**Шумилов Лев, РИС-20-1б, Вариант №25**

**Отчёт по практической работе №5 «Функции и массивы»**

Цель работы: получить практические навыки при работе с одномерными, многомерными и рваными массивами, практические навыки при работе с функциями, передачи данных, получение результатов их работы разными способами.

Дополнительно, реализовать диалоговое консольное меню приложения.

Задачи практической работы:

1. Сформировать динамический одномерный массив, заполнить его случайными числами и вывести на печать;
2. Сформировать динамический двумерный массив, заполнить его случайными числами и вывести на печать;
3. Сформировать динамический рваный массив, заполнить его случайными числами и вывести на печать;
4. Выполнить указанные в варианте задания и вывести полученные массивы на печать;
5. Продемонстрировать в функциях использование параметров разных типов и различные способы организации функций (без параметров, с параметрами);

Задания варианта №25

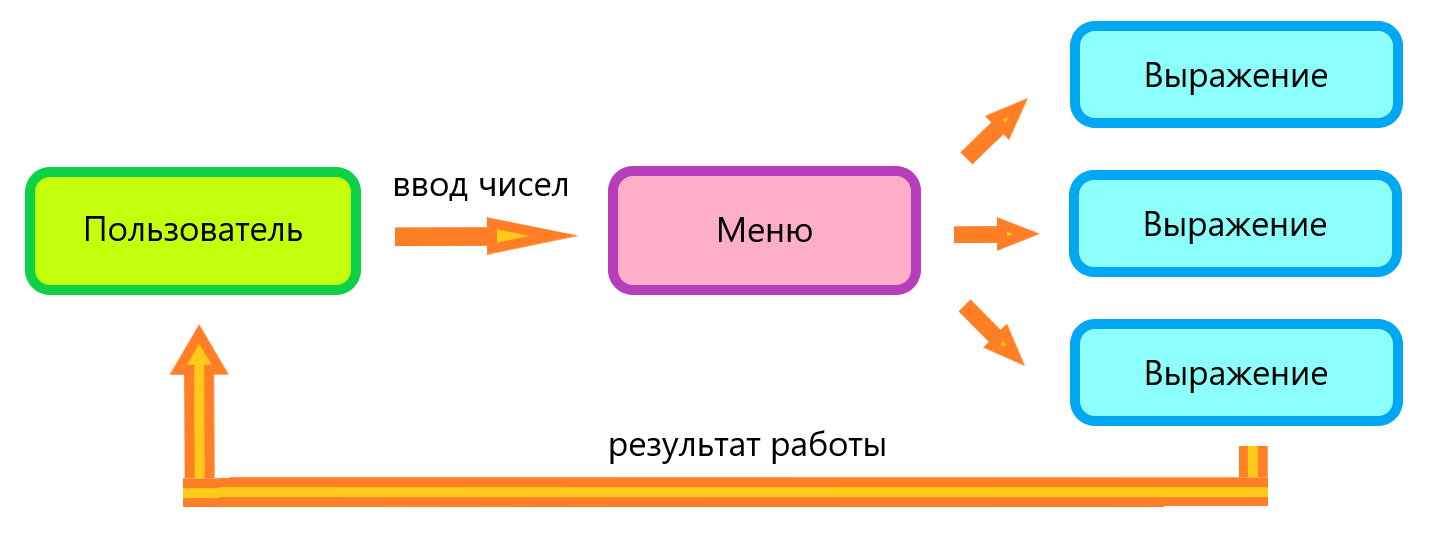
|  |  |  |
| --- | --- | --- |
| Одномерный массив | Двумерный массив | Рваный массив |
| Удалить все элементы с нечётным индексом | Добавить новый столбец после столбца с максимальным значением | Добавить новую строку к текущему рваному массиву |

**Анализ задач:**

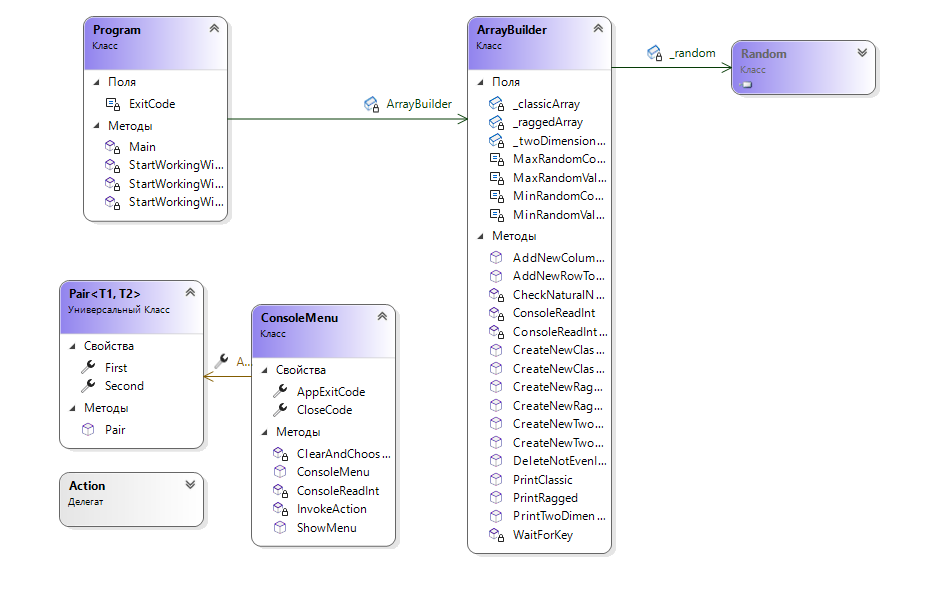
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Описание функции | Входные данные | Классы входных данных | Выходные данные |
| 1 | Вызов функции по номеру из задачи | Целое число | Число входит в заданный диапазон | Выполнение нужной функции |
| Число не входит в заданный диапазон | Сообщение об ошибке |
| 0 | Переход в общее меню |
| 2 | Формирование одномерного массива через ДСЧ |  | | |
| 3 | Формирование двумерного массива через ДСЧ |
| 4 | Формирование рваного массива через ДСЧ |
| 5 | Удаление всех элементов с нечётным индексом | Одномерный массив | Массив пустой | Сообщение об ошибке |
| Массив содержит 1 элемент | Пустой массив |
| Массив содержит чётное кол-во элементов | Массив, в котором элементов в два раза меньше |
| Массив содержит нечётное кол-во элементов | Массив, в котором в два раза меньше элементов (округление в меньшую сторону 5 => 2) |
| 6 | Добавление нового столбца после столбца с максимальным значением | Двумерный массив | Массив пустой | Сообщение об ошибке |
| Массив не пустой | Массив, где больше столбцов на 1 |
| 7 | Добавление новой строки к текущему рваному массиву | Рваный массив | Массив пустой | Сообщение об ошибке |
| Массив не пустой | Новый рваный массив, где есть новая строка |

**Проектирование:**

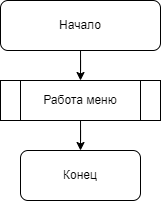
Взаимодействие пользователя с консольным приложением будет производиться посредством специального, меню, позволяющим выбрать номер выражения, с которым пользователь желает произвести работу.

