|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
|  |
| Гуцол Степан Дмитриевич  **ЛАБОРАТОРНАЯ РАБОТА №4**  по направлению подготовки *38.03.05 Бизнес-информатика*  образовательная программа «Разработка информационных систем для бизнеса»   |  |  | | --- | --- | |  | Руководитель  Преподаватель кафедры ИТБ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Найданов И.В. |   Пермь, 2022 |

**Оглавление**

[**Задача №1** 4](#_Toc118647143)

[**1. Постановка задачи** 4](#_Toc118647144)

[**2. Анализ** 4](#_Toc118647145)

[**3. Алгоритм (блок-схема)** 5](#_Toc118647146)

[**4. Программа (листинг)** 6](#_Toc118647147)

[**5. Тестирование** 7](#_Toc118647148)

[**Задача №2** 8](#_Toc118647149)

[**1. Постановка задачи** 8](#_Toc118647150)

[**2. Анализ** 8](#_Toc118647151)

[**3. Алгоритм (блок-схема)** 9](#_Toc118647152)

[**4. Программа (листинг)** 10](#_Toc118647153)

[**5. Тестирование** 11](#_Toc118647154)

[**Задача №3** 12](#_Toc118647155)

[**1. Постановка задачи** 12](#_Toc118647156)

[**2. Анализ** 12](#_Toc118647157)

[**3. Алгоритм (блок-схема)** 13](#_Toc118647158)

[**4. Программа (листинг)** 14](#_Toc118647159)

[**5. Тестирование** 15](#_Toc118647160)

[**Задача №4** 16](#_Toc118647161)

[**1. Постановка задачи** 16](#_Toc118647162)

[**2. Анализ** 16](#_Toc118647163)

[**3. Алгоритм (блок-схема)** 17](#_Toc118647164)

[**4. Программа (листинг)** 18](#_Toc118647165)

[**5. Тестирование** 19](#_Toc118647166)

[**Задача №5** 20](#_Toc118647167)

[**1. Постановка задачи** 20](#_Toc118647168)

[**2. Анализ** 20](#_Toc118647169)

[**3. Алгоритм (блок-схема)** 21](#_Toc118647170)

[**4. Программа (листинг)** 22](#_Toc118647171)

[**5. Тестирование** 23](#_Toc118647172)

[**Задача №6** 24](#_Toc118647173)

[**1. Постановка задачи** 24](#_Toc118647174)

[**2. Анализ** 24](#_Toc118647175)

[**3. Алгоритм (блок-схема)** 25](#_Toc118647176)

[**4. Программа (листинг)** 26](#_Toc118647177)

[**5. Тестирование** 27](#_Toc118647178)

[**Задача №7** 28](#_Toc118647179)

[**1. Постановка задачи** 28](#_Toc118647180)

[**2. Анализ** 28](#_Toc118647181)

[**3. Алгоритм (блок-схема)** 29](#_Toc118647182)

[**4. Программа (листинг)** 30](#_Toc118647183)

[**5. Тестирование** 31](#_Toc118647184)

**Задача №1**

**1. Постановка задачи**

Сформировать массив из n элементов с помощью датчика случайных чисел.

**2. Анализ**

Исходные данные:

n – целое число большее нуля.

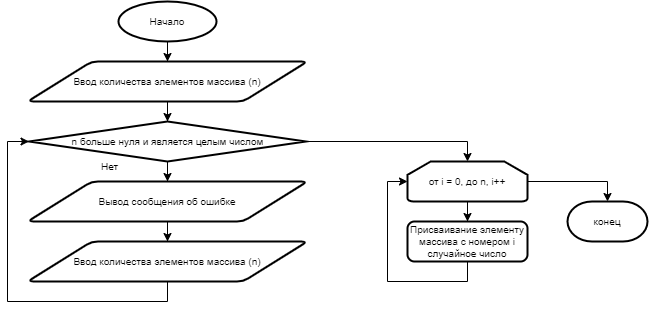
Выходные данные:

array – массив целых чисел от 0 до 100 с n количеством элементов.

**Таблица 1 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Сформировать массив из n элементов с помощью датчика случайных чисел | n | n – целое число > 0 | массив целых чисел от 0 до 100 с длиной n |
| * n – строки (не числа) * n –отрицательные числа * n – нуль | сообщение об ошибке ввода |

**3. Алгоритм (блок-схема)**



***Рисунок 1 – Блок-схема инициализации массива с помощью ДСЧ***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными способами создания массива, в этом случае пользователь выбирает создание с помощью ДСЧ. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 2 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | n = ‘0’ | Сообщение об ошибке | Сообщение об ошибке |
| **2** | n = ‘5’ | Массив из 5 случайных чисел | array = ‘47 21 58 32 21’ |
| **3** | n = ‘-3’ | Сообщение об ошибке | Сообщение об ошибке |
| **4** | n = ‘три’ | Сообщение об ошибке | Сообщение об ошибке |

**Задача №2**

**1. Постановка задачи**

Сформировать массив из n элементов (n задается пользователем с клавиатуры).

