortedMatrix;

}

static void PrintMatrix(int[,] matrix)

{

int size = matrix.GetLength(0);

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

{

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

{

Console.Write($"{matrix[i, j],5}");

}

Console.WriteLine();

}

}

static void PrintRowSums(int[] rowSums)

{

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

{

Console.WriteLine($"Строка {i}: {rowSums[i],6}");

}

}

static void PrintStatistics(int[,] originalMatrix, int[,] sortedMatrix)

{

int size = originalMatrix.GetLength(0);

int[] originalSums = CalculateRowSums(originalMatrix);

int[] sortedSums = CalculateRowSums(sortedMatrix);

int positiveSums = sortedSums.Count(sum => sum > 0);

int negativeSums = sortedSums.Count(sum => sum < 0);

int zeroSums = sortedSums.Count(sum => sum == 0);

}

static int GetPositiveInteger(string message)

{

while (true)

{

Console.Write(message);

if (int.TryParse(Console.ReadLine(), out int number) && number > 0)

return number;

Console.WriteLine("Ошибка: введите положительное целое число!");

}

}

}

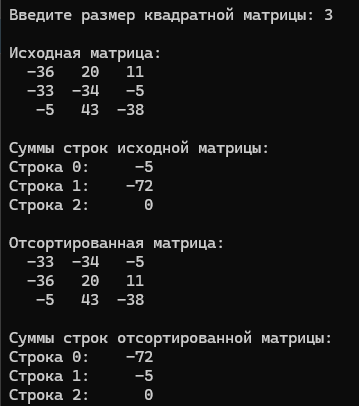


Рисунок 1.3.3 – Результат выполнения программы листинга 3

Модуль 2

Задание 1. Создайте класс Person, представляющий человека. У этого класса должны быть поля для хранения имени, возраста и адреса. Добавьте методы для установки и получения значений этих полей. Затем создайте объекты этого класса и выведите информацию о них.

Листинг 1. Код программы на C#

using System;

public class Person

{

private string name;

private int age;

private string address;

public Person()

{

name = "Неизвестно";

age = 0;

address = "Не указан";

}

public Person(string name, int age, string address)

{

this.name = name;

this.age = age;

this.address = address;

}

public string GetName()

{

return name;

}

public int GetAge()

{

return age;

}

public string GetAddress()

{

return address;

}

public void SetName(string name)

{

if (!string.IsNullOrWhiteSpace(name))

{

this.name = name;

}

}

public void SetAge(int age)

{

if (age >= 0 && age <= 150)

{

this.age = age;

}

}

public void SetAddress(string address)

{

if (!string.IsNullOrWhiteSpace(address))

{

this.address = address;

}

}

public void DisplayInfo()

{

Console.WriteLine($"Имя: {name}");

Console.WriteLine($"Возраст: {age} лет");

Console.WriteLine($"Адрес: {address}");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

Person person1 = new Person();

person1.SetName("Иван Петров");

person1.SetAge(25);

person1.SetAddress("г. Москва, ул. Ленина, д. 10");

Person person2 = new Person("Мария Сидорова", 30, "г. Санкт-Петербург, Невский пр., д. 25");

Person person3 = new Person();

person3.SetName("Алексей Иванов");

person3.SetAge(35);

person3.SetAddress("г. Екатеринбург, ул. Мира, д. 15");

person1.DisplayInfo();

person2.DisplayInfo();

person3.DisplayInfo();

Console.WriteLine($"Имя первого человека: {person1.GetName()}");

Console.WriteLine($"Возраст второго человека: {person2.GetAge()} лет");

Console.WriteLine($"Адрес третьего человека: {person3.GetAddress()}");

Console.ReadLine();

}

}

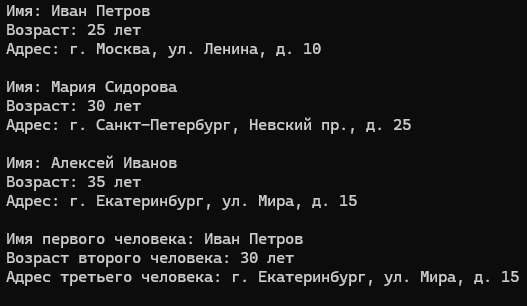


Рисунок 2.1.1 – Результат выполнения программы листинга 1

Задача 2: Наследование и полиморфизм

Создайте базовый класс Shape, представляющий геометрическую фигуру, и производные классы Circle и Rectangle. В базовом классе определите метод Area(), который возвращает площадь фигуры, и метод Perimeter(), который возвращает периметр фигуры. В производных классах переопределите эти методы для соответствующих фигур (круг и прямоугольник). Создайте объекты всех классов и выведите их площади и периметры.

Листинг 2. Код программы на C#

using System;

public abstract class Shape

{

public abstract double Area();

public abstract double Perimeter();

public virtual void DisplayInfo()

{

Console.WriteLine($"Площадь: {Area():F2}");

Console.WriteLine($"Периметр: {Perimeter():F2}");

}

}

// Производный класс Circle (круг)

public class Circle : Shape

{

public double Radius { get; set; }

// Конструктор

public Circle(double radius)

{

if (radius <= 0)

throw new ArgumentException("Радиус должен быть положительным числом");

Radius = radius;

}

// Переопределение метода Area для круга

public override double Area()

{

return Math.PI \* Radius \* Radius;

}

// Переопределение метода Perimeter для круга

public override double Perimeter()

{

return 2 \* Math.PI \* Radius;

}

// Переопределение метода DisplayInfo

public override void DisplayInfo()

{

Console.WriteLine($"Круг с радиусом {Radius:F2}");

base.DisplayInfo();

Console.WriteLine();

}

}

public class Rectangle : Shape

{

public double Width { get; set; }

public double Height { get; set; }

// Конструктор

public Rectangle(double width, double height)

{

if (width <= 0 || height <= 0)

throw new ArgumentException("Ширина и высота должны быть положительными числами");

Width = width;

Height = height;

}

// Переопределение метода Area для прямоугольника

public override double Area()

{

return Width \* Height;

}

// Переопределение метода Perimeter для прямоугольника

public override double Perimeter()

{

return 2 \* (Width + Height);

}

// Переопределение метода DisplayInfo

public override void DisplayInfo()

{

Console.WriteLine($"Прямоугольник {Width:F2} x {Height:F2}");

base.DisplayInfo();

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

try

{

// Круги

Circle circle1 = new Circle(5.0);

Circle circle2 = new Circle(3.5);

// Прямоугольники

Rectangle rectangle1 = new Rectangle(4.0, 6.0);

Rectangle rectangle2 = new Rectangle(2.5, 8.0);

Rectangle square = new Rectangle(5.0, 5.0); // Квадрат

// Вывод информации о фигурах

Console.WriteLine("Информация о фигурах:");

Console.WriteLine("---------------------");

circle1.DisplayInfo();

circle2.DisplayInfo();

rectangle1.DisplayInfo();

rectangle2.DisplayInfo();

square.DisplayInfo();

Shape[] shapes = new Shape[]

{

new Circle(7.0),

new Rectangle(3.0, 4.0),

new Circle(2.0),

new Rectangle(6.0, 8.0)

};

double totalArea = 0;

double totalPerimeter = 0;

foreach (Shape shape in shapes)

{

shape.DisplayInfo();

totalArea += shape.Area();

totalPerimeter += shape.Perimeter();

}

Console.WriteLine($"Общая площадь всех фигур: {totalArea:F2}");

Console.WriteLine($"Общий периметр всех фигур: {totalPerimeter:F2}");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Console.ReadLine();

}

}

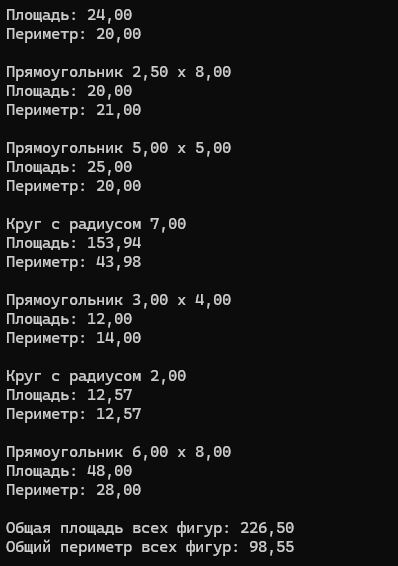


Рисунок 2.2.1 – Результат выполнения программы листинга 2

Задача 3.

Создайте классы Author и Book. Класс Author должен содержать информацию об авторе (имя и год рождения). Класс Book должен содержать информацию о книге (название, год выпуска и автора). Используйте композицию, чтобы связать объекты Author и Book. Создайте несколько объектов Author и Book и выведите информацию о них.

Листинг 3. Код программы на C#

using System;

using System.Collections.Generic;

public class Author

{

public string Name { get; set; }

public int BirthYear { get; set; }

public Author()

{

Name = "Неизвестный автор";

BirthYear = DateTime.Now.Year;

}

// Конструктор с параметрами

public Author(string name, int birthYear)

{

if (string.IsNullOrWhiteSpace(name))

throw new ArgumentException("Имя автора не может быть пустым");

if (birthYear <= 0 || birthYear > DateTime.Now.Year)

throw new ArgumentException("Некорректный год рождения");

Name = name;

BirthYear = birthYear;

}

public virtual void DisplayInfo()

{

Console.WriteLine($"Автор: {Name}");

Console.WriteLine($"Год рождения: {BirthYear}");

}

public int CalculateAge(int currentYear)

{

return currentYear - BirthYear;

}

}

public class Book

{

public string Title { get; set; }

public int PublicationYear { get; set; }

public Author Author { get; set; } // Композиция: книга содержит автора

// Конструктор по умолчанию

public Book()

{

Title = "Без названия";

PublicationYear = DateTime.Now.Year;

Author = new Author(); // Создание автора по умолчанию

}

// Конструктор с параметрами

public Book(string title, int publicationYear, Author author)

{

if (string.IsNullOrWhiteSpace(title))

throw new ArgumentException("Название книги не может быть пустым");

if (publicationYear <= 0 || publicationYear > DateTime.Now.Year + 1)

throw new ArgumentException("Некорректный год публикации");

if (author == null)

throw new ArgumentNullException(nameof(author), "Автор не может быть null");

Title = title;

PublicationYear = publicationYear;

Author = author; // Композиция: книга использует существующий объект Author

}

// Метод для вывода информации о книге

public virtual void DisplayInfo()

{

Console.WriteLine($"Книга: \"{Title}\"");

Console.WriteLine($"Год публикации: {PublicationYear}");

Console.WriteLine("--- Информация об авторе ---");

Author.DisplayInfo();

Console.WriteLine($"Возраст автора на момент публикации: {Author.CalculateAge(PublicationYear)} лет");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

try

{

// Создание объектов Author

Author author1 = new Author("Лев Толстой", 1828);

Author author2 = new Author("Фёдор Достоевский", 1821);

Author author3 = new Author("Антон Чехов", 1860);

Author author4 = new Author("Александр Пушкин", 1799);

// Создание объектов Book с использованием композиции

Book book1 = new Book("Война и мир", 1869, author1);

Book book2 = new Book("Анна Каренина", 1877, author1);

Book book3 = new Book("Преступление и наказание", 1866, author2);

Book book4 = new Book("Земля", 1869, author2);

Book book5 = new Book("Вишнёвый сад", 1904, author3);

Book book6 = new Book("Евгений Онегин", 1833, author4);

// Вывод информации о книгах

Console.WriteLine("Информация о книгах:");

Console.WriteLine("====================\n");

book1.DisplayInfo();

book2.DisplayInfo();

book3.DisplayInfo();

book4.DisplayInfo();

book5.DisplayInfo();

book6.DisplayInfo();

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Console.ReadLine();

}

}

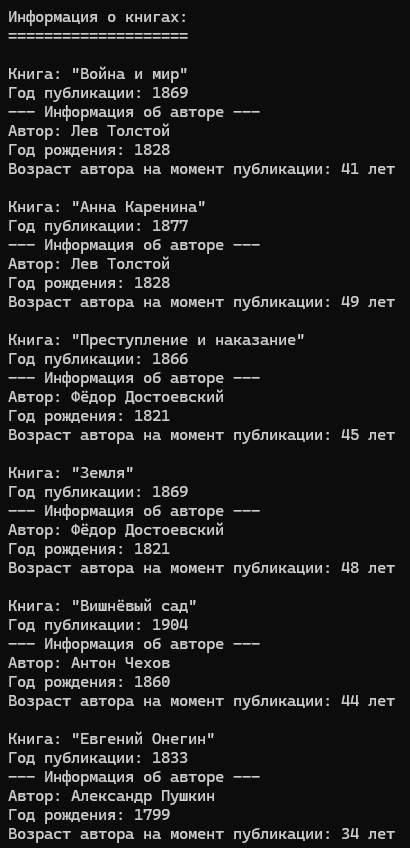


Рисунок 2.3.1 – Результат выполнения программы листинга 3

Задача 4: Интерфейсы и абстрактные классы

Создайте интерфейс IDrawable с методом Draw(), который выводит информацию о рисуемом объекте. Создайте классы Circle, Rectangle и Triangle, реализующие этот интерфейс. Создайте массив объектов, реализующих интерфейс IDrawable, и вызовите метод Draw() для каждого из них.

