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

Лабораторная работа №4

По дисциплине

“Объектно-ориентированное программирование”

# Тема “Работа с одномерными массивами”

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 массив из n элементов с помощью датчика случайных чисел (n задается пользователем с клавиатуры).
2. Распечатать массив.
3. Выполнить удаление указанных элементов из массива.
4. Выполнить добавление указанных элементов в массив.
5. Выполнить перестановку элементов в массиве.
6. Выполнить поиск указанных в массиве элементов и подсчитать количество сравнений, необходимых для поиска нужного элемента.
7. Выполнить сортировку массива указанным методом.
8. Выполнить поиск указанных элементов в отсортированном массиве и подсчитать количество сравнений, необходимых для поиска нужного элемента.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Удаление | Добавление | Перестановка | Поиск | Сортировка |
| Все элементы с четными индексами | N элементов, начиная с номера К | Перевернуть массив | Элемент с заданным ключом (значением) | Простое включение |

**2 АЛГОРИТМ ПРОГРАММЫ**

**2.1 Удаление элементов с четными индексами**

int[] temp = array;

Int n = temp.Length / 2;

array = new int[n];

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

array[i] = temp[2\*I+1];

**2.2 Добавление n элементов с k элемента**

int[] temp = array;

Int oldSize = temp.Length

Int newSize = oldSize + n

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

array[i] = temp[i]

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

array[i] = GetNum();

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

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

array = new int[newSize];

**2.3 Переворот массива**

for(int i = 0, N = array.Length – 1 ; i < N/2; ++i)

int temp = array[i];

array[i] = array[N - i];

array[N – i] = temp;

**2.4 Поиск элемента по значению**

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

return I;

array[i] == Elem

return -1;

**2.5 Сортировка массива методом вставок**

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

temp = array[0];

array[j + 1] = temp;

int i, j, temp, n = array.Length;

for(j = i - 1; j >= 0 && array[j] > temp; j--)

array[j + 1] = array[j];

**3 ИСХОДНЫЙ КОД ПРОГРАММЫ**

**3.1 Kernel.cs**

using System;

namespace Lab4

{

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

delegate GetNumber ModeGetNumber();

delegate bool IsValidate(int x, int top);

class NullArrayException : Exception { }

class NullFunctionException : Exception { }

class FoundElemException : Exception

{

public int N { get; set; }

public FoundElemException(int n) { N = n; }

}

class Kernel

{

public static int[] array { 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)

{

do GetNum(out number, simbol);

while (!IsValid(number, top));

}

public static void CheckArray()

{

if (array == null)

throw new NullArrayException();

}

// Создание массива

public static void Formation(GetNumber GetNum)

{

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

array = new int[n];

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

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

}

// Удаление всех элементов с четными индексами

public static void DeleteElems()

{

CheckArray();

int[] temp = array;

int n = temp.Length / 2;

array = new int[n];

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

array[i] = temp[i \* 2 + 1];

}

// Добавление n элементов, начиная с k

public static void Addition(ModeGetNumber ModeGetNum)

{

CheckArray();

GetNumber GetNum = ModeGetNum();

int oldSize = array.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 = array;

array = new int[newSize];

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

array[i] = temp[i];

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

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

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

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

}

// Переворот массива

public static void Reverse()

{

CheckArray();

int N = array.Length - 1;

int n = N / 2;

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

{

int temp = array[i];

array[i] = array[N - i];

array[N - i] = temp;

}

}

// Поиск элемента по значению

public static void FindElem(ModeGetNumber ModeGetNum)

{

CheckArray();

GetNumber GetNum = ModeGetNum();

GetNum(out int k, CLI.c\_cK);

int n = 0;

while (array[n++] != k) ;

throw new FoundElemException(n);

}

// Сортировка вставками

public static void InsertSort()

{

CheckArray();

int i, j, temp, n = array.Length;

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

{

temp = array[i];

for (j = i - 1; j >= 0 && array[j] > temp; j--)

array[j + 1] = array[j];

array[j + 1] = temp;

}

}

}

}

**3.2 CLI.cs**

using System;

namespace Lab4

{

delegate void Task();

class CLI

{

public const char c\_cN = 'N';

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\_sNullArray = "Ошибка! Массив не создан.\n";

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

private const string c\_sNotFound = "Элемент не найден\n";

private const string c\_sFoundElem =

"Порядковый номер искомого элемента: {0}\n";

private const string c\_sGetMode =

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

private const string c\_sGetTask =

"МЕНЮ\n" +

"1. Вывести массив\n" +

"2. Создать массив\n" +

"3. Удалить четные элементы в массиве\n" +

"4. Добавить n элементов с k элемента\n" +

"5. Перевернуть массив\n" +

"6. Поиск элемента\n" +

"7. Сортировка 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 (FoundElemException e)

{

Console.WriteLine(CLI.c\_sFoundElem, e.N);

}

catch (IndexOutOfRangeException)

{

Console.WriteLine(CLI.c\_sNotFound);

}

catch (NullFunctionException)

{

Console.Write(c\_sNullFunction);

break;

}

}

}

// Запись числа со стандартного потока ввода в выходной параметр

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

{

number = 0;

for (bool flag = false; !flag;)