**2. Анализ**

Исходные данные:

* n – целое число большее нуля,
* element – целое число

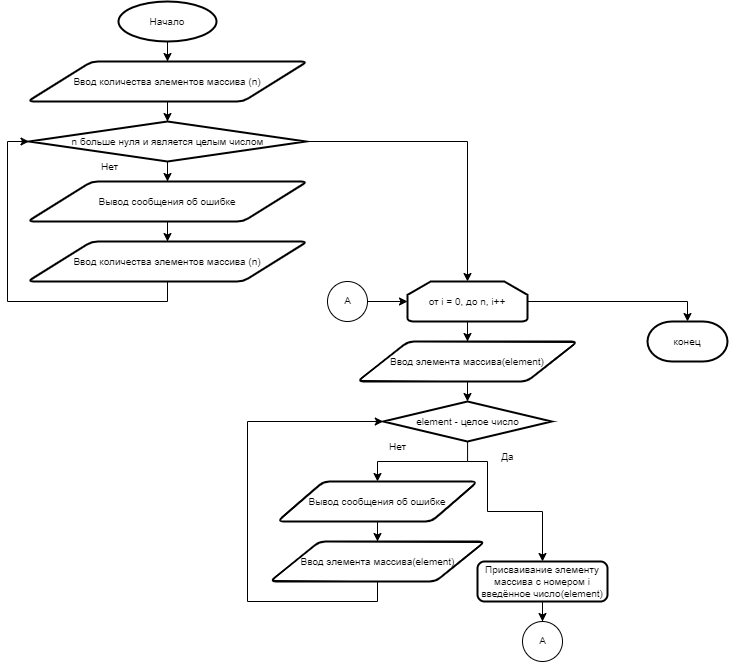
Выходные данные:

array – массив, состоящий из элементов(element), с n количеством элементов.

**Таблица 3 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Сформировать массив из n элементов с помощью ручного ввода | n | n – целое число > 0 | ввод элементов массива |
| * n – строки (не числа) * n – отрицательные числа * n – нуль | сообщение об ошибке ввода |
| element | element – целое число | массив целых чисел с длиной n |
| * element – строки (не числа) * element – вещественные числа | сообщение об ошибке ввода |

**3. Алгоритм (блок-схема)**



***Рисунок 2 – Блок-схема инициализации массива с помощью ручного ввода***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными способами создания массива, в этом случае пользователь выбирает создание с помощью ручного ввода. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 4 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | n = ‘0’  element= ’’ | Сообщение об ошибке | Сообщение об ошибке |
| **2** | n = ‘5’  element= ’47 21 58 32 21’ | Массив из 5 случайных чисел | array = ‘47 21 58 32 21’ |
| **3** | n = ‘-3’  element = ’’ | Сообщение об ошибке | Сообщение об ошибке |
| **4** | n = ‘три’  element = ‘’ | Сообщение об ошибке | Сообщение об ошибке |
| **5** | n = ‘3’  element = ‘2 -5 четыре’ | Сообщение об ошибке | Сообщение об ошибке |

**Задача №3**

**1. Постановка задачи**

Распечатать массив.

**2. Анализ**

Исходные данные:

array – массив, состоящий из целых чисел.

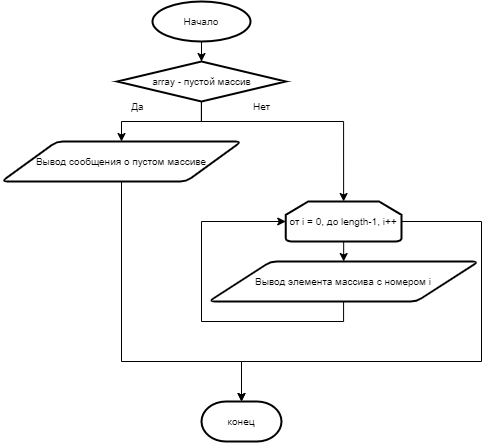
Выходные данные:

tmp – временный массив, с которым будут производится последующие действия.

**Таблица 5 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Распечатать массив | array | array – массив с количеством элементов больше нуля | вывод элементов массива |
| array – пустой массив | сообщение о пустом массиве |

**3. Алгоритм (блок-схема)**



***Рисунок 3 – Блок-схема печати массива***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает печать массива. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 6 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | array = ‘’ | Сообщение о пустом массиве | Сообщение о пустом массиве |
| **2** | array = ‘2 5 7’ | Массив из 3 случайных чисел | tmp = ‘2 5 7’ |

**Задача №4**

**1. Постановка задачи**

Выполнить удаление чётных элементов из массива.

**2. Анализ**

Исходные данные:

tmp – массив, состоящий из целых чисел.

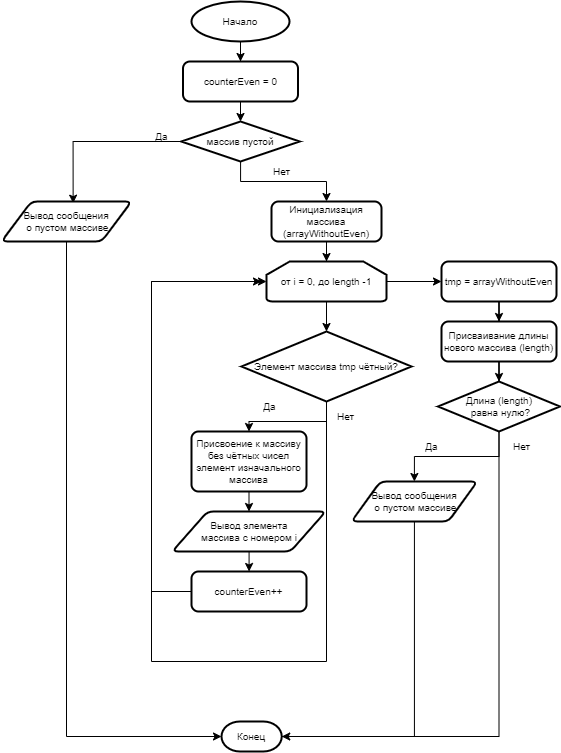
Выходные данные:

arrayWithoutEven – временный массив, с которым будут производится последующие действия.