Листинг 4. Код программы на C#

using System;

using System.Collections.Generic;

public interface IDrawable

{

void Draw();

}

public abstract class Shape

{

public string Name { get; protected set; }

public string Color { get; set; }

protected Shape(string name, string color = "Black")

{

Name = name;

Color = color;

}

// Абстрактный метод для вычисления площади

public abstract double CalculateArea();

// Виртуальный метод для вывода общей информации

public virtual void DisplayInfo()

{

Console.WriteLine($"Фигура: {Name}");

Console.WriteLine($"Цвет: {Color}");

Console.WriteLine($"Площадь: {CalculateArea():F2}");

}

}

// Класс Circle

public class Circle : Shape, IDrawable

{

public double Radius { get; private set; }

public Circle(double radius, string color = "Red") : base("Круг", color)

{

if (radius <= 0)

throw new ArgumentException("Радиус должен быть положительным числом");

Radius = radius;

}

// Реализация метода из интерфейса IDrawable

public void Draw()

{

Console.WriteLine("Отрисовка круга:");

DisplayInfo();

Console.WriteLine($"Радиус: {Radius:F2}");

Console.WriteLine(" \*\*\* ");

Console.WriteLine(" \* \* ");

Console.WriteLine("\* \*");

Console.WriteLine(" \* \* ");

Console.WriteLine(" \*\*\* ");

Console.WriteLine();

}

// Реализация абстрактного метода из Shape

public override double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

}

// Класс Rectangle (Прямоугольник)

public class Rectangle : Shape, IDrawable

{

public double Width { get; private set; }

public double Height { get; private set; }

public Rectangle(double width, double height, string color = "Blue")

: base("Прямоугольник", color)

{

if (width <= 0 || height <= 0)

throw new ArgumentException("Ширина и высота должны быть положительными числами");

Width = width;

Height = height;

}

// Реализация метода из интерфейса IDrawable

public void Draw()

{

Console.WriteLine("Отрисовка прямоугольника:");

DisplayInfo();

Console.WriteLine($"Ширина: {Width:F2}, Высота: {Height:F2}");

Console.WriteLine("+-------+");

for (int i = 0; i < 3; i++)

{

Console.WriteLine("| |");

}

Console.WriteLine("+-------+");

Console.WriteLine();

}

// Реализация абстрактного метода из Shape

public override double CalculateArea()

{

return Width \* Height;

}

// Переопределение метода DisplayInfo

public override void DisplayInfo()

{

base.DisplayInfo();

if (Width == Height)

Console.WriteLine("Это квадрат!");

}

}

// Класс Triangle (Треугольник)

public class Triangle : Shape, IDrawable

{

public double SideA { get; private set; }

public double SideB { get; private set; }

public double SideC { get; private set; }

public Triangle(double sideA, double sideB, double sideC, string color = "Green")

: base("Треугольник", color)

{

if (sideA <= 0 || sideB <= 0 || sideC <= 0)

throw new ArgumentException("Все стороны должны быть положительными числами");

// Проверка неравенства треугольника

if (sideA + sideB <= sideC || sideA + sideC <= sideB || sideB + sideC <= sideA)

throw new ArgumentException("Некорректные стороны треугольника");

SideA = sideA;

SideB = sideB;

SideC = sideC;

}

// Реализация метода из интерфейса IDrawable

public void Draw()

{

Console.WriteLine("Отрисовка треугольника:");

DisplayInfo();

Console.WriteLine($"Стороны: {SideA:F2}, {SideB:F2}, {SideC:F2}");

Console.WriteLine($"Тип: {GetTriangleType()}");

Console.WriteLine(" /\\ ");

Console.WriteLine(" / \\ ");

Console.WriteLine(" / \\ ");

Console.WriteLine(" /\_\_\_\_\_\_\\ ");

Console.WriteLine();

}

// Реализация абстрактного метода из Shape

public override double CalculateArea()

{

// Формула Герона

double p = (SideA + SideB + SideC) / 2;

return Math.Sqrt(p \* (p - SideA) \* (p - SideB) \* (p - SideC));

}

// Метод для определения типа треугольника

private string GetTriangleType()

{

if (SideA == SideB && SideB == SideC)

return "Равносторонний";

else if (SideA == SideB || SideA == SideC || SideB == SideC)

return "Равнобедренный";

else

return "Разносторонний";

}

}

// Дополнительный класс Star (Звезда), реализующий только IDrawable

public class Star : IDrawable

{

public string Color { get; set; }

public int Points { get; set; }

public Star(string color = "Yellow", int points = 5)

{

Color = color;

Points = points;

}

// Реализация метода из интерфейса IDrawable

public void Draw()

{

Console.WriteLine("Отрисовка звезды:");

Console.WriteLine($"Цвет: {Color}");

Console.WriteLine($"Количество лучей: {Points}");

Console.WriteLine(" \* ");

Console.WriteLine(" \* \* ");

Console.WriteLine(" \* ");

Console.WriteLine(" \* \* ");

Console.WriteLine(" \* ");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("Демонстрация интерфейсов и абстрактных классов");

Console.WriteLine("==============================================\n");

// Создание массива объектов, реализующих интерфейс IDrawable

IDrawable[] drawableObjects = new IDrawable[]

{

new Circle(5.0, "Red"),

new Rectangle(4.0, 6.0, "Blue"),

new Triangle(3.0, 4.0, 5.0, "Green"),

new Rectangle(5.0, 5.0, "Purple"), // Квадрат

new Circle(3.5, "Orange"),

new Triangle(6.0, 6.0, 6.0, "Pink"), // Равносторонний треугольник

new Star("Gold", 5),

new Star("Silver", 6)

};

// Вызов метода Draw() для каждого объекта

Console.WriteLine("Отрисовка всех объектов:");

Console.WriteLine("=======================\n");

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

{

Console.WriteLine($"Объект #{i + 1}:");

drawableObjects[i].Draw();

Console.WriteLine(new string('-', 40));

}

// Демонстрация полиморфизма

Console.WriteLine("\nДемонстрация полиморфизма:");

Console.WriteLine("==========================\n");

// Группировка по типам

var circles = new List<IDrawable>();

var rectangles = new List<IDrawable>();

var triangles = new List<IDrawable>();

var stars = new List<IDrawable>();

foreach (var obj in drawableObjects)

{

if (obj is Circle)

circles.Add(obj);

else if (obj is Rectangle)

rectangles.Add(obj);

else if (obj is Triangle)

triangles.Add(obj);

else if (obj is Star)

stars.Add(obj);

}

Console.WriteLine($"Кругов: {circles.Count}");

Console.WriteLine($"Прямоугольников: {rectangles.Count}");

Console.WriteLine($"Треугольников: {triangles.Count}");

Console.WriteLine($"Звезд: {stars.Count}");

// Отрисовка по группам

Console.WriteLine("\nОтрисовка кругов:");

foreach (var circle in circles)

{

circle.Draw();

}

Console.WriteLine("Отрисовка прямоугольников:");

foreach (var rectangle in rectangles)

{

rectangle.Draw();

}

// Создание списка только фигур (наследников Shape)

Console.WriteLine("\nТолько геометрические фигуры:");

Console.WriteLine("============================\n");

List<Shape> shapes = new List<Shape>();

foreach (var obj in drawableObjects)

{

Shape shape = obj as Shape;

if (shape != null)

{

shapes.Add(shape);

}

}

double totalArea = 0;

foreach (var shape in shapes)

{

shape.DisplayInfo();

totalArea += shape.CalculateArea();

Console.WriteLine();

}

Console.WriteLine($"Общая площадь всех фигур: {totalArea:F2}");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Console.ReadLine();

}

}

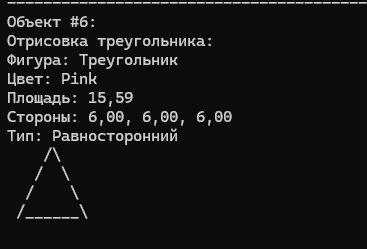


Рисунок 2.4.1 – Результат выполнения программы листинга 4

Задача 5: События

Создайте класс TemperatureSensor, который генерирует событие TemperatureChanged, когда измеренная температура меняется. Создайте класс Thermostat, который подписывается на событие TemperatureChanged и реагирует на изменение температуры, включая или выключая отопление.

Листинг 5. Код программы на C#

using System;

// Класс TemperatureSensor

public class TemperatureSensor

{

private double currentTemperature;

// Событие TemperatureChanged

public event Action<double> TemperatureChanged;

public double CurrentTemperature

{

get => currentTemperature;

set

{

if (currentTemperature != value)

{

currentTemperature = value;

TemperatureChanged?.Invoke(currentTemperature);

}

}

}

}

// Класс Thermostat

public class Thermostat

{

public Thermostat(TemperatureSensor sensor)

{

// Подписка на событие TemperatureChanged

sensor.TemperatureChanged += OnTemperatureChanged;

}

private void OnTemperatureChanged(double temperature)

{

if (temperature < 20)

{

Console.WriteLine($"Температура {temperature}°C - Включить отопление");

}

else if (temperature > 25)

{

Console.WriteLine($"Температура {temperature}°C - Выключить отопление");

}

else

{

Console.WriteLine($"Температура {temperature}°C - Поддерживать текущий режим");

}

}

}

class Program

{

static void Main(string[] args)

{

TemperatureSensor sensor = new TemperatureSensor();

Thermostat thermostat = new Thermostat(sensor);

sensor.CurrentTemperature = 18;

sensor.CurrentTemperature = 22;

sensor.CurrentTemperature = 28;

sensor.CurrentTemperature = 19;

Console.ReadLine();

}

}

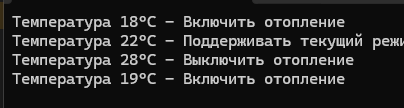


Рисунок 2.5.1 – Результат выполнения программы листинга 5

Вариант 1

Задача 1. Создайте класс "Студент" со свойствами: имя, фамилия, возраст и средний балл. Создайте несколько экземпляров этого класса и выведите информацию о них.

