Лабораторная работа №5

Разработка программ с использованием массивов

Цель работы: Научиться разрабатывать программы с использованием массивов с использованием среды Visual Studio.Net.

Вариант 4

**Задание 1.1** Обработка одномерных массивов.

Дан массив из n целых чисел. Написать программу, которая подсчитывает

количество пар соседних элементов массива, для которых сумма нечетная.

static void Main(string[] args)

{

// Запрос у пользователя ввода размера массива

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

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

// Создание массива заданного размера

int[] array = new int[n];

// Ввод элементов массива

Console.WriteLine("Введите элементы массива:");

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

{

Console.Write($"Элемент {i + 1}: ");

array[i] = int.Parse(Console.ReadLine());

}

// Вызов функции для подсчета пар

int count = CountOddSumPairs(array);

// Вывод результата

Console.WriteLine($"Количество пар соседних элементов с нечетной суммой: {count}");

}

static int CountOddSumPairs(int[] array)

{

int count = 0;

// Проход по массиву и проверка соседних элементов

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

{

// Проверка, является ли сумма соседних элементов нечетной

if ((array[i] + array[i + 1]) % 2 != 0)

{

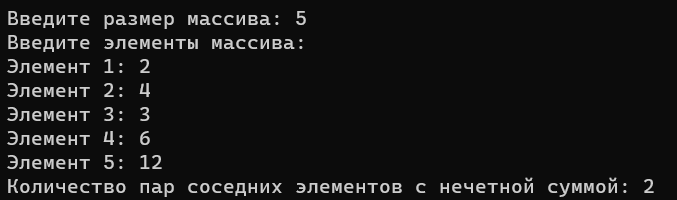
count++;

}

}

return count;

}



**Задание 1.2** Обработка одномерных массивов.

В целочисленный массив X(n) после каждого нечетного элемента вставить

максимальный простой элемент этого же массива. Определить среднее арифметическое простых элементов массива до и после вставки.

using System;

class Program

{

static void Main(string[] args)

{

// Запрос у пользователя ввода размера массива

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

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

// Создание массива заданного размера

int[] array = new int[n];

// Ввод элементов массива

Console.WriteLine("Введите элементы массива:");

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

{

Console.Write($"Элемент {i + 1}: ");

array[i] = int.Parse(Console.ReadLine());

}

// Определяем максимальный простой элемент и среднее арифметическое простых чисел до вставки

int maxPrime = FindMaxPrime(array);

double averageBefore = CalculateAverageOfPrimes(array);

Console.WriteLine($"Максимальный простой элемент: {maxPrime}");

Console.WriteLine($"Среднее арифметическое простых элементов до вставки: {averageBefore}");

// Модифицируем массив, вставляя максимальный простой элемент после каждого нечетного элемента

ModifyArray(ref array, maxPrime);

// Вычисляем среднее арифметическое простых элементов после вставки

double averageAfter = CalculateAverageOfPrimes(array);

// Вывод модифицированного массива

Console.WriteLine("Модифицированный массив:");

foreach (var item in array)

{

Console.Write(item + " ");

}

Console.WriteLine($"nСреднее арифметическое простых элементов после вставки: {averageAfter}");

}

static int FindMaxPrime(int[] array)

{

int maxPrime = int.MinValue;

foreach (var number in array)

{

if (IsPrime(number) && number > maxPrime)

{

maxPrime = number;

}

}

return maxPrime == int.MinValue ? 0 : maxPrime; // Возвращаем 0, если простых чисел нет

}

static bool IsPrime(int number)

{

if (number <= 1) return false;

for (int i = 2; i <= Math.Sqrt(number); i++)

{

if (number % i == 0) return false;

}

return true;

}

static double CalculateAverageOfPrimes(int[] array)

{

int sum = 0;

int count = 0;

foreach (var number in array)

{

if (IsPrime(number))

{

sum += number;

count++;

}

}

return count > 0 ? (double)sum / count : 0; // Если простых чисел нет, возвращаем 0

}

static void ModifyArray(ref int[] array, int maxPrime)

{

int n = array.Length;

// Проходим по массиву и вставляем максимальный простой элемент после каждого нечетного числа

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

{

if (array[i] % 2 != 0) // Если число нечетное

{

// Увеличиваем размер массива

Array.Resize(ref array, array.Length + 1);

// Сдвигаем элементы вправо для освобождения места

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

{

array[j] = array[j - 1];

}

// Вставляем максимальный простой элемент

array[i + 1] = maxPrime;

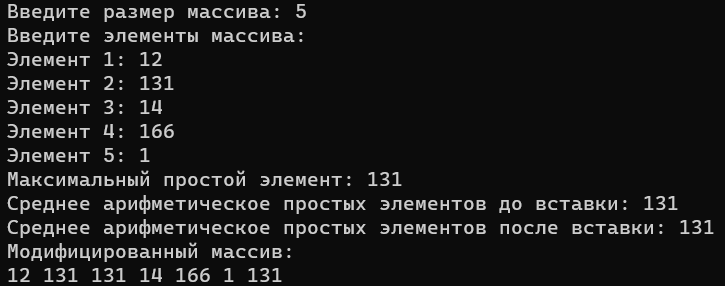
// Увеличиваем индекс, чтобы пропустить вставленный элемент

i++;

}

}

}

}

**Задание 2.** Обработка двумерных массивов

Задана матрица В(n,m). Определить количество столбцов, упорядоченных по

возрастанию.

using System;

class Program

{

static void Main()

{

// Запрашиваем у пользователя количество строк матрицы

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

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

// Запрашиваем у пользователя количество столбцов матрицы

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

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

// Создаем двумерный массив (матрицу) заданного размера

int[,] matrix = new int[rows, cols];

// Ввод элементов матрицы пользователем

Console.WriteLine("Введите элементы матрицы (по строкам):");

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

{

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

{

// Запрашиваем элемент матрицы по индексам [i, j]

Console.Write($"Элемент [{i}, {j}]: ");

matrix[i, j] = int.Parse(Console.ReadLine());

}

}

// Подсчитываем количество упорядоченных по возрастанию столбцов

int count = CountSortedColumns(matrix);

// Выводим результат на экран

Console.WriteLine($"Количество упорядоченных по возрастанию столбцов: {count}");

}

// Метод для подсчета количества упорядоченных по возрастанию столбцов

static int CountSortedColumns(int[,] matrix)

{

// Проверяем, что матрица не null и имеет ненулевые размеры

if (matrix == null || matrix.GetLength(0) == 0 || matrix.GetLength(1) == 0)

return 0;

// Получаем количество строк и столбцов матрицы

int rows = matrix.GetLength(0);

int cols = matrix.GetLength(1);

// Переменная для подсчета упорядоченных столбцов

int sortedColumnsCount = 0;

// Перебираем каждый столбец матрицы

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

{

bool isSorted = true; // Предполагаем, что столбец упорядочен

// Проверяем, упорядочен ли текущий столбец

for (int row = 1; row < rows; row++)

{

// Если текущий элемент меньше предыдущего, столбец не упорядочен

if (matrix[row, col] < matrix[row - 1, col])

{

isSorted = false;

break; // Выходим из цикла, если нашли несоответствие

}

}

// Если столбец упорядочен, увеличиваем счетчик

if (isSorted)

{

sortedColumnsCount++;

}

}

// Возвращаем общее количество упорядоченных столбцов

return sortedColumnsCount;

}

}