**Таблица 7 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Выполнить удаление чётных элементов из массива | tmp | tmp – массив с количеством элементов больше нуля | вывод элементов массива без чётных элементов |
| tmp – пустой массив | сообщение о пустом массиве |

**3. Алгоритм (блок-схема)**



***Рисунок 4 – Блок-схема удаления элементов массива***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает удаление чётных элементов массива. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 8 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | tmp = ‘’ | Сообщение о пустом массиве | Сообщение о пустом массиве |
| **2** | tmp = ‘2 5 7’ | arrayWithoutEven = ‘5 7’ | arrayWithoutEven = ‘ 5 7’ |
| **3** | tmp = ‘2’ | Сообщение о пустом массиве | Сообщение о пустом массиве |

**Задача №5**

**1. Постановка задачи**

Выполнить добавление N элементов, начиная с номера K, в массив.

**2. Анализ**

Исходные данные:

* tmp – массив, состоящий из целых чисел или пустой,
* n2 – целое число, большее нуля,
* k – целое число, большее нуля.

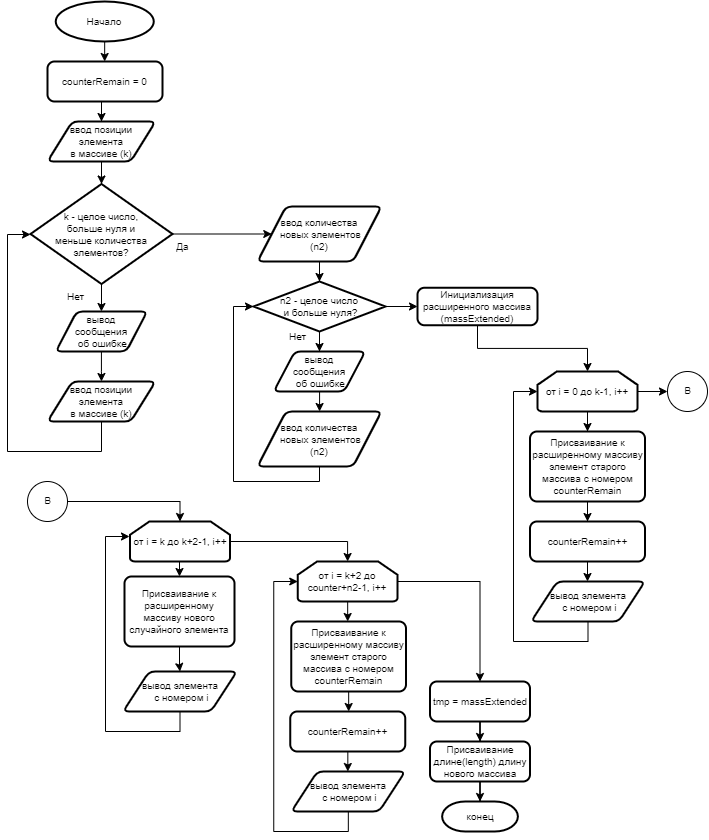
Выходные данные:

massExtended – временный массив, с которым будут производится последующие действия.

**Таблица 9 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Выполнить добавление N элементов, начиная с номера K, в массив | tmp | tmp – массив с любым количеством элементов | вывод нового массива с добавленными элементами |
| n2 | n2 – целое число, большее нуля | новый увеличенный массив |
| * n2 – строки (не числа) * n2 – отрицательные числа * n2 – нуль | сообщение об ошибке |
| k | k – целое число, большее нуля | ввод количества новых элементов |
| * k – строки (не числа) * k – отрицательные числа * k – нуль | сообщение об ошибке |

**3. Алгоритм (блок-схема)**



***Рисунок 5 – Блок-схема добавления новых элементов***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает добавление новых элементов в массив. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 10 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | tmp = ‘2 3’  k = ‘1’  n2 = ‘2’ | Новый массив, состоящий из 4-ёх элементов | massExtended = ‘2 31 76 3’ |
| **2** | tmp = ‘’  k= ’1’  n2=’3’ | Новый массив, состоящий из 3-ёх элементов | massExtended = ’52 47 76’ |
| **3** | tmp = ‘2 5’  k= ’2’  n2 = ‘два’ | Сообщение об ошибке | Сообщение об ошибке |
| **4** | tmp = ‘2 5’  k=’два’  n2 = ‘2’ | Сообщение об ошибке | Сообщение об ошибке |
| **5** | tmp = ‘2 5’  k= ’3’  n2 = ‘2’ | Сообщение об ошибке (так как «к» больше, чем позиций в массиве) | Сообщение об ошибке |

**Задача №6**

**1. Постановка задачи**

Сдвинуть циклически на M элементов вправо.

**2. Анализ**

Исходные данные:

* tmp – массив, состоящий из целых чисел или пустой,
* m – целое положительное число.