Листинг 6. Код программы на C#

using System;

public class Student

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

public double AverageGrade { get; set; }

// Конструктор

public Student(string firstName, string lastName, int age, double averageGrade)

{

FirstName = firstName;

LastName = lastName;

Age = age;

AverageGrade = averageGrade;

}

public void DisplayInfo()

{

Console.WriteLine($"Студент: {FirstName} {LastName}");

Console.WriteLine($"Возраст: {Age} лет");

Console.WriteLine($"Средний балл: {AverageGrade:F2}");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

Student student1 = new Student("Иван", "Петров", 20, 4.5);

Student student2 = new Student("Мария", "Сидорова", 19, 4.8);

Student student3 = new Student("Алексей", "Иванов", 21, 3.9);

student1.DisplayInfo();

student2.DisplayInfo();

student3.DisplayInfo();

}

}

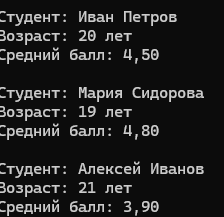


Рисунок 2.6.1 – Результат выполнения программы листинга 6

Задача 2. Создайте структуру с именем train, содержащую поля: название пункта назначения, номер поезда, время отправления. Ввести данные в массив из пяти элементов типа train, упорядочить элементы по номерам поездов. Добавить возможность вывода информации о поезде, номер которого введен пользователем. Добавить возможность сортировки массив по пункту назначения, причем поезда с одинаковыми пунктами назначения должны быть упорядочены по времени отправления.

Листинг 7. Код программы на C#

using System;

// Структура Train

public struct Train

{

public string Destination;

public int TrainNumber;

public DateTime DepartureTime;

public Train(string destination, int trainNumber, DateTime departureTime)

{

Destination = destination;

TrainNumber = trainNumber;

DepartureTime = departureTime;

}

public void DisplayInfo()

{

Console.WriteLine($"Поезд №{TrainNumber}");

Console.WriteLine($"Пункт назначения: {Destination}");

Console.WriteLine($"Время отправления: {DepartureTime:HH:mm}");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

// Создание массива из 5 поездов

Train[] trains = new Train[5]

{

new Train("Москва", 104, new DateTime(2024, 1, 1, 14, 30, 0)),

new Train("Санкт-Петербург", 78, new DateTime(2024, 1, 1, 8, 15, 0)),

new Train("Москва", 56, new DateTime(2024, 1, 1, 20, 45, 0)),

new Train("Казань", 123, new DateTime(2024, 1, 1, 16, 0, 0)),

new Train("Санкт-Петербург", 91, new DateTime(2024, 1, 1, 10, 20, 0))

};

// Упорядочивание по номерам поездов

Array.Sort(trains, (t1, t2) => t1.TrainNumber.CompareTo(t2.TrainNumber));

Console.WriteLine("Поезда, упорядоченные по номерам:");

Console.WriteLine("=================================");

foreach (var train in trains)

{

train.DisplayInfo();

}

// Поиск поезда по номеру

Console.Write("Введите номер поезда для поиска: ");

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

bool found = false;

foreach (var train in trains)

{

if (train.TrainNumber == searchNumber)

{

Console.WriteLine("Найденный поезд:");

train.DisplayInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine($"Поезд с номером {searchNumber} не найден.");

}

// Сортировка по пункту назначения и времени отправления

Array.Sort(trains, (t1, t2) =>

{

int destinationCompare = t1.Destination.CompareTo(t2.Destination);

if (destinationCompare == 0)

{

return t1.DepartureTime.CompareTo(t2.DepartureTime);

}

return destinationCompare;

});

Console.WriteLine("Поезда, упорядоченные по пункту назначения и времени отправления:");

Console.WriteLine("=================================================================");

foreach (var train in trains)

{

train.DisplayInfo();

}

}

}

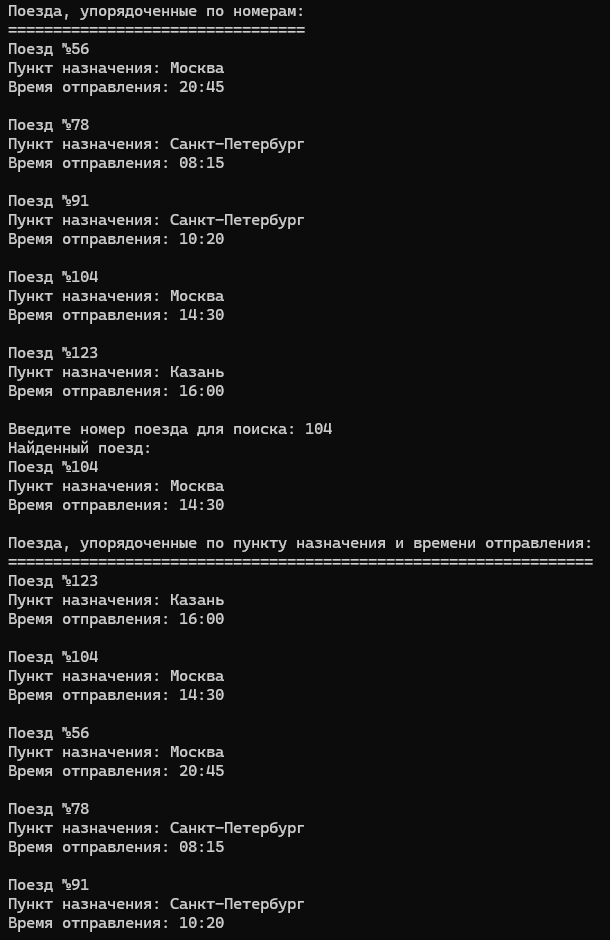


Рисунок 2.7.1 – Результат выполнения программы листинга 7

Задача 3. Разработайте иерархию классов для геометрических фигур (круг, прямоугольник, треугольник). Реализуйте методы вычисления площади и периметра для каждой фигуры.

Листинг 8. Код программы на C#

using System;

// Базовый класс для геометрических фигур

public abstract class Shape

{

public abstract double CalculateArea();

public abstract double CalculatePerimeter();

}

// Класс Circle (Круг)

public class Circle : Shape

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

public override double CalculatePerimeter()

{

return 2 \* Math.PI \* Radius;

}

}

// Класс Rectangle (Прямоугольник)

public class Rectangle : Shape

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public override double CalculateArea()

{

return Width \* Height;

}

public override double CalculatePerimeter()

{

return 2 \* (Width + Height);

}

}

// Класс Triangle (Треугольник)

public class Triangle : Shape

{

public double SideA { get; set; }

public double SideB { get; set; }

public double SideC { get; set; }

public Triangle(double sideA, double sideB, double sideC)

{

SideA = sideA;

SideB = sideB;

SideC = sideC;

}

public override double CalculateArea()

{

// Формула Герона

double p = (SideA + SideB + SideC) / 2;

return Math.Sqrt(p \* (p - SideA) \* (p - SideB) \* (p - SideC));

}

public override double CalculatePerimeter()

{

return SideA + SideB + SideC;

}

}

class Program

{

static void Main(string[] args)

{

// Создание объектов различных фигур

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

Shape triangle = new Triangle(3, 4, 5);

// Вычисление и вывод площадей и периметров

Console.WriteLine("Круг (радиус = 5):");

Console.WriteLine($"Площадь: {circle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {circle.CalculatePerimeter():F2}");

Console.WriteLine();

Console.WriteLine("Прямоугольник (4 x 6):");

Console.WriteLine($"Площадь: {rectangle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {rectangle.CalculatePerimeter():F2}");

Console.WriteLine();

Console.WriteLine("Треугольник (3, 4, 5):");

Console.WriteLine($"Площадь: {triangle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {triangle.CalculatePerimeter():F2}");

}

}

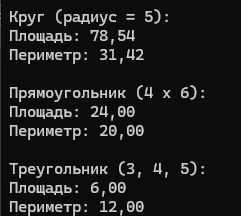


Рисунок 2.8.1 – Результат выполнения программы листинга 8

Модуль 3

Задача 1. Создайте базовый класс "Фигура" с методом для вычисления площади. Затем создайте производные классы для разных геометрических фигур (круг, прямоугольник, треугольник) и используйте делегат для динамического вызова метода вычисления площади.

Листинг 1. Код программы на C#

using System;

public abstract class Shape

{

public abstract double CalculateArea();

}

public class Circle : Shape

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public override double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

}

public class Rectangle : Shape

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public override double CalculateArea()

{

return Width \* Height;

}

}

public class Triangle : Shape

{

public double Base { get; set; }

public double Height { get; set; }

public Triangle(double @base, double height)

{

Base = @base;

Height = height;

}

public override double CalculateArea()

{

return 0.5 \* Base \* Height;

}

}

class Program

{

// Делегат для вычисления площади

public delegate double AreaCalculator();

static void Main(string[] args)

{

// Создание объектов фигур

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

Shape triangle = new Triangle(3, 4);

// Создание делегатов для каждой фигуры

AreaCalculator circleArea = circle.CalculateArea;

AreaCalculator rectangleArea = rectangle.CalculateArea;

AreaCalculator triangleArea = triangle.CalculateArea;

// Динамический вызов методов через делегаты

Console.WriteLine("Площади фигур (через делегаты):");

Console.WriteLine($"Круг: {circleArea():F2}");

Console.WriteLine($"Прямоугольник: {rectangleArea():F2}");

Console.WriteLine($"Треугольник: {triangleArea():F2}");

Console.WriteLine();

// Использование массива делегатов

AreaCalculator[] calculators = new AreaCalculator[]

{

circleArea,

rectangleArea,

triangleArea

};

Console.WriteLine("Площади фигур (через массив делегатов):");

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

{

Console.WriteLine($"Фигура {i + 1}: {calculators[i]():F2}");

}

}

}

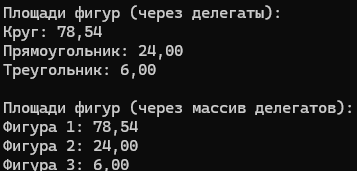


Рисунок 3.1.1 – Результат выполнения программы листинга 1

Задача 2. Реализуйте систему событий для мобильного приложения. Создайте класс "Уведомление" с событиями для отправки уведомлений (сообщения, звонки, электронные письма). Зарегистрируйте обработчики событий для разных типов уведомлений.

Листинг 2. Код программы на C#

using System;

public class Notification

{

// События для разных типов уведомлений

public event Action<string> MessageReceived;

public event Action<string> CallReceived;

public event Action<string, string> EmailReceived;

// Методы для генерации событий

public void SendMessage(string message)

{

Console.WriteLine($"Новое сообщение: {message}");

MessageReceived?.Invoke(message);

}

public void MakeCall(string caller)

{

Console.WriteLine($"Входящий звонок от: {caller}");

CallReceived?.Invoke(caller);

}

public void SendEmail(string sender, string subject)

{

Console.WriteLine($"Новое письмо от: {sender} - {subject}");

EmailReceived?.Invoke(sender, subject);

}

}

class Program

{

static void Main(string[] args)

{

Notification notification = new Notification();

// Регистрация обработчиков событий для сообщений

notification.MessageReceived += (message) =>

{

Console.WriteLine($"Обработка сообщения: показать уведомление - '{message}'");

};

notification.MessageReceived += (message) =>

{

Console.WriteLine($"Обработка сообщения: сохранить в историю - '{message}'");

};

// Регистрация обработчиков событий для звонков

notification.CallReceived += (caller) =>

{

Console.WriteLine($"Обработка звонка: показать экран вызова от '{caller}'");

};

notification.CallReceived += (caller) =>

{

Console.WriteLine($"Обработка звонка: записать в журнал вызовов '{caller}'");

};

// Регистрация обработчиков событий для email

notification.EmailReceived += (sender, subject) =>

{

Console.WriteLine($"Обработка email: уведомить пользователя о письме от '{sender}'");

};

notification.EmailReceived += (sender, subject) =>

{

Console.WriteLine($"Обработка email: добавить в папку 'Входящие' - '{subject}'");

};

// Генерация событий

Console.WriteLine("=== Отправка сообщения ===");

notification.SendMessage("Привет! Как дела?");

Console.WriteLine("\n=== Входящий звонок ===");

notification.MakeCall("Иван Петров");

Console.WriteLine("\n=== Получение email ===");

notification.SendEmail("company@example.com", "Ваш заказ готов");

Console.WriteLine("\n=== Еще одно сообщение ===");

notification.SendMessage("Напоминание о встрече");

}

}

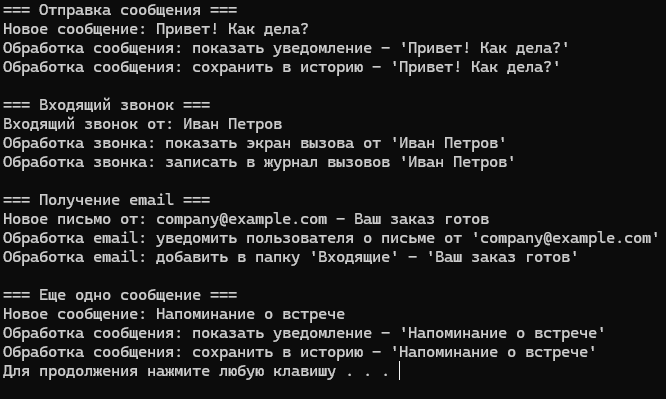


Рисунок 3.2.1 – Результат выполнения программы листинга 2

Задача 3. Создайте приложение для управления задачами с использованием делегатов. Пользователь должен иметь возможность добавлять задачи и выбирать делегата для выполнения каждой задачи (например, отправка уведомления или запись в журнал).

Листинг 3. Код программы на C#

using System;

using System.Collections.Generic;

public class Task

{

public string Description { get; set; }

public DateTime CreatedAt { get; set; }

public Task(string description)

{

Description = description;

CreatedAt = DateTime.Now;

}

}

class Program

{

public delegate void TaskExecutor(Task task);

// Методы-обработчики задач

static void SendNotification(Task task)

{

Console.WriteLine($" Уведомление: выполнена задача '{task.Description}'");

Console.WriteLine($" Время создания: {task.CreatedAt:HH:mm:ss}");

}

static void LogToJournal(Task task)

{

Console.WriteLine($"Запись в журнал: '{task.Description}'");

Console.WriteLine($" Время создания: {task.CreatedAt:HH:mm:ss}");

}

static void SendEmail(Task task)

{

Console.WriteLine($"Email отправлен: уведомление о задаче '{task.Description}'");

Console.WriteLine($" Время создания: {task.CreatedAt:HH:mm:ss}");

}

static void SaveToDatabase(Task task)

{

Console.WriteLine($"Сохранено в базу данных: '{task.Description}'");

Console.WriteLine($" Время создания: {task.CreatedAt:HH:mm:ss}");

}

static void Main(string[] args)

{

List<Task> tasks = new List<Task>();

Dictionary<string, TaskExecutor> executors = new Dictionary<string, TaskExecutor>

{

{ "1", SendNotification },

{ "2", LogToJournal },

{ "3", SendEmail },

{ "4", SaveToDatabase }

};

while (true)

{

Console.WriteLine("\n=== Управление задачами ===");

Console.WriteLine("1. Добавить задачу");

Console.WriteLine("2. Выполнить задачи");

Console.WriteLine("3. Выйти");

Console.Write("Выберите действие: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Введите описание задачи: ");

string description = Console.ReadLine();

tasks.Add(new Task(description));

Console.WriteLine("Задача добавлена!");

break;

case "2":

if (tasks.Count == 0)

{

Console.WriteLine("Список задач пуст!");

break;

}

Console.WriteLine("\nДоступные исполнители:");

Console.WriteLine("1. Отправить уведомление");

Console.WriteLine("2. Записать в журнал");

Console.WriteLine("3. Отправить email");

Console.WriteLine("4. Сохранить в базу данных");

Console.Write("Выберите исполнителя: ");

string executorChoice = Console.ReadLine();

if (executors.ContainsKey(executorChoice))

{

TaskExecutor executor = executors[executorChoice];

Console.WriteLine($"\nВыполнение задач:");

foreach (var task in tasks)

{

executor(task);

}

tasks.Clear();

Console.WriteLine("Все задачи выполнены и удалены из списка!");

}

else

{

Console.WriteLine("Неверный выбор исполнителя!");

}

break;

case "3":

Console.WriteLine("Выход из приложения...");

return;

default:

Console.WriteLine("Неверный выбор!");

break;

}

}

}

}

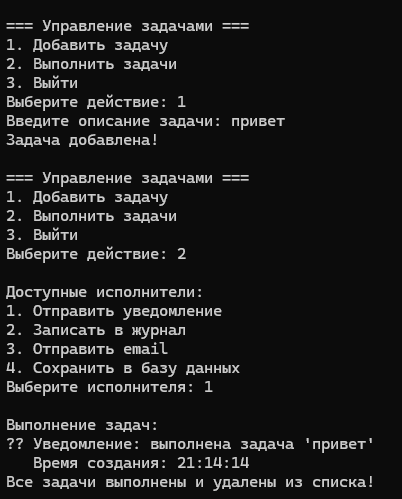


Рисунок 3.3.1 – Результат выполнения программы листинга 3

Задача 4. Разработайте систему фильтрации данных с использованием делегатов. Пользователь должен иметь возможность выбрать фильтр для списка данных (например, фильтр по дате или по ключевым словам).

Листинг 4. Код программы на C#

using System;

using System.Collections.Generic;

public class DataItem

{

public string Title { get; set; }

public string Content { get; set; }

public DateTime Date { get; set; }

public string Category { get; set; }

public DataItem(string title, string content, DateTime date, string category)

{

Title = title;

Content = content;

Date = date;

Category = category;

}

public override string ToString()

{

return $"{Title} ({Date:dd.MM.yyyy}) - {Category}";

}

}

class Program

{

// Делегат для фильтрации данных

public delegate bool FilterDelegate(DataItem item);

// Методы-фильтры

static bool FilterByDate(DataItem item)

{

Console.Write("Введите дату для фильтра (dd.MM.yyyy): ");

if (DateTime.TryParse(Console.ReadLine(), out DateTime filterDate))

{

return item.Date.Date == filterDate.Date;

}

return false;

}

static bool FilterByKeyword(DataItem item)

{

Console.Write("Введите ключевое слово: ");

string keyword = Console.ReadLine().ToLower();

return item.Title.ToLower().Contains(keyword) ||

item.Content.ToLower().Contains(keyword);

}

static bool FilterByCategory(DataItem item)

{

Console.Write("Введите категорию: ");

string category = Console.ReadLine().ToLower();

return item.Category.ToLower().Contains(category);

}

static bool FilterByDateRange(DataItem item)

{

Console.Write("Введите начальную дату (dd.MM.yyyy): ");

DateTime startDate = DateTime.Parse(Console.ReadLine());

Console.Write("Введите конечную дату (dd.MM.yyyy): ");

DateTime endDate = DateTime.Parse(Console.ReadLine());

return item.Date >= startDate && item.Date <= endDate;

}

// Метод для применения фильтра

static List<DataItem> ApplyFilter(List<DataItem> data, FilterDelegate filter)

{

List<DataItem> result = new List<DataItem>();

foreach (var item in data)

{

if (filter(item))

{

result.Add(item);

}

}

return result;

}

static void Main(string[] args)

{

// Создание тестовых данных

List<DataItem> data = new List<DataItem>

{

new DataItem("Отчет за январь", "Ежемесячный финансовый отчет", new DateTime(2024, 1, 31), "Финансы"),

new DataItem("Встреча с клиентом", "Обсуждение нового проекта", new DateTime(2024, 2, 15), "Встречи"),

new DataItem("Техническое задание", "Разработка нового функционала", new DateTime(2024, 2, 20), "Разработка"),

new DataItem("Финансовый план", "Планирование бюджета на квартал", new DateTime(2024, 3, 1), "Финансы"),

new DataItem("Обучение сотрудников", "Проведение тренинга по новым технологиям", new DateTime(2024, 3, 10), "Обучение")

};

// Словарь доступных фильтров

Dictionary<string, FilterDelegate> filters = new Dictionary<string, FilterDelegate>

{

{ "1", FilterByDate },

{ "2", FilterByKeyword },

{ "3", FilterByCategory },

{ "4", FilterByDateRange }

};

while (true)

{

Console.WriteLine("\n=== Система фильтрации данных ===");

Console.WriteLine("Исходные данные:");

foreach (var item in data)

{

Console.WriteLine($" {item}");

}

Console.WriteLine("\nДоступные фильтры:");

Console.WriteLine("1. Фильтр по дате");

Console.WriteLine("2. Фильтр по ключевому слову");

Console.WriteLine("3. Фильтр по категории");

Console.WriteLine("4. Фильтр по диапазону дат");

Console.WriteLine("5. Выйти");

Console.Write("Выберите фильтр: ");

string choice = Console.ReadLine();

if (choice == "5")

{

Console.WriteLine("Выход из системы...");

break;

}

if (filters.ContainsKey(choice))

{

FilterDelegate selectedFilter = filters[choice];

List<DataItem> filteredData = ApplyFilter(data, selectedFilter);

Console.WriteLine("\nРезультаты фильтрации:");

if (filteredData.Count > 0)

{

foreach (var item in filteredData)

{

Console.WriteLine($" {item}");

Console.WriteLine($" Содержание: {item.Content}");

}

}

else

{

Console.WriteLine(" Данные не найдены");

}

}

else

{

Console.WriteLine("Неверный выбор фильтра!");

}

}

}

}

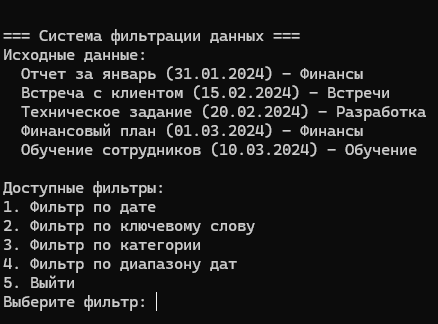


Рисунок 3.4.1 – Результат выполнения программы листинга 4

Задача 5. Создайте приложение для сортировки числовых данных. Пользователь должен иметь возможность выбрать метод сортировки (например, сортировка пузырьком или быстрая сортировка) с помощью делегатов.

Листинг 5. Код программы на C#

using System;

using System.Collections.Generic;

class Program

{

// Делегат для методов сортировки

public delegate void SortDelegate(List<int> data);

// Метод сортировки пузырьком

static void BubbleSort(List<int> data)

{

for (int i = 0; i < data.Count - 1; i++)

{

for (int j = 0; j < data.Count - i - 1; j++)

{

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

{

int temp = data[j];

data[j] = data[j + 1];

data[j + 1] = temp;

}

}

}

}

// Метод быстрой сортировки

static void QuickSort(List<int> data)

{

QuickSortRecursive(data, 0, data.Count - 1);

}

static void QuickSortRecursive(List<int> data, int left, int right)

{

if (left < right)

{

int pivotIndex = Partition(data, left, right);

QuickSortRecursive(data, left, pivotIndex - 1);

QuickSortRecursive(data, pivotIndex + 1, right);

}

}

static int Partition(List<int> data, int left, int right)

{

int pivot = data[right];

int i = left - 1;

for (int j = left; j < right; j++)

{

if (data[j] <= pivot)

{

i++;

int temp = data[i];

data[i] = data[j];

data[j] = temp;

}

}

int temp2 = data[i + 1];

data[i + 1] = data[right];

data[right] = temp2;

return i + 1;

}

// Метод сортировки выбором

static void SelectionSort(List<int> data)

{

for (int i = 0; i < data.Count - 1; i++)

{

int minIndex = i;

for (int j = i + 1; j < data.Count; j++)

{

if (data[j] < data[minIndex])

{

minIndex = j;

}

}

if (minIndex != i)

{

int temp = data[i];

data[i] = data[minIndex];

data[minIndex] = temp;

}

}

}

// Метод для вывода массива

static void PrintArray(List<int> data)

{

Console.WriteLine(string.Join(", ", data));

}

// Метод для генерации случайных данных

static List<int> GenerateRandomData(int count)

{

Random random = new Random();

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

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

{

data.Add(random.Next(1, 100));

}

return data;

}

static void Main(string[] args)

{

// Словарь доступных методов сортировки

Dictionary<string, SortDelegate> sortMethods = new Dictionary<string, SortDelegate>

{

{ "1", BubbleSort },

{ "2", QuickSort },

{ "3", SelectionSort }

};

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

while (true)

{

Console.WriteLine("\n=== Приложение для сортировки ===");

Console.WriteLine("1. Ввести данные вручную");

Console.WriteLine("2. Сгенерировать случайные данные");

Console.WriteLine("3. Выбрать метод сортировки");

Console.WriteLine("4. Выйти");

Console.Write("Выберите действие: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Введите числа через пробел: ");

string input = Console.ReadLine();

string[] numbers = input.Split(' ');

data.Clear();

foreach (string number in numbers)

{

if (int.TryParse(number, out int num))

{

data.Add(num);

}

}

Console.WriteLine("Данные сохранены: " + string.Join(", ", data));

break;

case "2":

Console.Write("Введите количество элементов: ");

if (int.TryParse(Console.ReadLine(), out int count) && count > 0)

{

data = GenerateRandomData(count);

Console.WriteLine("Сгенерированные данные: " + string.Join(", ", data));

}

else

{

Console.WriteLine("Неверное количество!");

}

break;

case "3":

if (data.Count == 0)

{

Console.WriteLine("Сначала введите данные!");

break;

}

Console.WriteLine("\nДоступные методы сортировки:");

Console.WriteLine("1. Сортировка пузырьком");

Console.WriteLine("2. Быстрая сортировка");

Console.WriteLine("3. Сортировка выбором");

Console.Write("Выберите метод: ");

string methodChoice = Console.ReadLine();

if (sortMethods.ContainsKey(methodChoice))

{

List<int> dataToSort = new List<int>(data); // Копия для сортировки

Console.WriteLine("\nИсходные данные:");

PrintArray(dataToSort);

SortDelegate selectedSort = sortMethods[methodChoice];

selectedSort(dataToSort);

Console.WriteLine("Отсортированные данные:");

PrintArray(dataToSort);

}

else

{

Console.WriteLine("Неверный выбор метода!");

}

break;

case "4":

Console.WriteLine("Выход из приложения...");

return;

default:

Console.WriteLine("Неверный выбор!");

break;

}

}

}

}

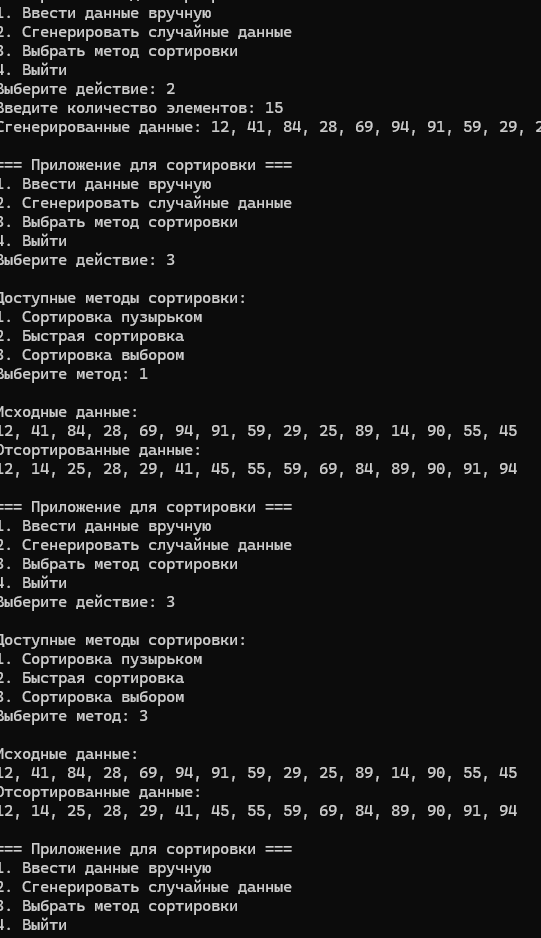


Рисунок 3.5.1 – Результат выполнения программы листинга 5

Модуль 4

Задача 1. Создайте интерфейс "Фигура" с методами для вычисления площади и периметра. Затем реализуйте этот интерфейс в классах геометрических фигур (круг, прямоугольник, треугольник).

Листинг 1. Код программы на C#

using System;

// Интерфейс Фигура

public interface IShape

{

double CalculateArea();

double CalculatePerimeter();

}

// Класс Circle (Круг)

public class Circle : IShape

{

public double Radius { get; set; }

public Circle(double radius)

{

Radius = radius;

}

public double CalculateArea()

{

return Math.PI \* Radius \* Radius;

}

public double CalculatePerimeter()

{

return 2 \* Math.PI \* Radius;

}

}

// Класс Rectangle (Прямоугольник)

public class Rectangle : IShape

{

public double Width { get; set; }

public double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = height;

}

public double CalculateArea()

{

return Width \* Height;

}

public double CalculatePerimeter()

{

return 2 \* (Width + Height);

}

}

// Класс Triangle (Треугольник)

public class Triangle : IShape

{

public double SideA { get; set; }

public double SideB { get; set; }

public double SideC { get; set; }

public Triangle(double sideA, double sideB, double sideC)

{

SideA = sideA;

SideB = sideB;

SideC = sideC;

}

public double CalculateArea()

{

// Формула Герона

double p = (SideA + SideB + SideC) / 2;

return Math.Sqrt(p \* (p - SideA) \* (p - SideB) \* (p - SideC));

}

public double CalculatePerimeter()

{

return SideA + SideB + SideC;

}

}

class Program

{

static void Main(string[] args)

{

// Создание объектов фигур

IShape circle = new Circle(5);

IShape rectangle = new Rectangle(4, 6);

IShape triangle = new Triangle(3, 4, 5);

// Вычисление и вывод площадей и периметров

Console.WriteLine("Круг (радиус = 5):");

Console.WriteLine($"Площадь: {circle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {circle.CalculatePerimeter():F2}");

Console.WriteLine();

Console.WriteLine("Прямоугольник (4 x 6):");

Console.WriteLine($"Площадь: {rectangle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {rectangle.CalculatePerimeter():F2}");

Console.WriteLine();

Console.WriteLine("Треугольник (3, 4, 5):");

Console.WriteLine($"Площадь: {triangle.CalculateArea():F2}");

Console.WriteLine($"Периметр: {triangle.CalculatePerimeter():F2}");

}

}

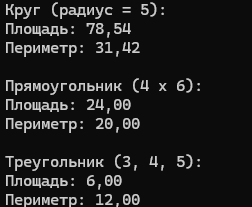


Рисунок 4.1.1 – Результат выполнения программы листинга 1

Задача 2. Разработайте приложение для учета продуктов в магазине с использованием интерфейсов. Создайте интерфейс "Товар" с методами для определения стоимости и остатка товара на складе. Реализуйте этот интерфейс в классах различных товаров.

Листинг 2. Код программы на C#

using System;

using System.Collections.Generic;

public interface IProduct

{

string Name { get; set; }

double GetPrice();

int GetStockQuantity();

void DisplayInfo();

}

public class FoodProduct : IProduct

{

public string Name { get; set; }

public double PricePerKg { get; set; }

public double Weight { get; set; }

public int Quantity { get; set; }

public FoodProduct(string name, double pricePerKg, double weight, int quantity)

{

Name = name;

PricePerKg = pricePerKg;

Weight = weight;

Quantity = quantity;

}

public double GetPrice()

{

return PricePerKg \* Weight;

}

public int GetStockQuantity()

{

return Quantity;

}

public void DisplayInfo()

{

Console.WriteLine($"Продукт: {Name}");

Console.WriteLine($"Цена за кг: {PricePerKg:F2} руб.");

Console.WriteLine($"Вес: {Weight:F2} кг");

Console.WriteLine($"Стоимость: {GetPrice():F2} руб.");

Console.WriteLine($"Остаток на складе: {Quantity} шт.");

Console.WriteLine();

}

}

public class ElectronicsProduct : IProduct

{

public string Name { get; set; }

public double Price { get; set; }

public int Quantity { get; set; }

public int WarrantyMonths { get; set; }

public ElectronicsProduct(string name, double price, int quantity, int warrantyMonths)

{

Name = name;

Price = price;

Quantity = quantity;

WarrantyMonths = warrantyMonths;

}

public double GetPrice()

{

return Price;

}

public int GetStockQuantity()

{

return Quantity;

}

public void DisplayInfo()

{

Console.WriteLine($"Электроника: {Name}");

Console.WriteLine($"Цена: {Price:F2} руб.");

Console.WriteLine($"Гарантия: {WarrantyMonths} мес.");

Console.WriteLine($"Остаток на складе: {Quantity} шт.");

Console.WriteLine();

}

}

public class ClothingProduct : IProduct

{

public string Name { get; set; }

public double Price { get; set; }

public int Quantity { get; set; }

public string Size { get; set; }

public ClothingProduct(string name, double price, int quantity, string size)

{

Name = name;

Price = price;

Quantity = quantity;

Size = size;

}

public double GetPrice()

{

return Price;

}

public int GetStockQuantity()

{

return Quantity;

}

public void DisplayInfo()

{

Console.WriteLine($"Одежда: {Name}");

Console.WriteLine($"Цена: {Price:F2} руб.");

Console.WriteLine($"Размер: {Size}");

Console.WriteLine($"Остаток на складе: {Quantity} шт.");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

List<IProduct> products = new List<IProduct>

{

new FoodProduct("Яблоки", 120.50, 2.5, 50),

new FoodProduct("Молоко", 85.00, 1.0, 30),

new ElectronicsProduct("Смартфон", 25000.00, 15, 24),

new ElectronicsProduct("Ноутбук", 65000.00, 8, 36),

new ClothingProduct("Футболка", 1500.00, 25, "L"),

new ClothingProduct("Джинсы", 3500.00, 18, "M")

};

Console.WriteLine("=== Учет продуктов в магазине ===");

Console.WriteLine("Все товары:");

Console.WriteLine("============");

foreach (var product in products)

{

product.DisplayInfo();

}

double totalValue = 0;

foreach (var product in products)

{

totalValue += product.GetPrice() \* product.GetStockQuantity();

}

Console.WriteLine($"Общая стоимость товаров на складе: {totalValue:F2} руб.");

Console.WriteLine("\nТовары с низким остатком (меньше 10 шт.):");

Console.WriteLine("========================================");

foreach (var product in products)

{

if (product.GetStockQuantity() < 10)

{

product.DisplayInfo();

}

}

Console.Write("\nВведите название товара для поиска: ");

string searchName = Console.ReadLine();

bool found = false;

foreach (var product in products)

{

if (product.Name.ToLower().Contains(searchName.ToLower()))

{

Console.WriteLine("Найденный товар:");

product.DisplayInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("Товар не найден!");

}

}

}

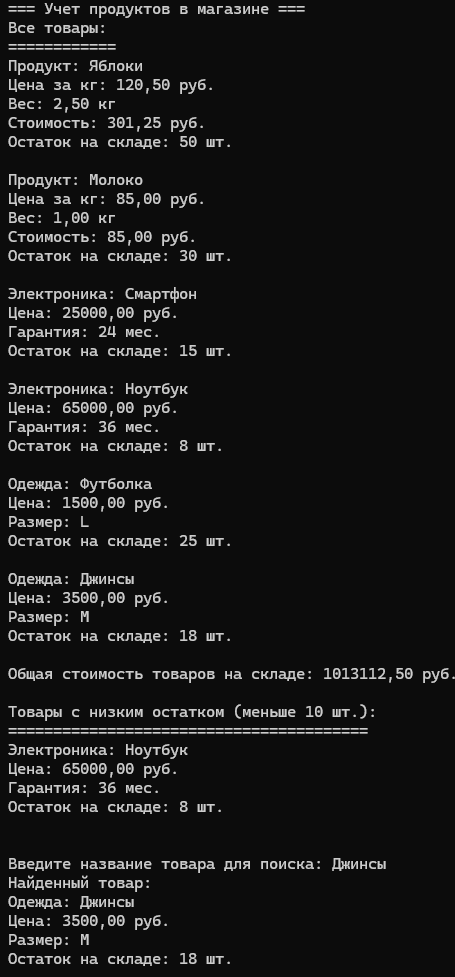


Рисунок 4.2.1 – Результат выполнения программы листинга 2

Задача 3. Создайте систему учета студентов в университете с помощью интерфейсов. Создайте интерфейс "Студент" с методами для определения среднего балла и получения информации о курсе. Реализуйте этот интерфейс в классах студентов разных курсов.