{

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

Console.Write(c\_sReadNumber, simbol);

else

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

string sNum = Console.ReadLine();

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

}

}

// Запись случайного числа в выходной параметр

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

{

if (simbol == c\_cN)

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;

}

// Вывод массива

private static void Output(int[] array)

{

Kernel.CheckArray();

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

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

Console.WriteLine("\n");

}

private static void Output()

{

Output(Kernel.array);

}

private static void Formation()

{

Kernel.Formation(GetMode());

Output(Kernel.array);

}

private static void DeleteElems()

{

Kernel.DeleteElems();

Output(Kernel.array);

}

private static void Addition()

{

Kernel.Addition(GetMode);

Output(Kernel.array);

}

private static void Reverse()

{

Kernel.Reverse();

Output(Kernel.array);

}

private static void FindElem()

{

Kernel.FindElem(GetMode);

}

private static void InsertSort()

{

Kernel.InsertSort();

Output(Kernel.array);

}

private static Task GetTask()

{

bool flag = true;

while(flag)

{

Console.Write(c\_sGetTask);

switch (Console.ReadLine())

{

case "1":

return Output;

case "2":

return Formation;

case "3":

return DeleteElems;

case "4":

return Addition;

case "5":

return Reverse;

case "6":

return FindElem;

case "7":

return InsertSort;

case "0":

flag = false;

break;

default:

continue;

}

}

throw new NullFunctionException();

}

}

}

**3.3 Program.cs**

using System;

namespace Lab4

{

class Program

{

public static void Main(string[] args)

{

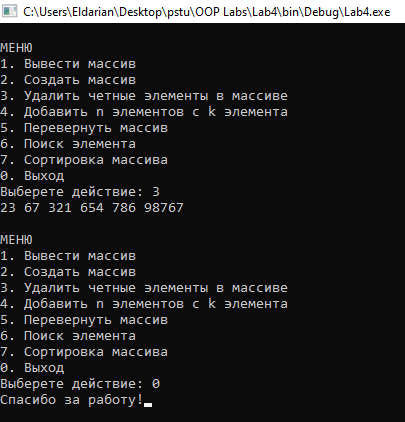
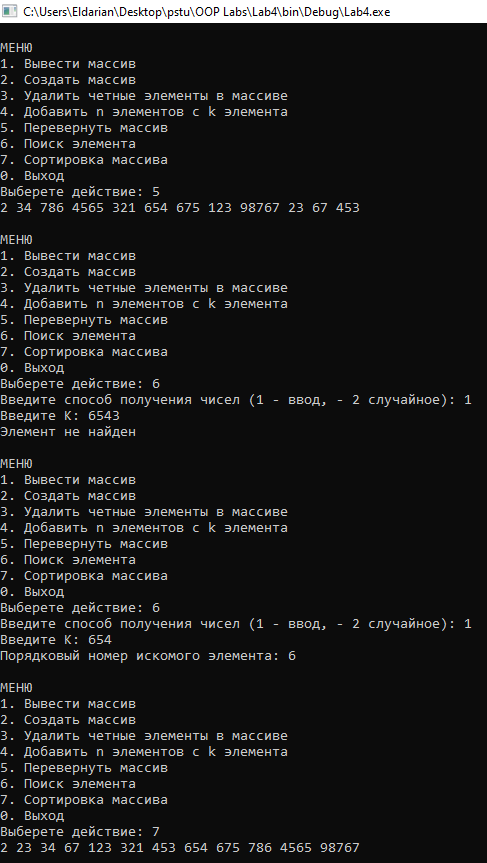
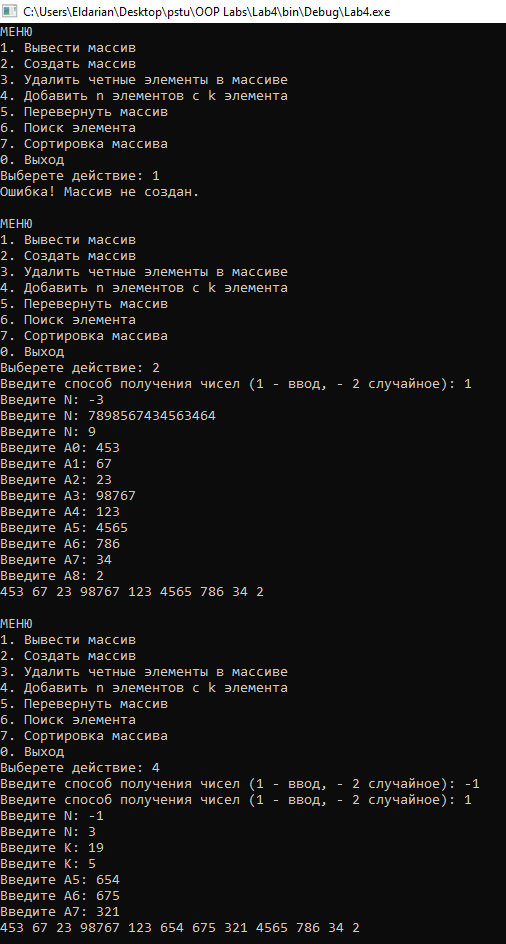
CLI.Run();

Console.ReadKey();

}

}

}

**4 СКРИНШОТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ**