Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №4

По дисциплине

“Объектно-ориентированное программирование”

# Тема “Функции и массивы”

Вариант 15

Выполнил работу

студент группы РИС-19-1б

Миннахметов Э.Ю.

Проверила

доцент кафедры ИТАС

Викентьева О.Л.

Работу выполнил:

Пермь 2020

**1 ПОСТАНОВКА ЗАДАЧИ**

1. Сформировать динамический одномерный массив, заполнить его случайными числами и вывести на печать.
2. Выполнить указанное в варианте задание и вывести полученный массив на печать.
3. Сформировать динамический двумерный массив, заполнить его случайными числами и вывести на печать.
4. Выполнить указанное в варианте задание и вывести полученный массив на печать.
5. Сформировать динамический двумерный массив, заполнить его случайными числами и вывести на печать.
6. Выполнить указанное в варианте задание и вывести полученный массив на печать.
7. **При реализации функций необходимо продемонстрировать использование параметров разных типов и различные способы организации функций (параметры по умолчанию, перегрузку функций, и .т.д.)**

|  |  |  |
| --- | --- | --- |
| Одномерный массив | Двумерный массив | Рваный массив |
| Добавить К элементов, начиная с номера N | Удалить все четные строки | Добавить строку в конец массива |

**2 ОПИСАНИЕ ЭТАПА ПРОЕКТИРОВАНИЯ**

Лучше всего будет разделить программный код на ядро (Kernel), в котором будут храниться функции работы с данными, уровень работы с командной строкой (CLI) и точку входа в программу (Main).

Содержание ядра:

* void GetValid(out int number, GetNumber GetNum, char simbol, int top) - позволит получать значения, прошедшие определенное правило проверки на нерушимость логики программы;
* void CheckArray<T>(T array) – будет проверять ссылки на массив на «ненулевость»;
* void Formation1D(GetNumber GetNum), void Formation2D(GetNumber GetNum) и void FormationRagged(GetNumber GetNum) – создают одномерный, двумерный и рванный массивы соответственно;
* void Addition(ModeGetNumber Mode), void DeleteEven() и void AdditionLine(ModeGetNumber Mode) – добавление n значений в одномерный массив, удаление четных строк из двумерного массива и добавление строки в рванный массив соответственно.

Уровень командной строки:

* void Run()– запуск программы с необходимыми проверками;
* void ReadNum(out int number, char simbol, int i, int j), void RandNum(out int number, char simbol, int i, int j) – получение чисел с ввода или случайного соответственно;
* GetNumber GetMode() – получение способа получения числа (ввод или случайное);
* void Output(int[] array), void Output(int[,] array), void Output(int[][] array) – набор перегруженных функций для вывода одномерного, двумерного и рваного массивов;
* Output1, Output2, Output3, Formation1, Formation2, Formation3, Operation1, Operation2, Operation3 – функции-обертки для вывода, создания и операций над массивами;
* Task GetTask()– получение функции-обертки для выполнения задачи.

**3 ИСХОДНЫЙ КОД ПРОГРАММЫ**

**3.1 Kernel.cs**

using System;

namespace Lab5

{

delegate void GetNumber(out int number, char simbol, int i = -1, int j = -1);

delegate GetNumber ModeGetNumber();

delegate bool IsValidate(int x, int top);

class NullArrayException : Exception { }

class CleanArrayException : Exception { }

class NullFunctionException : Exception { }

class Kernel

{

public static int[] array1D { get; set; }

public static int[,] array2D { get; set; }

public static int[][] arrayRagged { get; set; }

private static IsValidate IsValid = (x, top) => (x >= 0 && x <= top);

// Получение валидного числа

private static void GetValid(out int number, GetNumber GetNum, char simbol, int top)

{

while (true)

{

GetNum(out number, simbol);

if (IsValid(number, top))

break;

else

CLI.IncorrectValue();

}

}

public static void CheckArray<T>(T array)

{

if (array == null)

throw new NullArrayException();

}

public static void Formation1D(GetNumber GetNum)

{

GetValid(out int n, GetNum, CLI.c\_cN, CLI.c\_iMaxInt);

array1D = new int[n];

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

GetNum(out array1D[i], CLI.c\_cA, i);

}

public static void Formation2D(GetNumber GetNum)

{

GetValid(out int n, GetNum, CLI.c\_cN, CLI.c\_iMaxInt);

GetValid(out int m, GetNum, CLI.c\_cM, CLI.c\_iMaxInt);

array2D = new int[n, m];

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

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

GetNum(out array2D[i, j], CLI.c\_cA, i, j);

}

public static void FormationRagged(GetNumber GetNum)

{

GetValid(out int n, GetNum, CLI.c\_cN, CLI.c\_iMaxInt);

arrayRagged = new int[n][];

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

{

GetValid(out int m, GetNum, CLI.c\_cM, CLI.c\_iMaxInt);

arrayRagged[i] = new int[m];

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

GetNum(out arrayRagged[i][j], CLI.c\_cA, i, j);

}

}

// Добавление N элементов после K элемента в одномерный массив

public static void Addition(ModeGetNumber Mode)

{

CheckArray(array1D);

GetNumber GetNum = Mode();

int oldSize = array1D.Length;

GetValid(out int n, GetNum, CLI.c\_cN, CLI.c\_iMaxInt - oldSize);

GetValid(out int k, GetNum, CLI.c\_cK, oldSize - 1);

int newSize = oldSize + n;

int[] temp = array1D;

array1D = new int[newSize];

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

array1D[i] = temp[i];

for (int i = k, j = k + n; i < j; ++i)

GetNum(out array1D[i], CLI.c\_cA, i);

for (int i = k + n; i < newSize; ++i)

array1D[i] = temp[i - n];

}

// Удаление четных строк в двумерном массиве

public static void DeleteEven()

{

CheckArray(array2D);

int[,] temp = array2D;

int n = array2D.GetUpperBound(0) + 1;

int k = array2D.Length / n;

int m = n / 2;

array2D = new int[m, k];

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

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

array2D[i, j] = temp[i \* 2 + 1, j];

if(array2D.Length == 0)

{

array2D = null;

throw new CleanArrayException();

}

}

// Добавление строки в конец рваного массива

public static void AdditionLine(ModeGetNumber Mode)

{

CheckArray(arrayRagged);

GetNumber GetNum = Mode();

int[][] temp = arrayRagged;

int n = temp.Length;

arrayRagged = new int[n + 1][];

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

arrayRagged[i] = temp[i];

GetValid(out int m, GetNum, CLI.c\_cM, CLI.c\_iMaxInt);

arrayRagged[n] = new int[m];

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

GetNum(out arrayRagged[n][i], CLI.c\_cA, n, i);

}

}

}

**3.2 CLI.cs**

using System;

namespace Lab5

{

delegate void Task();

class CLI

{

public const char c\_cN = 'N';

public const char c\_cM = 'M';

public const char c\_cK = 'K';

public const char c\_cA = 'A';

private const int c\_iMinK = 1;

private const int c\_iMaxK = 5;

private const int c\_iMinArray = 10;

private const int c\_iMaxArray = 20;

private const int c\_iMinNumber = 100;

private const int c\_iMaxNumber = 999;

public const int c\_iMaxInt = 2147483647;

private const string c\_sElem = "{0} ";

private const string c\_sReadNumber = "Введите {0}: ";

private const string c\_sReadNumberA = "Введите {0}{1}: ";

private const string c\_sReadNumberA2 = "Введите {0}({1},{2}): ";

private const string c\_sIncorrectValue = "Некорректное значение!";

private const string c\_sNullArray = "Ошибка! Массив не создан.\n";

private const string c\_sCleanArray = "Массив пуст, поэтому удален.\n";

private const string c\_sNullFunction = "Спасибо за работу!";

private const string c\_sGetMode =

"Введите способ получения чисел (1 - ввод, - 2 случайное): ";

private const string c\_sGetTask =

"МЕНЮ\n" +

"Одномерный массив\n" +

"\t1. вывести массив\n" +

"\t2. cоздать массив\n" +

"\t3. добавить N элементов, начиная с K элемента\n" +

"Двемерный массив\n" +

"\t4. вывести массив\n" +

"\t5. создать массив\n" +

"\t6. удалить чентные строки в массиве\n" +

"Рваный массив\n" +

"\t7. вывести массив\n" +

"\t8. создать массив\n" +

"\t9. добавть строку в конец массива\n" +

"0. Выход\n" +

"Выберете действие: ";

private static Random s\_rand = new Random();

public static void Run()

{

while (true)

{

try

{

GetTask()();

}

catch (NullArrayException)

{

Console.WriteLine(c\_sNullArray);

}

catch (CleanArrayException)

{

Console.WriteLine(c\_sCleanArray);

}

catch (NullFunctionException)

{

Console.Write(c\_sNullFunction);

break;

}

}

}

public static void ReadNum(out int number, char simbol, int i, int j)

{

number = 0;

for (bool flag = false; !flag;)

{

if (i == -1)

Console.Write(c\_sReadNumber, simbol);

else if (j == -1)

Console.Write(c\_sReadNumberA, simbol, i);

else

Console.Write(c\_sReadNumberA2, simbol, i, j);

string sNum = Console.ReadLine();

flag = int.TryParse(sNum, out number);

if (!flag)

IncorrectValue();

}

}

public static void RandNum(out int number, char simbol, int i, int j)

{

if (simbol == c\_cN || simbol == c\_cM)

number = s\_rand.Next(c\_iMinArray, c\_iMaxArray);

else if (simbol == c\_cK)

number = s\_rand.Next(c\_iMinK, c\_iMaxK);

else

number = s\_rand.Next(c\_iMinNumber, c\_iMaxNumber);

}

private static GetNumber GetMode()

{

string key = "";

do

{

Console.Write(c\_sGetMode);

key = Console.ReadLine();

} while (key != "1" && key != "2");

if (key == "1")

return ReadNum;

else

return RandNum;

}

public static void IncorrectValue()

{

Console.WriteLine(c\_sIncorrectValue);

}

public static void Output(int[] array)

{

Kernel.CheckArray(Kernel.array1D);

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

Console.Write(c\_sElem, array[i]);

Console.WriteLine("\n");

}

public static void Output(int[,] array)

{

Kernel.CheckArray(Kernel.array2D);

int n = array.GetUpperBound(0) + 1;

int k = array.Length / n;

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

{

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

Console.Write(c\_sElem, array[i, j]);

Console.WriteLine();

}

Console.WriteLine();

}

public static void Output(int[][] array)

{

Kernel.CheckArray(Kernel.arrayRagged);

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

{

for (int j = 0, k = array[i].Length; j < k; ++j)

Console.Write(c\_sElem, array[i][j]);

Console.WriteLine();

}

Console.WriteLine();

}

public static void Output1()

{

Output(Kernel.array1D);

}

public static void Output2()

{

Output(Kernel.array2D);

}

public static void Output3()

{

Output(Kernel.arrayRagged);

}

public static void Formation1()

{

Kernel.Formation1D(GetMode());

Output1();

}

public static void Formation2()

{

Kernel.Formation2D(GetMode());

Output2();

}

public static void Formation3()

{

Kernel.FormationRagged(GetMode());

Output3();

}

public static void Operation1()

{

Kernel.Addition(GetMode);

Output1();

}

public static void Operation2()

{

Kernel.DeleteEven();

Output2();

}

public static void Operation3()

{

Kernel.AdditionLine(GetMode);

Output3();

}

private static Task GetTask()

{

bool flag = true;

while (flag)

{

Console.Write(c\_sGetTask);

switch (Console.ReadLine())

{

case "1":

return Output1;

case "2":

return Formation1;

case "3":

return Operation1;

case "4":

return Output2;

case "5":

return Formation2;

case "6":

return Operation2;

case "7":

return Output3;

case "8":

return Formation3;

case "9":

return Operation3;

case "0":

flag = false;

break;

default:

continue;

}

}

throw new NullFunctionException();

}

}

}

**3.3 Program.cs**

using System;

// Вариант 15

namespace Lab5

{

class Program

{

public static void Main()

{

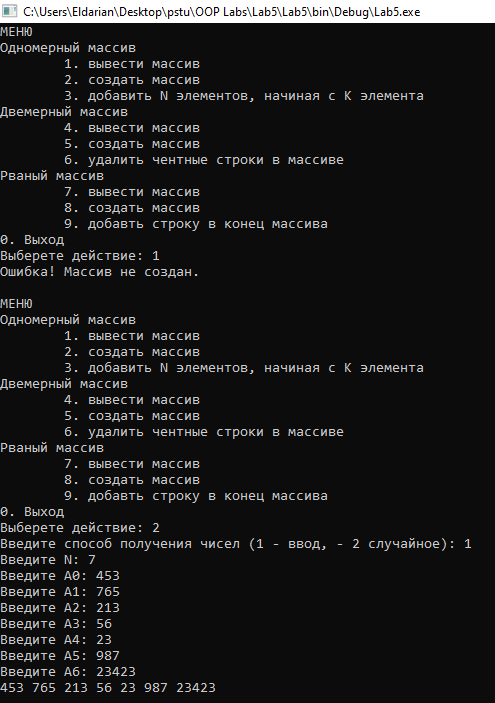
CLI.Run();

Console.ReadKey();

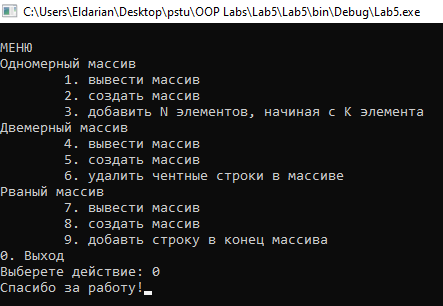
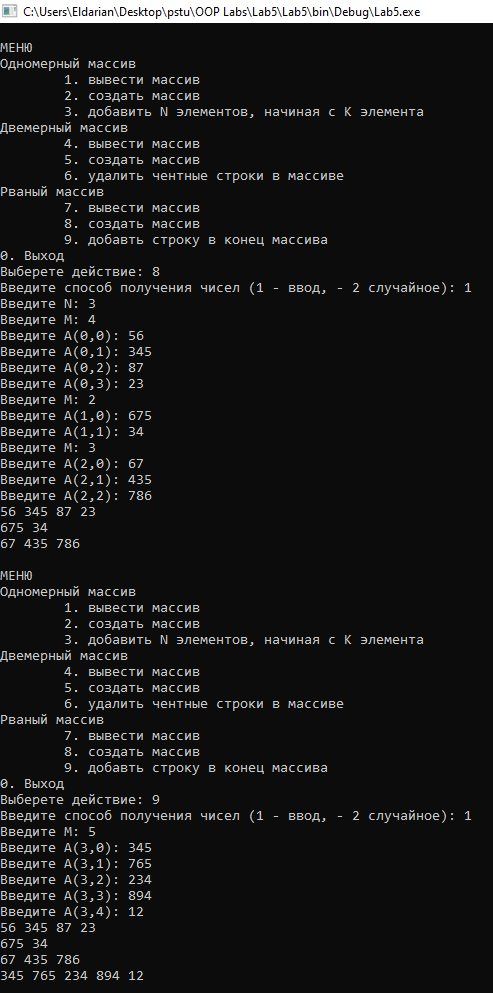
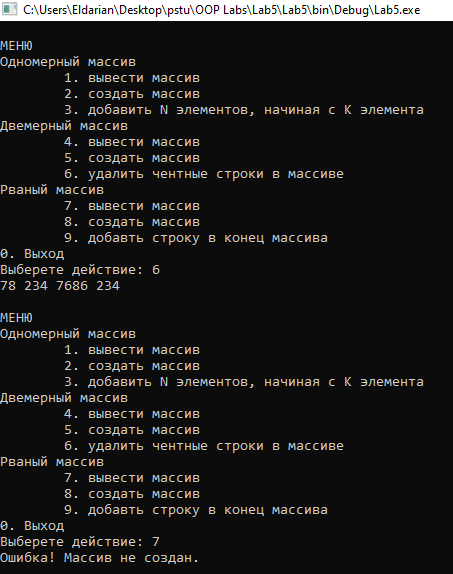
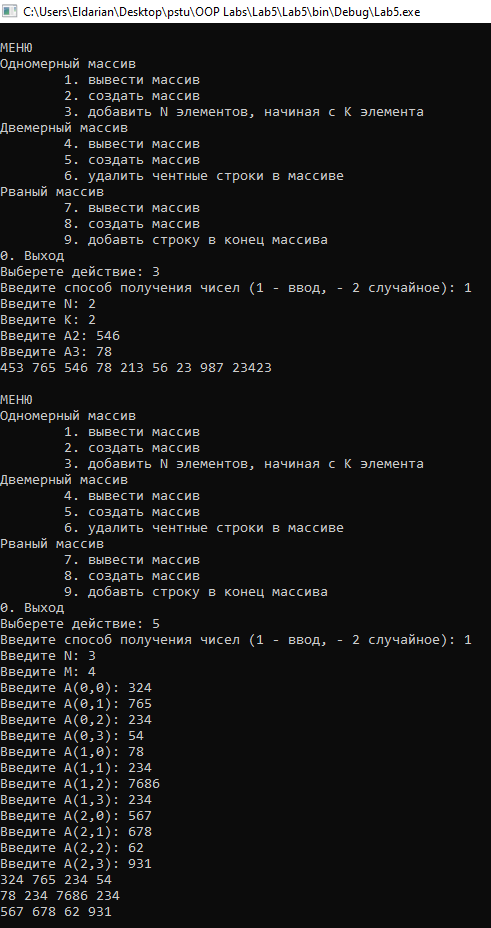
}

}

}

**4 СКРИНШОТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

При выполнении операций до создания массива, вызывается ошибка.

****