Листинг 3. Код программы на C#

using System;

using System.Collections.Generic;

// Интерфейс Студент

public interface IStudent

{

string Name { get; set; }

string Specialty { get; set; }

double CalculateAverageGrade();

string GetCourseInfo();

void DisplayInfo();

}

// Класс FirstYearStudent (Студент 1 курса)

public class FirstYearStudent : IStudent

{

public string Name { get; set; }

public string Specialty { get; set; }

public List<double> Grades { get; set; }

public FirstYearStudent(string name, string specialty)

{

Name = name;

Specialty = specialty;

Grades = new List<double>();

}

public void AddGrade(double grade)

{

Grades.Add(grade);

}

public double CalculateAverageGrade()

{

if (Grades.Count == 0) return 0;

double sum = 0;

foreach (var grade in Grades)

{

sum += grade;

}

return sum / Grades.Count;

}

public string GetCourseInfo()

{

return "1 курс - базовые дисциплины";

}

public void DisplayInfo()

{

Console.WriteLine($"Студент 1 курса: {Name}");

Console.WriteLine($"Специальность: {Specialty}");

Console.WriteLine($"Средний балл: {CalculateAverageGrade():F2}");

Console.WriteLine($"Курс: {GetCourseInfo()}");

Console.WriteLine($"Оценки: {string.Join(", ", Grades)}");

Console.WriteLine();

}

}

// Класс SecondYearStudent (Студент 2 курса)

public class SecondYearStudent : IStudent

{

public string Name { get; set; }

public string Specialty { get; set; }

public List<double> Grades { get; set; }

public string PracticeCompany { get; set; }

public SecondYearStudent(string name, string specialty, string practiceCompany = "")

{

Name = name;

Specialty = specialty;

PracticeCompany = practiceCompany;

Grades = new List<double>();

}

public void AddGrade(double grade)

{

Grades.Add(grade);

}

public double CalculateAverageGrade()

{

if (Grades.Count == 0) return 0;

double sum = 0;

foreach (var grade in Grades)

{

sum += grade;

}

return sum / Grades.Count;

}

public string GetCourseInfo()

{

string practiceInfo = string.IsNullOrEmpty(PracticeCompany) ?

"практика не назначена" : $"практика в {PracticeCompany}";

return $"2 курс - профильные дисциплины, {practiceInfo}";

}

public void DisplayInfo()

{

Console.WriteLine($"Студент 2 курса: {Name}");

Console.WriteLine($"Специальность: {Specialty}");

Console.WriteLine($"Средний балл: {CalculateAverageGrade():F2}");

Console.WriteLine($"Курс: {GetCourseInfo()}");

Console.WriteLine($"Оценки: {string.Join(", ", Grades)}");

Console.WriteLine();

}

}

// Класс ThirdYearStudent (Студент 3 курса)

public class ThirdYearStudent : IStudent

{

public string Name { get; set; }

public string Specialty { get; set; }

public List<double> Grades { get; set; }

public string DiplomaTopic { get; set; }

public ThirdYearStudent(string name, string specialty, string diplomaTopic = "")

{

Name = name;

Specialty = specialty;

DiplomaTopic = diplomaTopic;

Grades = new List<double>();

}

public void AddGrade(double grade)

{

Grades.Add(grade);

}

public double CalculateAverageGrade()

{

if (Grades.Count == 0) return 0;

double sum = 0;

foreach (var grade in Grades)

{

sum += grade;

}

return sum / Grades.Count;

}

public string GetCourseInfo()

{

string diplomaInfo = string.IsNullOrEmpty(DiplomaTopic) ?

"тема диплома не выбрана" : $"тема диплома: {DiplomaTopic}";

return $"3 курс - углубленная специализация, {diplomaInfo}";

}

public void DisplayInfo()

{

Console.WriteLine($"Студент 3 курса: {Name}");

Console.WriteLine($"Специальность: {Specialty}");

Console.WriteLine($"Средний балл: {CalculateAverageGrade():F2}");

Console.WriteLine($"Курс: {GetCourseInfo()}");

Console.WriteLine($"Оценки: {string.Join(", ", Grades)}");

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

// Создание списка студентов

List<IStudent> students = new List<IStudent>

{

new FirstYearStudent("Иван Петров", "Информатика"),

new FirstYearStudent("Мария Сидорова", "Математика"),

new SecondYearStudent("Алексей Иванов", "Физика", "НИИ Физики"),

new SecondYearStudent("Елена Козлова", "Химия"),

new ThirdYearStudent("Дмитрий Смирнов", "Программирование", "Разработка AI-систем"),

new ThirdYearStudent("Ольга Новикова", "Биология", "Исследование ДНК")

};

// Добавление оценок студентам

((FirstYearStudent)students[0]).AddGrade(4.5);

((FirstYearStudent)students[0]).AddGrade(5.0);

((FirstYearStudent)students[0]).AddGrade(4.0);

((FirstYearStudent)students[1]).AddGrade(3.5);

((FirstYearStudent)students[1]).AddGrade(4.0);

((SecondYearStudent)students[2]).AddGrade(5.0);

((SecondYearStudent)students[2]).AddGrade(4.5);

((SecondYearStudent)students[2]).AddGrade(5.0);

((SecondYearStudent)students[3]).AddGrade(4.0);

((SecondYearStudent)students[3]).AddGrade(3.0);

((ThirdYearStudent)students[4]).AddGrade(5.0);

((ThirdYearStudent)students[4]).AddGrade(5.0);

((ThirdYearStudent)students[4]).AddGrade(4.5);

((ThirdYearStudent)students[5]).AddGrade(4.0);

((ThirdYearStudent)students[5]).AddGrade(4.5);

// Вывод информации о всех студентах

Console.WriteLine("=== Система учета студентов университета ===");

Console.WriteLine("Все студенты:");

Console.WriteLine("==============");

foreach (var student in students)

{

student.DisplayInfo();

}

// Средний балл по всем студентам

double totalAverage = 0;

foreach (var student in students)

{

totalAverage += student.CalculateAverageGrade();

}

totalAverage /= students.Count;

Console.WriteLine($"Средний балл по всем студентам: {totalAverage:F2}");

// Студенты с высоким средним баллом

Console.WriteLine("\nСтуденты с высоким средним баллом (>= 4.5):");

Console.WriteLine("===========================================");

foreach (var student in students)

{

if (student.CalculateAverageGrade() >= 4.5)

{

student.DisplayInfo();

}

}

// Поиск студента по имени

Console.Write("\nВведите имя студента для поиска: ");

string searchName = Console.ReadLine();

bool found = false;

foreach (var student in students)

{

if (student.Name.ToLower().Contains(searchName.ToLower()))

{

Console.WriteLine("Найденный студент:");

student.DisplayInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("Студент не найден!");

}

}

}

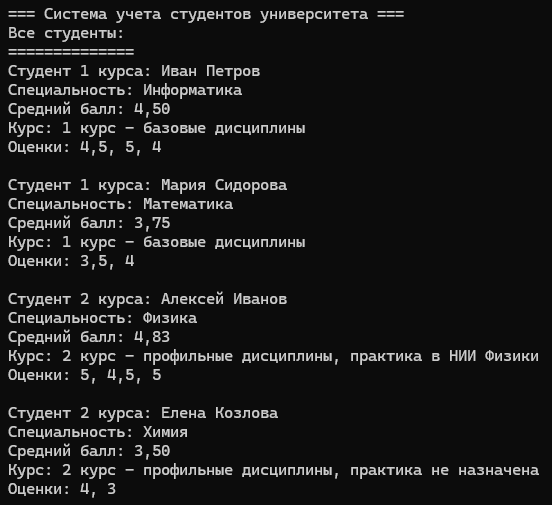


Рисунок 4.3.1 – Результат выполнения программы листинга 3

Задача 4. Реализуйте приложение для работы с библиотекой книг с использованием интерфейсов. Создайте интерфейс "Книга" с методами для проверки доступности и выдачи книги. Реализуйте этот интерфейс в классах различных книг.

Листинг 4. Код программы на C#

using System;

using System.Collections.Generic;

public interface IBook

{

string Title { get; set; }

string Author { get; set; }

bool IsAvailable();

void BorrowBook();

void ReturnBook();

}

public class PrintedBook : IBook

{

public string Title { get; set; }

public string Author { get; set; }

private bool isBorrowed;

public PrintedBook(string title, string author)

{

Title = title;

Author = author;

isBorrowed = false;

}

public bool IsAvailable()

{

return !isBorrowed;

}

public void BorrowBook()

{

if (!isBorrowed)

{

isBorrowed = true;

Console.WriteLine($"Книга '{Title}' выдана");

}

else

{

Console.WriteLine($"Книга '{Title}' уже выдана");

}

}

public void ReturnBook()

{

if (isBorrowed)

{

isBorrowed = false;

Console.WriteLine($"Книга '{Title}' возвращена");

}

else

{

Console.WriteLine($"Книга '{Title}' уже в библиотеке");

}

}

}

// Класс EBook (Электронная книга)

public class EBook : IBook

{

public string Title { get; set; }

public string Author { get; set; }

private int downloadCount;

private const int MaxDownloads = 3;

public EBook(string title, string author)

{

Title = title;

Author = author;

downloadCount = 0;

}

public bool IsAvailable()

{

return downloadCount < MaxDownloads;

}

public void BorrowBook()

{

if (downloadCount < MaxDownloads)

{

downloadCount++;

Console.WriteLine($"Электронная книга '{Title}' скачана. Скачиваний: {downloadCount}/{MaxDownloads}");

}

else

{

Console.WriteLine($"Достигнут лимит скачиваний для книги '{Title}'");

}

}

public void ReturnBook()

{

if (downloadCount > 0)

{

downloadCount--;

Console.WriteLine($"Скачивание электронной книги '{Title}' отменено. Скачиваний: {downloadCount}/{MaxDownloads}");

}

else

{

Console.WriteLine($"Нет активных скачиваний для книги '{Title}'");

}

}

}

class Program

{

static void Main(string[] args)

{

// Создание списка книг

List<IBook> books = new List<IBook>

{

new PrintedBook("Война и мир", "Лев Толстой"),

new PrintedBook("Преступление и наказание", "Федор Достоевский"),

new EBook("1984", "Джордж Оруэлл"),

new EBook("Мастер и Маргарита", "Михаил Булгаков")

};

// Вывод информации о книгах

Console.WriteLine("=== Библиотека книг ===");

foreach (var book in books)

{

string status = book.IsAvailable() ? "доступна" : "не доступна";

Console.WriteLine($"{book.Title} - {book.Author} ({status})");

}

// Работа с книгами

Console.WriteLine("\n=== Выдача книг ===");

books[0].BorrowBook(); // Печатная книга

books[2].BorrowBook(); // Электронная книга

books[2].BorrowBook(); // Электронная книга еще раз

Console.WriteLine("\n=== Возврат книг ===");

books[0].ReturnBook(); // Печатная книга

books[2].ReturnBook(); // Электронная книга

Console.WriteLine("\n=== Проверка доступности ===");

foreach (var book in books)

{

string status = book.IsAvailable() ? "доступна" : "не доступна";

Console.WriteLine($"{book.Title} - {status}");

}

}

}

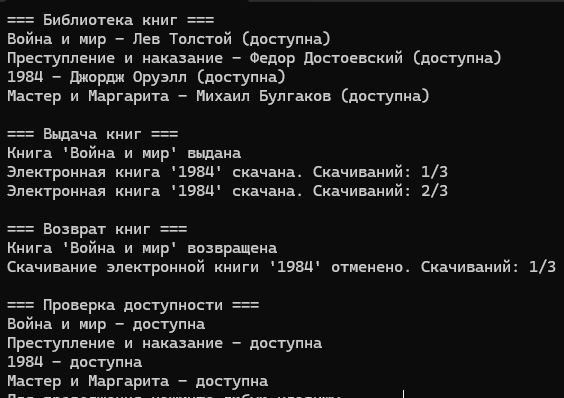


Рисунок 4.4.1 – Результат выполнения программы листинга 4

Задача 5. Создайте приложение для рисования на холсте с использованием интерфейсов. Создайте интерфейс "Рисунок" с методами для рисования линий, кругов и прямоугольников. Реализуйте этот интерфейс в классе для работы с холстом.

Листинг 5. Код программы на C#

using System;

// Интерфейс Рисунок

public interface IDrawing

{

void DrawLine(int x1, int y1, int x2, int y2);

void DrawCircle(int x, int y, int radius);

void DrawRectangle(int x, int y, int width, int height);

}

// Класс Canvas (Холст)

public class Canvas : IDrawing

{

private int width;

private int height;

private char[,] canvas;

public Canvas(int width, int height)

{

this.width = width;

this.height = height;

canvas = new char[height, width];

ClearCanvas();

}

private void ClearCanvas()

{

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

canvas[y, x] = ' ';

}

}

}

public void DrawLine(int x1, int y1, int x2, int y2)

{

Console.WriteLine($"Рисую линию от ({x1},{y1}) до ({x2},{y2})");

// Простой алгоритм Брезенхема для горизонтальных и вертикальных линий

if (x1 == x2)

{

// Вертикальная линия

int startY = Math.Min(y1, y2);

int endY = Math.Max(y1, y2);

for (int y = startY; y <= endY; y++)

{

if (IsInBounds(x1, y))

canvas[y, x1] = '|';

}

}

else if (y1 == y2)

{

// Горизонтальная линия

int startX = Math.Min(x1, x2);

int endX = Math.Max(x1, x2);

for (int x = startX; x <= endX; x++)

{

if (IsInBounds(x, y1))

canvas[y1, x] = '-';

}

}

}

public void DrawCircle(int x, int y, int radius)

{

Console.WriteLine($"Рисую круг в точке ({x},{y}) с радиусом {radius}");

// Простая реализация окружности

for (int angle = 0; angle < 360; angle++)

{

double rad = angle \* Math.PI / 180;

int px = (int)(x + radius \* Math.Cos(rad));

int py = (int)(y + radius \* Math.Sin(rad));

if (IsInBounds(px, py))

canvas[py, px] = '\*';

}

}

public void DrawRectangle(int x, int y, int width, int height)

{

Console.WriteLine($"Рисую прямоугольник в точке ({x},{y}) размером {width}x{height}");

// Верхняя и нижняя стороны

for (int i = x; i < x + width; i++)

{

if (IsInBounds(i, y))

canvas[y, i] = '-';

if (IsInBounds(i, y + height - 1))

canvas[y + height - 1, i] = '-';

}

// Левая и правая стороны

for (int i = y; i < y + height; i++)

{

if (IsInBounds(x, i))

canvas[i, x] = '|';

if (IsInBounds(x + width - 1, i))

canvas[i, x + width - 1] = '|';

}

// Углы

if (IsInBounds(x, y))

canvas[y, x] = '+';

if (IsInBounds(x + width - 1, y))

canvas[y, x + width - 1] = '+';

if (IsInBounds(x, y + height - 1))

canvas[y + height - 1, x] = '+';

if (IsInBounds(x + width - 1, y + height - 1))

canvas[y + height - 1, x + width - 1] = '+';

}

private bool IsInBounds(int x, int y)

{

return x >= 0 && x < width && y >= 0 && y < height;

}

public void Display()

{

Console.WriteLine("Холст:");

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

Console.Write(canvas[y, x]);

}

Console.WriteLine();

}

}

}

class Program

{

static void Main(string[] args)

{

// Создание холста

Canvas canvas = new Canvas(30, 15);

IDrawing drawing = canvas;

// Рисование различных фигур

drawing.DrawLine(5, 5, 25, 5);

drawing.DrawLine(5, 5, 5, 12);

drawing.DrawCircle(15, 7, 4);

drawing.DrawRectangle(20, 2, 8, 6);

drawing.DrawRectangle(2, 8, 10, 4);

// Отображение холста

canvas.Display();

}

}

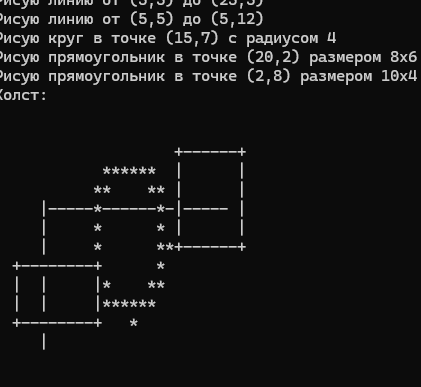


Рисунок 4.5.1 – Результат выполнения программы листинга 5

Модуль 5

Задача 1. Создайте графическое приложение для рисования на холсте с использованием Windows Forms. Реализуйте функции рисования линий, кругов и квадратов.

Листинг 1. Код программы на C#

using System;

using System.Drawing;

using System.Windows.Forms;

namespace SimpleDrawingApp

{

public partial class Form1 : Form

{

private Point startPoint;

private Point endPoint;

private bool isDrawing = false;

private ShapeType currentShape = ShapeType.Line;

private Pen drawingPen = new Pen(Color.Black, 2);

private Bitmap canvasBitmap;

private Graphics canvasGraphics;

public Form1()

{

InitializeComponent();

InitializeCanvas();

}

private void InitializeCanvas()

{

canvasBitmap = new Bitmap(pictureBox.Width, pictureBox.Height);

canvasGraphics = Graphics.FromImage(canvasBitmap);

canvasGraphics.Clear(Color.White);

pictureBox.Image = canvasBitmap;

}

private void pictureBox\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

isDrawing = true;

startPoint = e.Location;

endPoint = e.Location;

}

}

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

if (isDrawing)

{

endPoint = e.Location;

pictureBox.Invalidate(); // Перерисовываем pictureBox для отображения временной фигуры

}

}

private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

{

if (isDrawing && e.Button == MouseButtons.Left)

{

isDrawing = false;

endPoint = e.Location;

DrawFinalShape();

pictureBox.Invalidate();

}

}

private void pictureBox\_Paint(object sender, PaintEventArgs e)

{

// Отображаем основное изображение

if (canvasBitmap != null)

{

e.Graphics.DrawImage(canvasBitmap, 0, 0);

}

// Рисуем временную фигуру при перетаскивании

if (isDrawing)

{

DrawTemporaryShape(e.Graphics);

}

}

private void DrawTemporaryShape(Graphics g)

{

using (Pen tempPen = new Pen(Color.Gray, 1))

{

tempPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;

DrawShape(g, tempPen, startPoint, endPoint);

}

}

private void DrawFinalShape()

{

DrawShape(canvasGraphics, drawingPen, startPoint, endPoint);

}

private void DrawShape(Graphics g, Pen pen, Point start, Point end)

{

int x = Math.Min(start.X, end.X);

int y = Math.Min(start.Y, end.Y);

int width = Math.Abs(end.X - start.X);

int height = Math.Abs(end.Y - start.Y);

switch (currentShape)

{

case ShapeType.Line:

g.DrawLine(pen, start, end);

break;

case ShapeType.Circle:

g.DrawEllipse(pen, x, y, width, height);

break;

case ShapeType.Rectangle:

g.DrawRectangle(pen, x, y, width, height);

break;

}

}

private void btnLine\_Click(object sender, EventArgs e)

{

currentShape = ShapeType.Line;

}

private void btnCircle\_Click(object sender, EventArgs e)

{

currentShape = ShapeType.Circle;

}

private void btnRectangle\_Click(object sender, EventArgs e)

{

currentShape = ShapeType.Rectangle;

}

private void btnClear\_Click(object sender, EventArgs e)

{

canvasGraphics.Clear(Color.White);

pictureBox.Invalidate();

}

private void colorButton\_Click(object sender, EventArgs e)

{

using (ColorDialog colorDialog = new ColorDialog())

{

if (colorDialog.ShowDialog() == DialogResult.OK)

{

drawingPen.Color = colorDialog.Color;

colorButton.BackColor = colorDialog.Color;

}

}

}

private void trackBarWidth\_Scroll(object sender, EventArgs e)

{

drawingPen.Width = trackBarWidth.Value;

lblWidth.Text = $"Толщина: {trackBarWidth.Value}px";

}

protected override void OnFormClosed(FormClosedEventArgs e)

{

base.OnFormClosed(e);

canvasGraphics?.Dispose();

drawingPen?.Dispose();

canvasBitmap?.Dispose();

}

}

public enum ShapeType

{

Line,

Circle,

Rectangle

}

}

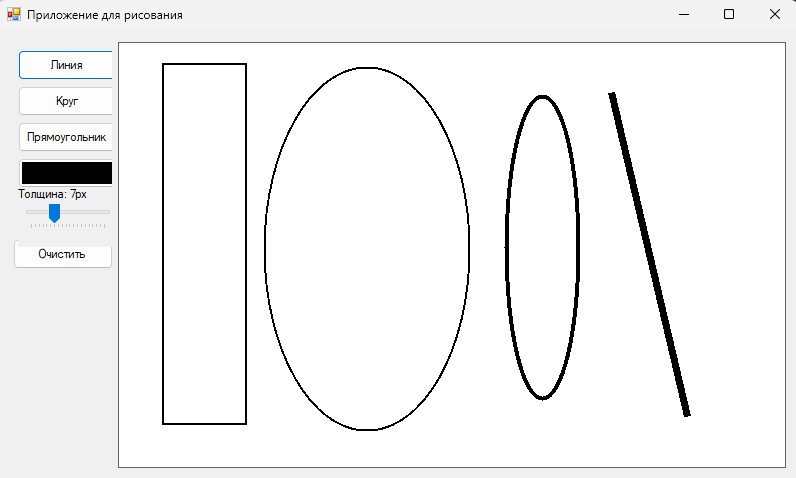


Рисунок 5.1.1 – Результат выполнения программы листинга 1

Задача 2. Разработайте текстовый редактор с возможностью открытия и сохранения текстовых файлов.

Листинг 2. Код программы на C#

public partial class Form1 : Form

{

private string currentFilePath = string.Empty;

private bool isModified = false;

public Form1()

{

InitializeComponent();

UpdateTitle();

}

private void UpdateTitle()

{

string fileName = string.IsNullOrEmpty(currentFilePath) ? "Новый файл" : Path.GetFileName(currentFilePath);

string modified = isModified ? " \*" : "";

this.Text = $"{fileName}{modified} - Текстовый редактор";

}

private void UpdateStatusBar()

{

int line = textBox.GetLineFromCharIndex(textBox.SelectionStart) + 1;

int column = textBox.SelectionStart - textBox.GetFirstCharIndexOfCurrentLine() + 1;

int totalLines = textBox.Lines.Length;

toolStripStatusLabel.Text = $"Строка: {line}, Колонка: {column} | Всего строк: {totalLines}";

}

private void NewFile()

{

if (CheckUnsavedChanges())

{

textBox.Clear();

currentFilePath = string.Empty;

isModified = false;

UpdateTitle();

UpdateStatusBar();

}

}

private void OpenFile()

{

if (CheckUnsavedChanges())

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 1;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

string filePath = openFileDialog.FileName;

string fileContent = File.ReadAllText(filePath);

textBox.Text = fileContent;

currentFilePath = filePath;

isModified = false;

UpdateTitle();

UpdateStatusBar();

toolStripStatusLabel.Text = $"Файл загружен: {Path.GetFileName(filePath)}";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при открытии файла: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

private void SaveFile()

{

if (string.IsNullOrEmpty(currentFilePath))

{

SaveFileAs();

}

else

{

SaveToFile(currentFilePath);

}

}

private void SaveFileAs()

{

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

saveFileDialog.FilterIndex = 1;

saveFileDialog.RestoreDirectory = true;

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = saveFileDialog.FileName;

SaveToFile(filePath);

}

}

}

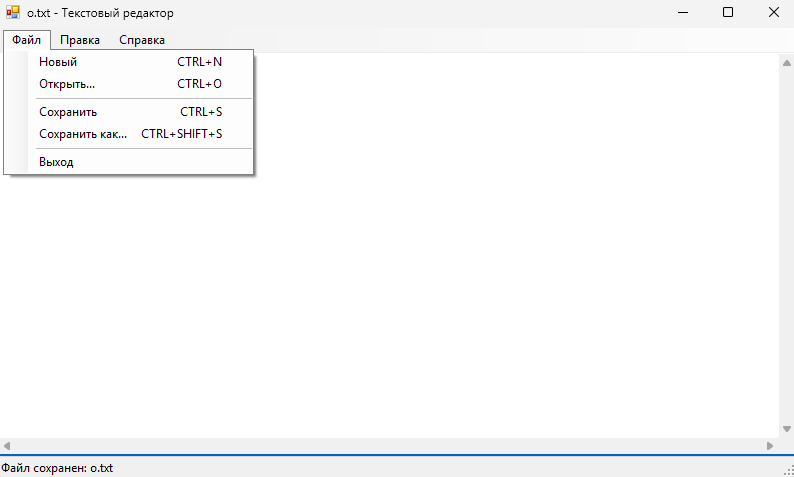


Рисунок 5.2.1 – Результат выполнения программы листинга 2

Задача 3. Создайте приложение для учета задач с использованием Windows Forms. Пользователь должен иметь возможность добавлять, удалять и отмечать задачи как выполненные.

Листинг 3. Код программы на C#

public partial class Form1 : Form

{

private List<TaskItem> tasks;

private int nextId = 1;

public Form1()

{

InitializeComponent();

InitializeTasks();

RefreshTaskList();

UpdateStatistics();

}

private void InitializeTasks()

{

tasks = new List<TaskItem>();

// Добавляем несколько примеров задач

AddTask("Изучить C#", "Освоить Windows Forms программирование", DateTime.Now.AddDays(7), Priority.High);

AddTask("Купить продукты", "Молоко, хлеб, яйца", DateTime.Now.AddDays(1), Priority.Medium);

AddTask("Сделать зарядку", "Утренняя гимнастика", DateTime.Now, Priority.Low);

}

private void AddTask(string title, string description, DateTime dueDate, Priority priority)

{

var task = new TaskItem

{

Id = nextId++,

Title = title,

Description = description,

DueDate = dueDate,

Priority = priority

};

tasks.Add(task);

}

private void btnAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtTitle.Text))

{

MessageBox.Show("Введите название задачи!", "Внимание",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var task = new TaskItem

{

Id = nextId++,

Title = txtTitle.Text.Trim(),

Description = txtDescription.Text.Trim(),

DueDate = dtpDueDate.Value,

Priority = (Priority)cmbPriority.SelectedIndex

};

tasks.Add(task);

ClearInputs();

RefreshTaskList();

UpdateStatistics();

txtTitle.Focus();

}

private void btnDelete\_Click(object sender, EventArgs e)

{

if (lstTasks.SelectedItem != null && lstTasks.SelectedItem is TaskItem selectedTask)

{

var result = MessageBox.Show(

$"Удалить задачу \"{selectedTask.Title}\"?",

"Подтверждение удаления",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

tasks.Remove(selectedTask);

RefreshTaskList();

UpdateStatistics();

}

}

else

{

MessageBox.Show("Выберите задачу для удаления!", "Внимание",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

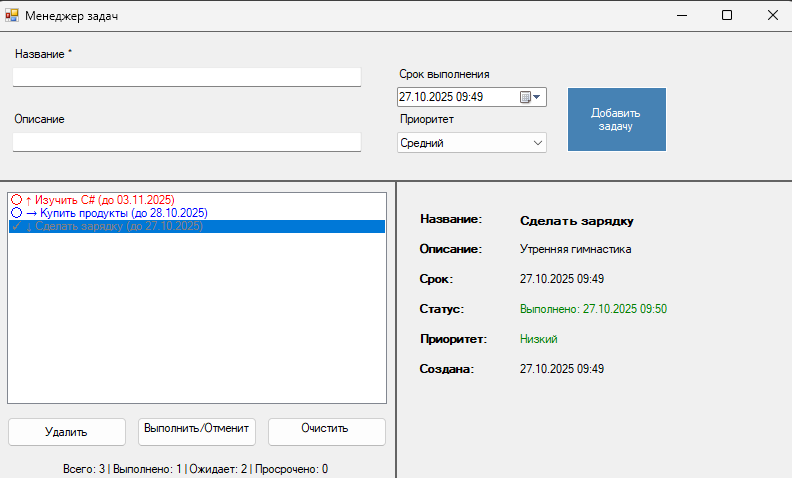


Рисунок 5.3.1 – Результат выполнения программы листинга 3

Задача 4. Реализуйте приложение для просмотра изображений. Пользователь должен иметь возможность выбирать изображение для просмотра и масштабировать его.

Листинг 4. Код программы на C#

public partial class Form1 : Form

{

private Image currentImage;

private float zoomFactor = 1.0f;

private const float ZOOM\_STEP = 0.25f;

private const float MIN\_ZOOM = 0.1f;

private const float MAX\_ZOOM = 5.0f;

private Point lastMousePosition;

private bool isPanning = false;

private Point imagePosition = Point.Empty;

public Form1()

{

InitializeComponent();

UpdateStatusBar();

UpdateZoomInfo();

}

private void OpenImage()

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "Image Files|\*.bmp;\*.jpg;\*.jpeg;\*.png;\*.gif;\*.tiff;\*.ico|All Files|\*.\*";

openFileDialog.FilterIndex = 1;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

// Освобождаем предыдущее изображение

currentImage?.Dispose();

// Загружаем новое изображение

currentImage = Image.FromFile(openFileDialog.FileName);

// Сбрасываем масштаб и позицию

zoomFactor = 1.0f;

imagePosition = Point.Empty;

// Обновляем интерфейс

pictureBox.Invalidate();

UpdateStatusBar();

UpdateZoomInfo();

// Обновляем заголовок окна

this.Text = $"Просмотр изображений - {Path.GetFileName(openFileDialog.FileName)}";

toolStripStatusLabel.Text = $"Загружено: {Path.GetFileName(openFileDialog.FileName)}";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке изображения: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void ZoomIn()

{

if (currentImage == null) return;

float newZoom = zoomFactor + ZOOM\_STEP;

if (newZoom <= MAX\_ZOOM)

{

zoomFactor = newZoom;

pictureBox.Invalidate();

UpdateZoomInfo();

}

}

private void ZoomOut()

{

if (currentImage == null) return;

float newZoom = zoomFactor - ZOOM\_STEP;

if (newZoom >= MIN\_ZOOM)

{

zoomFactor = newZoom;

pictureBox.Invalidate();

UpdateZoomInfo();

}

}

private void ZoomToFit()

{

if (currentImage == null) return;

// Вычисляем масштаб для полного отображения изображения

float zoomX = (float)pictureBox.Width / currentImage.Width;

float zoomY = (float)pictureBox.Height / currentImage.Height;

zoomFactor = Math.Min(zoomX, zoomY) \* 0.95f; // Небольшой отступ

imagePosition = Point.Empty;

pictureBox.Invalidate();

UpdateZoomInfo();

}

private void ZoomToActualSize()

{

if (currentImage == null) return;

zoomFactor = 1.0f;

imagePosition = Point.Empty;

pictureBox.Invalidate();

UpdateZoomInfo();

}

private void UpdateZoomInfo()

{

lblZoom.Text = $"Масштаб: {zoomFactor \* 100:F0}%";

btnZoomActual.Enabled = Math.Abs(zoomFactor - 1.0f) > 0.01f;

btnZoomFit.Enabled = currentImage != null;

}

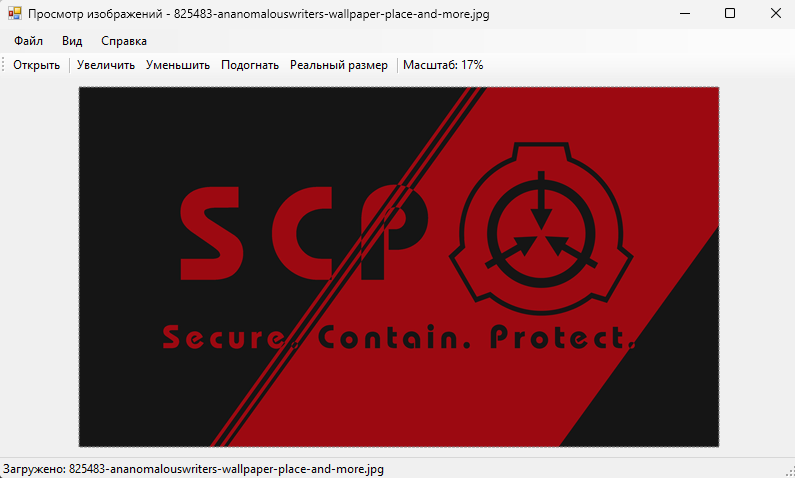


Рисунок 5.4.1 – Результат выполнения программы листинга 4

Задача 5. Создайте калькулятор с графическим интерфейсом. Пользователь должен иметь возможность выполнять арифметические операции.

Листинг 5. Код программы на C#

public partial class Form1 : Form

{

private double currentValue = 0;

private string currentOperation = "";

private bool isNewOperation = true;

private bool isOperationPending = false;

private double storedValue = 0;

public Form1()

{

InitializeComponent();

UpdateDisplay();

}

private void UpdateDisplay()

{

txtDisplay.Text = currentValue.ToString();

}

private void ClearAll()

{

currentValue = 0;

currentOperation = "";

isNewOperation = true;

isOperationPending = false;

storedValue = 0;

lblOperation.Text = "";

UpdateDisplay();

}

private void ClearEntry()

{

currentValue = 0;

isNewOperation = true;

UpdateDisplay();

}

private void NumberButton\_Click(object sender, EventArgs e)

{

if (sender is Button button)

{

string number = button.Text;

if (isNewOperation)

{

currentValue = 0;

isNewOperation = false;

}

if (currentValue == 0 && !txtDisplay.Text.Contains("."))

{

currentValue = double.Parse(number);

}

else

{

string currentText = currentValue.ToString();

currentText += number;

currentValue = double.Parse(currentText);

}

UpdateDisplay();

}

}

private void DecimalButton\_Click(object sender, EventArgs e)

{

if (isNewOperation)

{

currentValue = 0;

isNewOperation = false;

}

if (!txtDisplay.Text.Contains("."))

{

txtDisplay.Text += ".";

currentValue = double.Parse(txtDisplay.Text);

}

}

private void OperationButton\_Click(object sender, EventArgs e)

{

if (sender is Button button)

{

if (isOperationPending)

{

CalculateResult();

}

storedValue = currentValue;

currentOperation = button.Text;

isOperationPending = true;

isNewOperation = true;

lblOperation.Text = $"{storedValue} {currentOperation}";

}

}

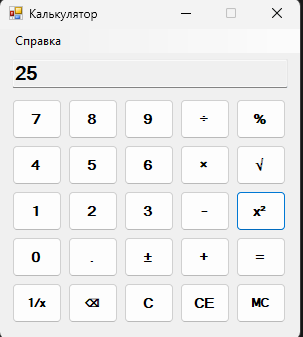


Рисунок 5.5.1 – Результат выполнения программы листинга 5

Модуль 6