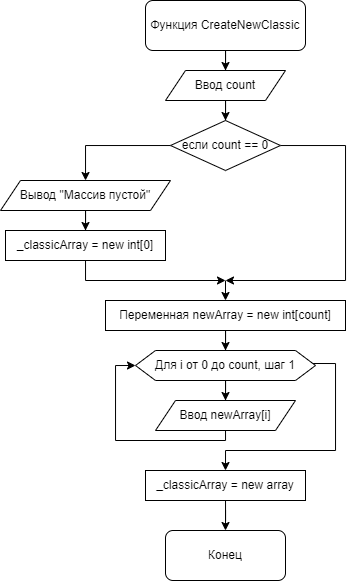


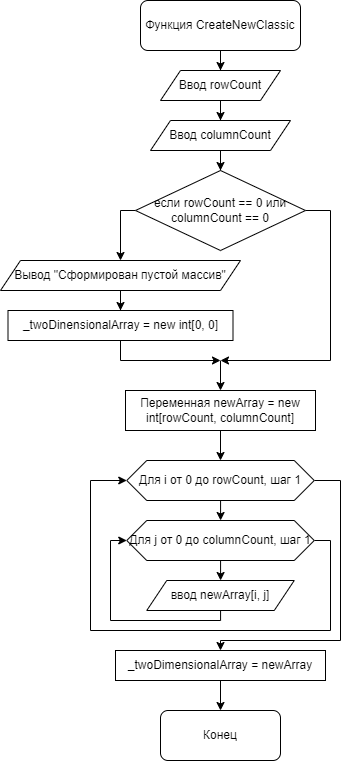
|  |  |  |  |
| --- | --- | --- | --- |
| № | Метод | Входные данные | Выходные данные |
| 1 | CreateNewClassicByUserInput | ничего | Одномерный массив |
| 2 | CreateNewClassicByRandom | ничего | Одномерный массив |
| 3 | PrintClassic | Одномерный массив | Вывод на экран |
| 4 | CreateNewTwoDimensionalByUserInput | ничего | Двумерный массив |
| 5 | CreateNewTwoDimensionalByRandom | ничего | Двумерный массив |
| 6 | PrintTwoDimensional | Двумерный массив | Вывод на экран |
| 7 | CreateNewRaggedByUserInput | ничего | Рваный массив |
| 8 | CreateNewRaggedByRandom | ничего | Рваный массив |
| 9 | PrintRagged | Рваный массив | Вывод на экран |
| 10 | DeleteNotEvenIndexFromClassic | Одномерный массив | Одномерный массив |
| 11 | AddNewColumnAfterMaxValueToTwoDimension | Двумерный массив | Двумерный массив |
| 12 | AddNewRowToRagged | Рваный массив | Рваный массив |

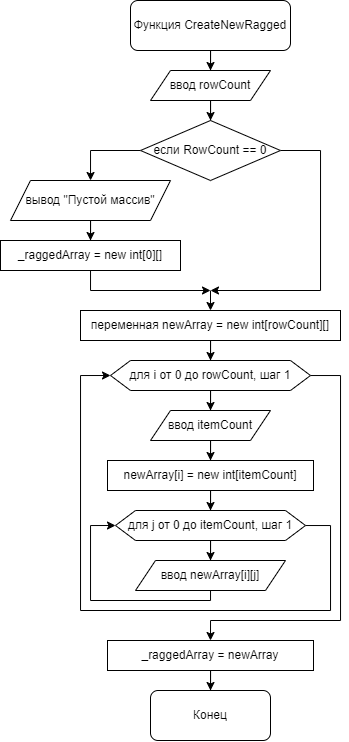


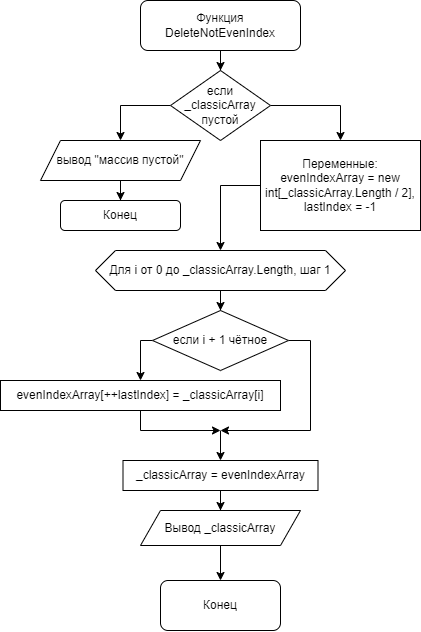
**Блок-схемы:**

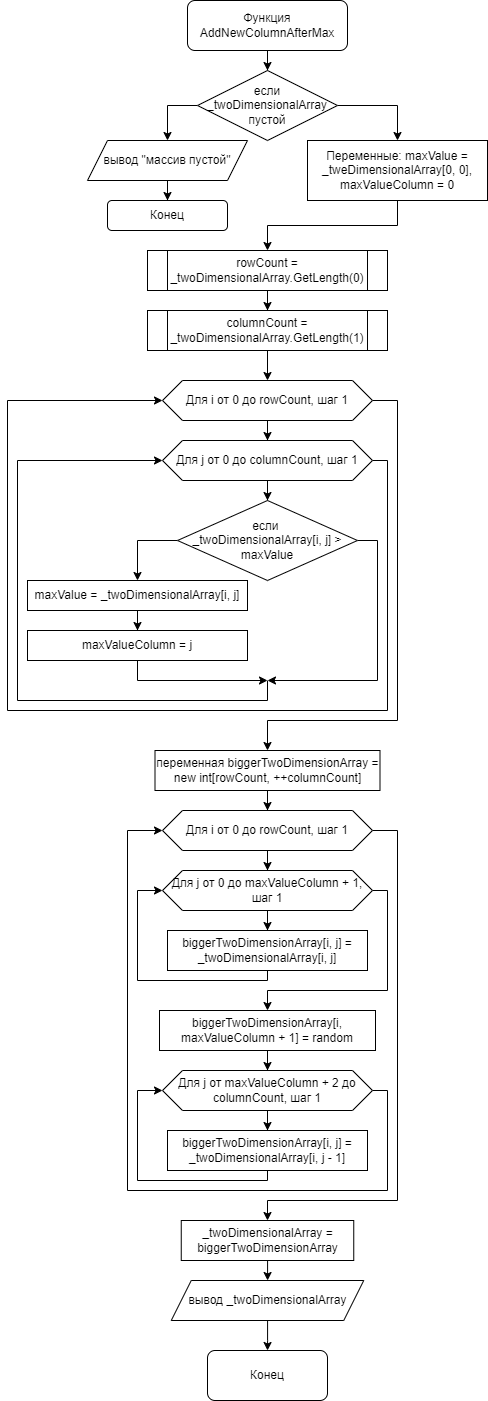
****

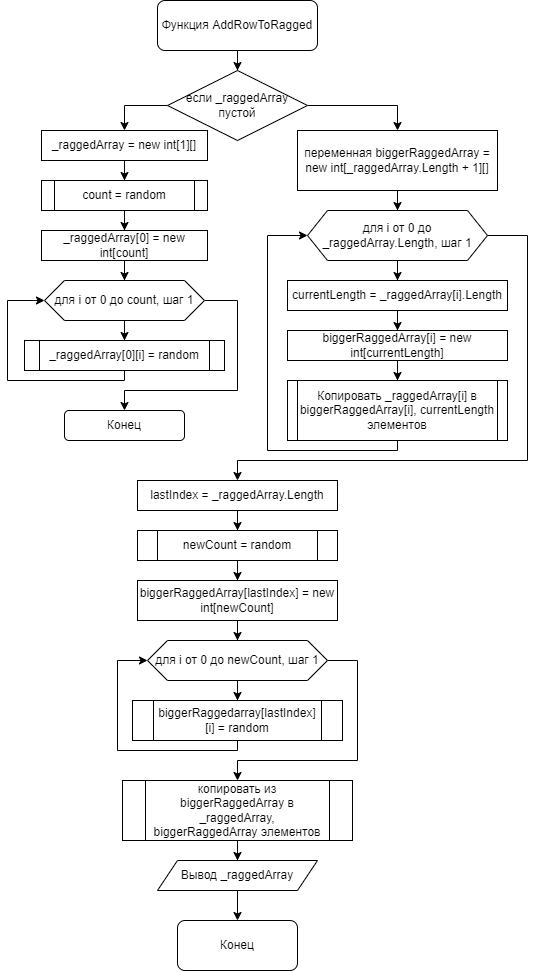
****

****

****

****

****

****

**Полный исходный код на языке C#**

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LaboratoryWorkNo5

{

class Program

{

const int ExitCode = 0;

static readonly ArrayBuilder ArrayBuilder = new ArrayBuilder();

static void Main(string[] args)

{

var actions = new Pair<Action, string>[]

{

new Pair<Action, string>(

StartWorkingWithClassicArrays,

"Работа с одномерными массивами"),

new Pair<Action, string>(

StartWorkingWithTwoDimArrays,

"Работа с двумерными массивами"),

new Pair<Action, string>(

StartWorkingWithRaggedArrays,

"Работа с рваными массивами"),

};

var mainMenu = new ConsoleMenu(actions, 0, 0);

mainMenu.ShowMenu();

}

static void StartWorkingWithClassicArrays()

{

var actions = new Pair<Action, string>[]

{

new Pair<Action, string>(

ArrayBuilder.CreateNewClassicByUserInput,

"Сформировать новый массив вручную"),

new Pair<Action, string>(

ArrayBuilder.CreateNewClassicByRandom,

"Сформировать новый массив случайным образом"),

new Pair<Action, string>(

ArrayBuilder.DeleteNotEvenIndexFromClassic,

"Удалить все элементы с нечётным индексом"),

new Pair<Action, string>(

ArrayBuilder.PrintClassic,

"Вывести элементы массива на экран"),

};

var menu = new ConsoleMenu(actions);

menu.ShowMenu();

}

static void StartWorkingWithTwoDimArrays()

{

var actions = new Pair<Action, string>[]

{

new Pair<Action, string>(

ArrayBuilder.CreateNewTwoDimensionalByUserInput,

"Сформировать новый двумерный массив вручную"),

new Pair<Action, string>(

ArrayBuilder.CreateNewTwoDimensionalByRandom,

"Сформировать новый двумерный массив случайным образом"),

new Pair<Action, string>(

ArrayBuilder.AddNewColumnAfterMaxValueToTwoDimension,

"Добавить новый столбец после столбца с максимальным значением"),

new Pair<Action, string>(

ArrayBuilder.PrintTwoDimensional,

"Вывести элементы массива на экран"),

};

var menu = new ConsoleMenu(actions);

menu.ShowMenu();

}

static void StartWorkingWithRaggedArrays()

{

var actions = new Pair<Action, string>[]

{

new Pair<Action, string>(

ArrayBuilder.CreateNewRaggedByUserInput,

"Сформировать новый рваный массив вручную"),

new Pair<Action, string>(

ArrayBuilder.CreateNewRaggedByRandom,

"Сформировать новый рваный массив случайным образом"),

new Pair<Action, string>(

ArrayBuilder.AddNewRowToRagged,

"Добавить новую строку к текущему рваному массиву"),

new Pair<Action, string>(

ArrayBuilder.PrintRagged,

"Вывести элементы массива на экран"),

};

var menu = new ConsoleMenu(actions);

menu.ShowMenu();

}

}

}

Pair.cs

namespace LaboratoryWorkNo5

{

public class Pair<T1, T2>

{

public T1 First { get; set; }

public T2 Second { get; set; }

public Pair(T1 first, T2 second)

{

First = first;

Second = second;

}

}

}

ConsoleMenu.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Console;

namespace LaboratoryWorkNo5

{

public delegate void Action();

public class ConsoleMenu

{

public Pair<Action, string>[] Actions { get; set; }

public int CloseCode { get; set; }

public int AppExitCode { get; set; }

public ConsoleMenu(Pair<Action, string>[] actions, int closeCode = 0, int appExitCode = -1)

{

if (actions == null)

throw new ArgumentNullException("Меню без действий - не меню вовсе");

if (closeCode > 0 && closeCode < actions.Length)

throw new ArgumentException("Код \"Назад\" совпадает с кодом одного из действий");

if (appExitCode > 0 && appExitCode < actions.Length)

throw new ArgumentException("Код \"Выход из приложения\" совпадает с кодом одного из действий");

Actions = new Pair<Action, string>[actions.Length];

Array.Copy(actions, Actions, actions.Length);

CloseCode = closeCode;

AppExitCode = appExitCode;

}

public void ShowMenu()

{

int action = -1;

do

{

action = ClearAndChooseAction();

InvokeAction(action);

if (action == AppExitCode)

Environment.Exit(0);

}

while (action != CloseCode);

}

private int ClearAndChooseAction()

{

SetWindowPosition(0, WindowTop - (WindowTop - CursorTop));

WriteLine();

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

WriteLine($"{i + 1}. {Actions[i].Second}");

WriteLine($"\n{AppExitCode}. Выход из приложения");

if (AppExitCode != CloseCode)

WriteLine($"{CloseCode}. Назад");

return ConsoleReadInt("\nВыберите действие: ");

}

private int ConsoleReadInt(string inputMessage = "")

{

Write(inputMessage);

int inputNumber;

while (!int.TryParse(ReadLine(), out inputNumber))

WriteLine(

"\nВнимание: введённое выражение не является целым числом!\n" +

"Повторите попытку ввода!!!\n");

return inputNumber;

}

private void InvokeAction(int actionIndex)

{

if (actionIndex < 1 || actionIndex > Actions.Length)

{

if (actionIndex != CloseCode && actionIndex != AppExitCode)

{

WriteLine($"Ошибка: действие с номером {actionIndex} отсутствует." +

"\nНажмите любую клавишу, чтобы продолжить...");

ReadKey();

}

return;

}

Actions[--actionIndex].First();

}

}

}

ArrayBuilder.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Console;

namespace LaboratoryWorkNo5

{

public enum Create

{

ByRandom,

ByUserInput,

}

public class ArrayBuilder

{

private const int MinRandomCount = 1;

private const int MaxRandomCount = 12;

private const int MinRandomValue = -100;

private const int MaxRandomValue = 100;

private int[] \_classicArray;

private int[,] \_twoDimensionalArray;

private int[][] \_raggedArray;

private static readonly Random \_random = new Random();

private void WaitForKey(ConsoleKey key)

{

WriteLine();

do

{

WriteLine($"Нажмите {key}, чтобы продолжить");

}

while (ReadKey(true).Key != key);

}

private void CheckNaturalNumbers(params int[] numbers)

{

foreach (int n in numbers)

{

if (n < 0)

{

throw new ArgumentException(

"кол-во элементов в массиве не может быть отрицательным");

}

}

}

public void CreateNewClassicByUserInput()

{

int count = ConsoleReadIntNonNegative("Кол-во элементов: ");

CheckNaturalNumbers(count);

if (count == 0)

{

WriteLine("Внимание: сформирован ПУСТОЙ массив!!!");

\_classicArray = new int[0];

}

var newArray = new int[count];

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

newArray[i] = ConsoleReadInt($"A[{i}]: ");

\_classicArray = newArray;

}

public void CreateNewClassicByRandom()

{

int count = \_random.Next(MinRandomCount, MaxRandomCount);

var newArray = new int[count];

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

{

var newValue = \_random.Next(MinRandomValue, MaxRandomValue);

newArray[i] = newValue;

}

\_classicArray = newArray;

PrintClassic();

}

public void PrintClassic()

{

if (\_classicArray == null || \_classicArray.Length == 0)

{

WriteLine("\* Массив пустой...");

WaitForKey(ConsoleKey.Enter);

return;

}

WriteLine($"Кол-во элементов = {\_classicArray.Length}:\n");

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

WriteLine($"A[{i + 1}] = {\_classicArray[i]}");

WaitForKey(ConsoleKey.Enter);

}

public void CreateNewTwoDimensionalByUserInput()

{

int rowCount = ConsoleReadIntNonNegative($"Кол-во строк: ");

int columnCount = ConsoleReadIntNonNegative($"Кол-во столбцов: ");

CheckNaturalNumbers(rowCount, columnCount);

if (rowCount == 0 || columnCount == 0)

{

WriteLine("Внимание: сформирован ПУСТОЙ массив!!!");

\_twoDimensionalArray = new int[0, 0];

}

var newArray = new int[rowCount, columnCount];

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

{

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

newArray[i, j] = ConsoleReadInt($"A[{i}, {j}]: ");

WriteLine();

}

\_twoDimensionalArray = newArray;

}

public void CreateNewTwoDimensionalByRandom()

{

int rowCount = \_random.Next(MinRandomCount, MaxRandomCount);

int columnCount = \_random.Next(MinRandomCount, MaxRandomCount);

var newArray = new int[rowCount, columnCount];

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

{

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

{

var newValue = \_random.Next(MinRandomValue, MaxRandomValue);

newArray[i, j] = newValue;

}

}

\_twoDimensionalArray = newArray;

PrintTwoDimensional();

}

public void PrintTwoDimensional()

{

if (\_twoDimensionalArray == null || \_twoDimensionalArray.Length == 0)

{

WriteLine("\* Массив пустой...");

WaitForKey(ConsoleKey.Enter);

return;

}

WriteLine($"Кол-во элементов = {\_twoDimensionalArray.Length}:\n");

WriteLine($"Кол-во строк = {\_twoDimensionalArray.GetLength(0)}");

WriteLine($"Кол-во столбцов = {\_twoDimensionalArray.GetLength(1)}\n");

for (int i = 0; i < \_twoDimensionalArray.GetLength(0); i++)

{

var outputLine = "";

for (int j = 0; j < \_twoDimensionalArray.GetLength(1); j++)

outputLine += $"{\_twoDimensionalArray[i, j]}\t";

WriteLine(outputLine);

}

WaitForKey(ConsoleKey.Enter);

}

public void CreateNewRaggedByUserInput()

{

int rowCount = ConsoleReadIntNonNegative($"Кол-во подмассивов рваного массива: ");

CheckNaturalNumbers(rowCount);

if (rowCount == 0)

{

WriteLine("Внимание: сформирован ПУСТОЙ массив!!!");

\_raggedArray = new int[0][];

}

var newArray = new int[rowCount][];

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

{

var itemCount = ConsoleReadIntNonNegative($"Кол-во элементов подмассива: ");

CheckNaturalNumbers(itemCount);

newArray[i] = new int[itemCount];

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

newArray[i][j] = ConsoleReadInt($"A[{i}][{j}]: ");

WriteLine();

}

\_raggedArray = newArray;

}

public void CreateNewRaggedByRandom()

{

int rowCount = \_random.Next(MinRandomCount, MaxRandomCount);

var newArray = new int[rowCount][];

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

{

var itemCount = \_random.Next(MinRandomCount, MaxRandomCount);

newArray[i] = new int[itemCount];

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

{

int newValue = \_random.Next(MinRandomValue, MaxRandomValue);

newArray[i][j] = newValue;

}

}

\_raggedArray = newArray;

PrintRagged();

}

public void PrintRagged()

{

if (\_raggedArray == null || \_raggedArray.Length == 0)

{

WriteLine("\* Массив пустой...");

WaitForKey(ConsoleKey.Enter);

return;

}

WriteLine($"Кол-во подмассивов = {\_raggedArray.Length}\n");

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

{

Write($"Элементов = {\_raggedArray[i].Length}: ");

var outputline = "";

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

outputline += $"{\_raggedArray[i][j]}\t";

WriteLine(outputline);

}

WaitForKey(ConsoleKey.Enter);

}

public void DeleteNotEvenIndexFromClassic()

{

if (\_classicArray == null || \_classicArray.Length == 0)

{

WriteLine("Массив пустой, извлечь элементы невозможно!!!");

WaitForKey(ConsoleKey.Enter);

return;

}

WriteLine("\nДо:");

PrintClassic();

var evenIndexArray = new int[\_classicArray.Length / 2];

int lastIndex = -1;

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

if ((i + 1) % 2 == 0)

evenIndexArray[++lastIndex] = \_classicArray[i];

\_classicArray = evenIndexArray;

WriteLine("\nПосле:");

PrintClassic();

}

public void AddNewColumnAfterMaxValueToTwoDimension()

{

if (\_twoDimensionalArray == null || \_twoDimensionalArray.Length == 0)

{

WriteLine("Массив пустой, найти максимальное значение невозможно!!!");

WaitForKey(ConsoleKey.Enter);

return;

}

WriteLine("\nДо:");

PrintTwoDimensional();

int maxValue = \_twoDimensionalArray[0, 0];

int maxValueColumn = 0;

int rowCount = \_twoDimensionalArray.GetLength(0);

int columnCount = \_twoDimensionalArray.GetLength(1);

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

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

if (\_twoDimensionalArray[i, j] > maxValue)

{

maxValue = \_twoDimensionalArray[i, j];

maxValueColumn = j;

WriteLine($"Новый элемент с мaкс-значением: столбец {maxValueColumn + 1}; x = {maxValue}");

}

var biggerTwoDimensionArray = new int[rowCount, ++columnCount];

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

{

for (int j = 0; j < maxValueColumn + 1; j++)

biggerTwoDimensionArray[i, j] = \_twoDimensionalArray[i, j];

biggerTwoDimensionArray[i, maxValueColumn + 1] = \_random.Next(MinRandomValue, MaxRandomValue);

for (int j = maxValueColumn + 2; j < columnCount; j++)

biggerTwoDimensionArray[i, j] = \_twoDimensionalArray[i, j - 1];

}

\_twoDimensionalArray = biggerTwoDimensionArray;

WriteLine("\nПосле:");

PrintTwoDimensional();

}

public void AddNewRowToRagged()

{

if (\_raggedArray == null)

{

\_raggedArray = new int[1][];

int count = \_random.Next(MinRandomCount, MaxRandomCount);

\_raggedArray[0] = new int[count];

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

\_raggedArray[0][i] = \_random.Next(MinRandomValue, MaxRandomValue);

return;

}

WriteLine("До:");

PrintRagged();

var biggerRaggedArray = new int[\_raggedArray.Length + 1][];

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

{

int currentLength = \_raggedArray[i].Length;

biggerRaggedArray[i] = new int[currentLength];

Array.Copy(\_raggedArray[i], biggerRaggedArray[i], currentLength);

}

int lastIndex = \_raggedArray.Length;

int newCount = \_random.Next(MinRandomCount, MaxRandomCount);

biggerRaggedArray[lastIndex] = new int[newCount];

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

biggerRaggedArray[lastIndex][i] = \_random.Next(MinRandomValue, MaxRandomValue);

\_raggedArray = new int[biggerRaggedArray.Length][];

Array.Copy(biggerRaggedArray, \_raggedArray, biggerRaggedArray.Length);

WriteLine("После:");

PrintRagged();

}

private int ConsoleReadInt(string inputMessage = "")

{

Write(inputMessage);

int inputNumber = 0;

while (!int.TryParse(ReadLine(), out inputNumber))

Write("\nВнимание: введённое выражение не является целым числом!\n" +

"Повторите попытку ввода!!!\n\n >> ");

return inputNumber;

}

private int ConsoleReadIntNonNegative(string inputMessage = "")

{

Write(inputMessage);

int inputNumber = 0;

var isSuit = false;

while (!isSuit)

{

var isValid = int.TryParse(ReadLine(), out inputNumber);

var isZeroOrMore = inputNumber >= 0;

isSuit = isValid && isZeroOrMore;

if (!isValid)

Write("\nВнимание: введённое выражение не является целым числом!\n" +

"Повторите попытку ввода!!!\n\n >> ");

if (!isZeroOrMore)

Write("\nВнимание: количество не может быть отрицательным!\n" +

"Повторите попытку ввода!!!\n\n >> ");

}

return inputNumber;

}

}

}