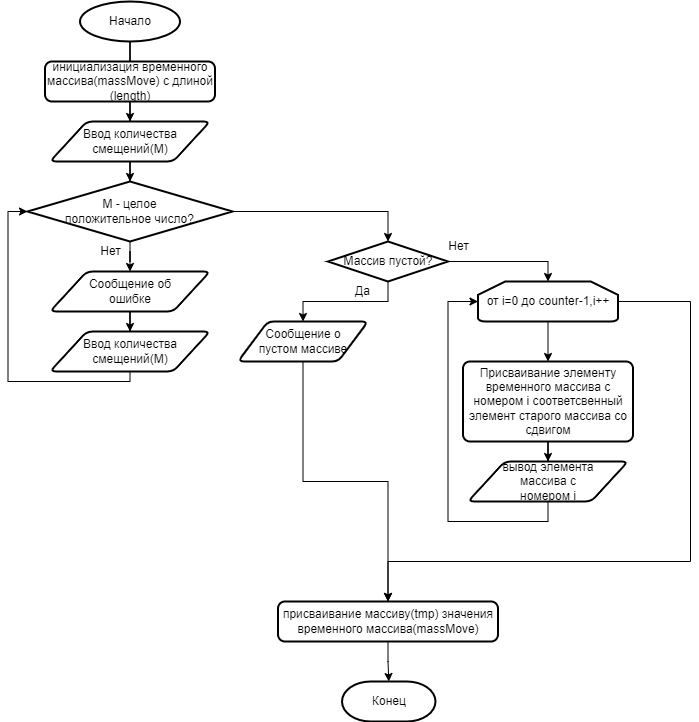
Выходные данные:

massMove – временный массив, с которым будут производится последующие действия.

**Таблица 11 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Сдвинуть циклически на M элементов вправо. | tmp | tmp – массив с положительном количеством элементов | вывод нового массива со сдвинутыми элементами |
| tmp – пустой массив | сообщение о пустом массиве |
| m | m – целое число, большее нуля | новый массив со сдвинутыми элементами |
| * m – строки (не числа) * m – отрицательные числа * m – нуль | сообщение об ошибке |

**3. Алгоритм (блок-схема)**



***Рисунок 6 – Блок-схема сдвига элементов массива***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает циклический сдвиг элементов массива вправо. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 12 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | tmp = ‘’  m = ‘2’ | Сообщение о пустом массиве | Сообщение о пустом массиве |
| **2** | tmp = ‘1 2 3’  m = ‘2’ | tmp = ‘2 3 1’ | tmp = ‘2 3 1’ |
| **3** | tmp = ‘1 2 3’  m = ‘0’ | Сообщение об ошибке | Сообщение об ошибке |
| **4** | tmp = ‘1 2 3’  m = ‘-2’ | Сообщение об ошибке | Сообщение об ошибке |
| **5** | tmp = ‘1 2 3’  m = ‘1.5’ | Сообщение об ошибке | Сообщение об ошибке |
| **6** | tmp = ‘1 2 3’  m = ‘4’ | tmp = ‘3 1 2’ | tmp = ‘3 1 2’ |

**Задача №7**

**1. Постановка задачи**

Выполнить поиск элемента с заданным ключом (значением) и подсчитать количество сравнений, необходимых для поиска нужного элемента.

**2. Анализ**

Исходные данные:

* tmp – массив, состоящий из целых чисел или пустой,
* needed – целое число.

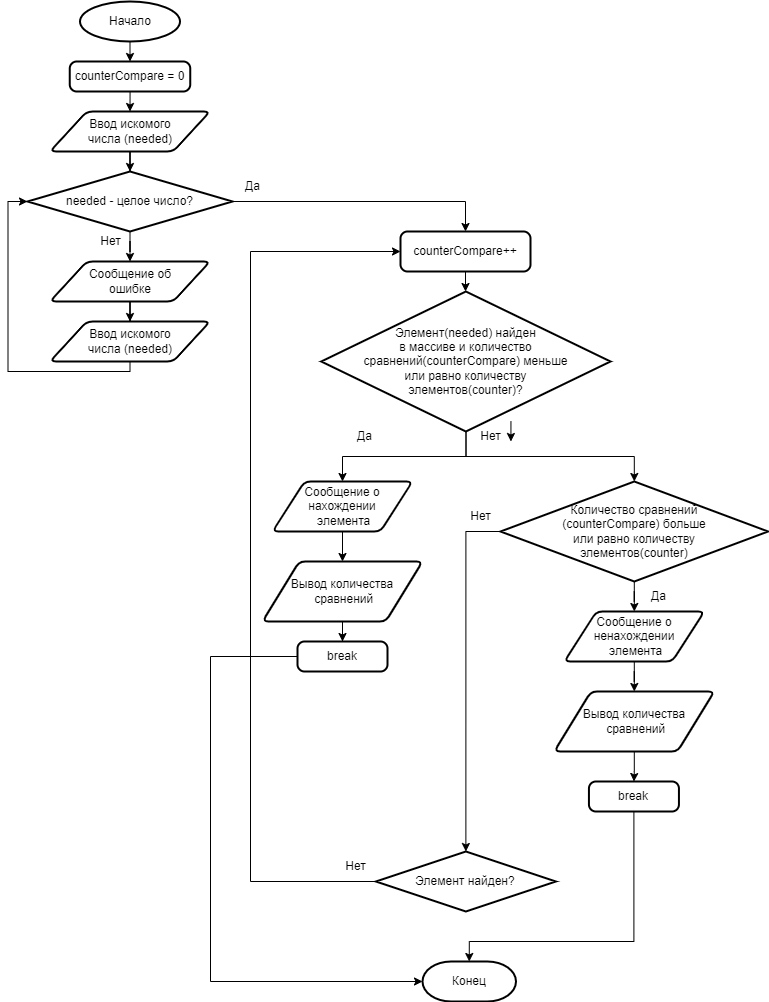
Выходные данные:

counterCompare– целое число, большее нуля.

**Таблица 13 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Выполнить поиск элемента с заданным ключом (значением) и подсчитать количество сравнений, необходимых для поиска нужного элемента. | tmp | tmp – массив с положительном количеством элементов | Количество сравнений |
| tmp – пустой массив | сообщение о ненахождении элемента |
| needed | needed – целое число | Количество сравнений |
| * needed – строки (не числа) | сообщение об ошибке |

**3. Алгоритм (блок-схема)**



***Рисунок 7 – Блок-схема поиска элемента***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает нахождение элемента с заданным ключом. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 14 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | tmp = ‘’  needed = ‘2’ | Сообщение о пустом массиве | Сообщение о пустом массиве |
| **2** | tmp = ‘1 2 3’  needed = ‘2’ | Сообщение о нахождении элемента  counterCompare = 2 | Сообщение о нахождении элемента  counterCompare = 2 |
| **3** | tmp = ‘1 2 3’  needed = ‘4’ | Сообщение о ненахождении элемента  counterCompare = 3 | Сообщение о ненахождении элемента  counterCompare = 3 |
| **4** | tmp = ‘1 2 3’  needed = ‘три’ | Сообщение об ошибке | Сообщение об ошибке |

**Задача №8**

**1. Постановка задачи**

Выполнить сортировку массива простым выбором.

**2. Анализ**

Исходные данные:

tmp – массив, состоящий из целых чисел или пустой,

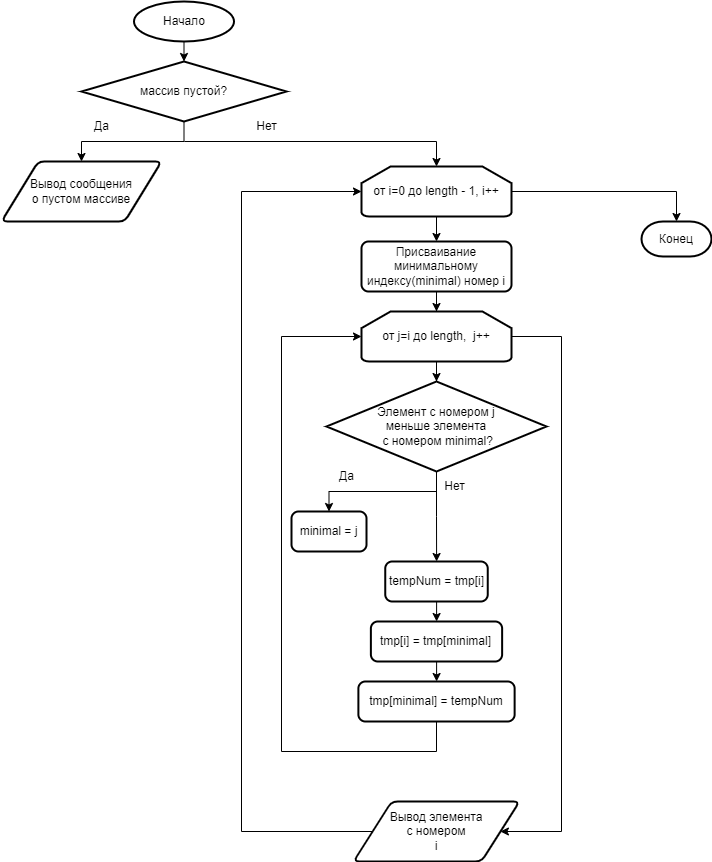
Выходные данные:

tmp– отсортированный массив.

**Таблица 15 – Анализ классов исходных данных**

| **Задача** | **Исходные данные** | **Класс входных данных** | **Класс выходных данных** |
| --- | --- | --- | --- |
| Выполнить сортировку массива простым выбором. | tmp | tmp – массив с положительном количеством элементов | отсортированный массив |
| tmp – пустой массив | сообщение о пустом массиве |

**3. Алгоритм (блок-схема)**



***Рисунок 8 – Блок-схема сортировки массива простым выбором***

**4. Программа (листинг)**

На выбор пользователю предоставляется возможность воспользоваться разными функциями меню, в этом случае пользователь выбирает сортировку массива простым выбором. Весь код программы представлен в приложении. См. Приложение А.

**5. Тестирование**

**Таблица 16 – Тестирование методом чёрного ящика**

| **№** | **Исходные данные** | **Ожидаемый результат** | **Полученный результат** |
| --- | --- | --- | --- |
| **1** | tmp = ‘’ | Сообщение о пустом массиве | Сообщение о пустом массиве |
| **2** | tmp = ‘49 53 32 61 59’ | tmp = ‘32 49 53 59 61’ | tmp = ‘32 49 53 59 61’ |
| **3** | tmp = ‘2 6 4 4’ | tmp = ‘2 4 4 6’ | tmp = ‘2 4 4 6’ |

# ПРИЛОЖЕНИЕ А **Программа**

//23 Вариант

using System.Diagnostics.Metrics;

using System.Net.Http.Headers;

using System.Security.Cryptography.X509Certificates;

int n, // Количество элементов массива

k, // Позиция для добавления элементов

element, // Элемент, который добавляется рандомом

n2, // Число n для второго задания

counterEven = 0, // Счётчик чётных

counterRemain = 0, // Счётчик оставшихся

counterCompare = 0, // Счётчик сравнений

counter = 0, // Счётчик элементов

enter, // Ввод символов

menu, // Ввод меню

length, // Длина массива

returnMenu, // Считывание возврата в меню

minimal = 0, // Минимальное значение индекса для сортировки

tempNum, //Временная переменная для сортировки

needed; // Число, которое надо будет найти в массиве

bool okN = false, // Проверка числа N

okN2 = false, // Проверка числа N в удалении

okK = false, // Проверка числа К

okM = false, // Проверка числа М

okNeeded = false, // Проверка искомого числа

okElement = false, // Проверка жлемента

exit = false, // Выход из всей программы

okEnter, // Проверка ввода

okMenu , // Проверка ввода

exitMenu = false, // Выход из меню

found = false; //Нахождение элемента

string strN,

strElement;

Random rand = new Random(); // Переменная рандома

do

{

do

{

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("Каким образом сформировать массив?"); // Меню ввода типа массива

Console.WriteLine("");

Console.WriteLine("[1] - Датчиком случайных чисел");

Console.WriteLine("[2] - Ручной ввод");

Console.WriteLine();

string strEnter = Console.ReadLine();

okEnter = int.TryParse(strEnter, out enter);

Console.Clear();

if (okEnter)

{

switch (enter)

{

case 1: // Массив из ДСЧ

Console.WriteLine("Введите число n - количество элементов массива, переменная n должна быть больше нуля и являться целым числом:");

strN = Console.ReadLine();

Console.WriteLine();

okN = (int.TryParse(strN, out n)) && (n > 0);

while (!okN) // Проверка N

{

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

Console.WriteLine("Введите число n - количество элементов массива, переменная n должна быть больше нуля и являться целым числом:");

strN = Console.ReadLine();

Console.WriteLine();

okN = (int.TryParse(strN, out n)) && (n > 0);

}

int[] array = new int[n];

for (int i = 0; i <= n - 1; i++) // Рандомные числа создаются

{

array[i] = (int)rand.Next(1,100);

}

Console.Clear();

int[] tmp = new int[100]; // Ввод массива

tmp = array;

length = tmp.Length;

do

{

bool returnOk = false;

counter = 0;

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Меню функций программы:"); // Контекстное меню

Console.WriteLine("");

Console.WriteLine("[1] - Распечатать массив");

Console.WriteLine("[2] - Удалить все чётные элементы массива");

Console.WriteLine("[3] - Добавить N элементов с позиции K");

Console.WriteLine("[4] - Циклически сдвинуть массив вправо");

Console.WriteLine("[5] - Поиск элемента с заданным ключом");

Console.WriteLine("[6] - Сортировка массива с помощью простого выбора");

Console.WriteLine("");

Console.WriteLine("[0] - Завершить работу программы");

Console.WriteLine("");

string strMenu = Console.ReadLine();

okMenu = int.TryParse(strMenu, out menu);

for (int i = 0; i < tmp.Length; i++) // Нахождение длины

{

if (tmp[i] != 0)

{

counter++;

}

}

if (counter == 0)

{

counter = 1;

}

Console.ForegroundColor = ConsoleColor.Magenta;

if (okMenu)

{

switch (menu)

{

case 1: // Первый вариант выбора меню / Распечатать массив

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0)||(counter ==1 & tmp[0] == 0))

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

for (int i = 0; i <= length - 1; i++) // Печатать массив

{

if (tmp[i] != 0)

{

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

}

}

Console.WriteLine("");

Console.WriteLine("");

}

Console.WriteLine("[1] - Вернуться в меню"); // Возврат в меню

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu) // Проверка возврата в меню

{

case 1: //Возврат в меню

returnOk = true;

Console.Clear();

break;

case 2: // Выход из программы

returnOk = true;

exitMenu = true;

break;

default: // Игнорирование

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 0: // Нулевой пункт меню - выход

exitMenu = true;

break;

case 2: // Пункт меню / Удаление чётных элементов

counterEven =0;

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0)) // Пустой массив

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

int[] arrayWithoutEven = new int[tmp.Length];

for (int i = 0; i <= tmp.Length - 1; i++) // Удаление элементов

{

if (tmp[i]% 2 != 0) // Проверка на чётность

{

arrayWithoutEven[counterEven] = tmp[i];

Console.Write(arrayWithoutEven[counterEven] +" ");

counterEven++;

}

}

tmp = arrayWithoutEven; // Временный массив = массив без чётных элементов

length = arrayWithoutEven.Length;

if (length == 1 & tmp[0] == 0)

{

Console.Write("пуст");

}

Console.WriteLine("");

Console.WriteLine("");

}

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 3: // Третий пункт меню / Добавление новых элементов

counterRemain = 0;

Console.Clear();

Console.WriteLine("Введите число k - индекс,с которого будут добавляять новые элементы, 0<k<=n, являться целым числом");

string strK = Console.ReadLine();

Console.WriteLine();

okK = (int.TryParse(strK, out k)) & (k > 0) & (k <= counter);

while (!okK) // Проверка ввода K

{

Console.WriteLine("Введите число k - индекс,с которого будут добавляять новые элементы, 0<k<=n, являться целым числом");

strK = Console.ReadLine();

Console.WriteLine();

okK = (int.TryParse(strK, out k)) & (k > 0) & (k <= counter);

}

Console.Clear();

Console.WriteLine("Введите число n - количество новых элементов массива, n должна быть больше нуля и являться целым числом");

string strN2 = Console.ReadLine();

Console.WriteLine();

okN2 = (int.TryParse(strN2, out n2)) && (n2 > 0);

while (!okN2) // Проверка ввода N

{

Console.WriteLine("Введите число n - количество новых элементов массива, n должна быть больше нуля и являться целым числом");

strN2 = Console.ReadLine();

Console.WriteLine();

okN2 = (int.TryParse(strN2, out n2)) && (n2 > 0);

}

Console.Clear();

Console.Write("Ваш массив:");

int[] massExtended = new int[counter + n2]; // Расширенный массив

for (int i = 0; i <= k - 1; i++) // Добавление элементов до позиции К, если они есть

{

massExtended[i] = tmp[counterRemain];

counterRemain++;

if (massExtended[i] != 0)

{

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

}

}

for (int i = k; i <= k + n2-1; i++) // Добавление новых элементов

{

massExtended[i] = (int)rand.Next(1,100);

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

}

for (int i = k + n2; i <= counter + n2 - 1; i++) // Добавление оставшихся элементов, если они остались

{

massExtended[i] = tmp[counterRemain];

counterRemain++;

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

}

tmp = massExtended;

length = massExtended.Length;

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 4: //Четвёртый пункт, циклическое смещение вправо

Console.Clear();

int[] massMove = new int[length]; // Новый массив, который будет свдинут

int m;

Console.WriteLine("Введите число M - количество позиций для смещения элементов, M должна являться целым положительным числом");

string strM = Console.ReadLine();

Console.WriteLine();

okM = (int.TryParse(strM, out m)) && (m > 0);

while (!okM) // Проверка ввода М

{

Console.WriteLine("Введите число M - количество позиций для смещения элементов, M должна являться целым положительным числом");

strM = Console.ReadLine();

Console.WriteLine();

okM = (int.TryParse(strM, out m)) && (m > 0);

}

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0))

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

for (int i = 0; i <= counter - 1; i++) // Смещение вправо

{

if (tmp[(((i + counter) - (m % counter)) % counter)] != 0)

{

massMove[i] = tmp[(((i + counter) - (m % counter)) % counter)];

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

}

}

}

tmp = massMove; // Присваивание к временному массиву

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 5: // Пятый пункт / Поиск элемента

Console.Clear();

counterCompare = 0;

okNeeded = false;

Console.WriteLine("Введите элемент, который надо будет найти в массиве, элемент должен являться целым числом:");

string strNeeded = Console.ReadLine();

Console.WriteLine();

okNeeded = (int.TryParse(strNeeded, out needed));

while (!okNeeded) // Проверка ввода искомого числа

{

Console.WriteLine("Введите элемент, который надо будет найти в массиве, элемент должен являться целым числом:");

strNeeded = Console.ReadLine();

Console.WriteLine();

okNeeded = (int.TryParse(strNeeded, out needed));

}

Console.Clear();

do // Поиск элемента

{

found = (tmp[counterCompare] == needed) && (counterCompare <= counter);

counterCompare = ++counterCompare;

if (found)

{

Console.WriteLine($"Найденный элемент находится на {counterCompare} позиции.");

Console.WriteLine($"Количество сравнений = {counterCompare}.");

break;

}

if (counterCompare >= counter) //Если прошёлся по массиву,а числа не нашёл

{

Console.WriteLine($"Элемент не найден в данном массиве!");

Console.WriteLine($"Количество сравнений = {counterCompare}.");

break;

}

} while (!found);

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 6: // Шестой пункт / Сортировка массива

Console.Clear();

Console.Write("Ваш массив: ");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0)) // Пустой массив

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

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

{

minimal = i;

for (int j = i; j < length; j++)

{

if (tmp[j] < tmp[minimal])

{

minimal = j;

}

tempNum = tmp[i];

tmp[i] = tmp[minimal];

tmp[minimal] = tempNum;

}

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

}

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

}

}

else

{

Console.Clear();

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

}

} while (exitMenu == false);

exit = true;

break;

case 2:

Console.WriteLine("Введите число n - количество элементов массива, переменная n должна быть больше нуля и являться целым числом:");

strN = Console.ReadLine();

Console.WriteLine();

okN = (int.TryParse(strN, out n)) && (n > 0);

while (!okN)

{

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

Console.WriteLine("Введите число n - количество элементов массива, переменная n должна быть больше нуля и являться целым числом:");

strN = Console.ReadLine();

Console.WriteLine();

okN = (int.TryParse(strN, out n)) && (n > 0);

}

array = new int[n];

for (int i = 0; i <= n - 1; i++) // Ручной ввод элементов массива

{

Console.WriteLine("Введите элемент массива, элемент должен являться целым числом:");

strElement = Console.ReadLine();

Console.WriteLine();

okElement = (int.TryParse(strElement, out element));

while (!okElement)

{

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

Console.WriteLine("Введите элемент массива, элемент должен являться целым числом:");

strElement = Console.ReadLine();

Console.WriteLine();

okElement = (int.TryParse(strElement, out element));

}

array[i] = element;

}

Console.Clear();

tmp = new int[100]; // Ввод массива

tmp = array;

length = tmp.Length;

do

{

bool returnOk = false;

counter = 0;

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Меню функций программы:"); // Контекстное меню

Console.WriteLine("");

Console.WriteLine("[1] - Распечатать массив");

Console.WriteLine("[2] - Удалить все чётные элементы массива");

Console.WriteLine("[3] - Добавить N элементов с позиции K");

Console.WriteLine("[4] - Циклически сдвинуть массив вправо");

Console.WriteLine("[5] - Поиск элемента с заданным ключом");

Console.WriteLine("[6] - Сортировка массива с помощью простого выбора");

Console.WriteLine("");

Console.WriteLine("[0] - Завершить работу программы");

Console.WriteLine("");

string strMenu = Console.ReadLine();

okMenu = int.TryParse(strMenu, out menu);

for (int i = 0; i < tmp.Length; i++) // Нахождение длины

{

if (tmp[i] != 0)

{

counter++;

}

}

if (counter == 0)

{

counter = 1;

}

Console.ForegroundColor = ConsoleColor.Magenta;

if (okMenu)

{

switch (menu)

{

case 1: // Первый вариант выбора меню / Распечатать массив

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0))

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

for (int i = 0; i <= length - 1; i++) // Печатать массив

{

if (tmp[i] != 0)

{

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

}

}

Console.WriteLine("");

Console.WriteLine("");

}

Console.WriteLine("[1] - Вернуться в меню"); // Возврат в меню

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu) // Проверка возврата в меню

{

case 1: //Возврат в меню

returnOk = true;

Console.Clear();

break;

case 2: // Выход из программы

returnOk = true;

exitMenu = true;

break;

default: // Игнорирование

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 0: // Нулевой пункт меню - выход

exitMenu = true;

break;

case 2: // Пункт меню / Удаление чётных элементов

counterEven = 0;

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0)) // Пустой массив

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

int[] arrayWithoutEven = new int[tmp.Length];

for (int i = 0; i <= tmp.Length - 1; i++) // Удаление элементов

{

if (tmp[i] % 2 != 0) // Проверка на чётность

{

arrayWithoutEven[counterEven] = tmp[i];

Console.Write(arrayWithoutEven[counterEven] + " ");

counterEven++;

}

}

tmp = arrayWithoutEven; // Временный массив = массив без чётных элементов

length = arrayWithoutEven.Length;

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0))

{

Console.Write("пуст");

}

Console.WriteLine("");

Console.WriteLine("");

}

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 3: // Третий пункт меню / Добавление новых элементов

counterRemain = 0;

Console.Clear();

Console.WriteLine("Введите число k - индекс,с которого будут добавляять новые элементы, 0<k<=n, являться целым числом");

string strK = Console.ReadLine();

Console.WriteLine();

okK = (int.TryParse(strK, out k)) & (k > 0) & (k <= counter);

while (!okK) // Проверка ввода K

{

Console.WriteLine("Введите число k - индекс,с которого будут добавляять новые элементы, 0<k<=n, являться целым числом");

strK = Console.ReadLine();

Console.WriteLine();

okK = (int.TryParse(strK, out k)) & (k > 0) & (k <= counter);

}

Console.Clear();

Console.WriteLine("Введите число n - количество новых элементов массива, n должна быть больше нуля и являться целым числом");

string strN2 = Console.ReadLine();

Console.WriteLine();

okN2 = (int.TryParse(strN2, out n2)) && (n2 > 0);

while (!okN2) // Проверка ввода N

{

Console.WriteLine("Введите число n - количество новых элементов массива, n должна быть больше нуля и являться целым числом");

strN2 = Console.ReadLine();

Console.WriteLine();

okN2 = (int.TryParse(strN2, out n2)) && (n2 > 0);

}

Console.Clear();

Console.Write("Ваш массив:");

int[] massExtended = new int[counter + n2]; // Расширенный массив

for (int i = 0; i <= k - 1; i++) // Добавление элементов до позиции К, если они есть

{

massExtended[i] = tmp[counterRemain];

counterRemain++;

if (massExtended[i] != 0)

{

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

}

}

for (int i = k; i <= k + n2 - 1; i++) // Добавление новых элементов

{

massExtended[i] = (int)rand.Next(1, 100);

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

}

for (int i = k + n2; i <= counter + n2 - 1; i++) // Добавление оставшихся элементов, если они остались

{

massExtended[i] = tmp[counterRemain];

counterRemain++;

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

}

tmp = massExtended;

length = massExtended.Length;

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 4: //Четвёртый пункт, циклическое смещение вправо

Console.Clear();

int[] massMove = new int[length]; // Новый массив, который будет свдинут

int m;

Console.WriteLine("Введите число M - количество позиций для смещения элементов, M должна являться целым положительным числом");

string strM = Console.ReadLine();

Console.WriteLine();

okM = (int.TryParse(strM, out m)) && (m > 0);

while (!okM) // Проверка ввода М

{

Console.WriteLine("Введите число M - количество позиций для смещения элементов, M должна являться целым положительным числом");

strM = Console.ReadLine();

Console.WriteLine();

okM = (int.TryParse(strM, out m)) && (m > 0);

}

Console.Clear();

Console.Write("Ваш массив:");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0))

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

for (int i = 0; i <= counter - 1; i++) // Смещение вправо

{

if (tmp[(((i + counter) - (m % counter)) % counter)] != 0)

{

massMove[i] = tmp[(((i + counter) - (m % counter)) % counter)];

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

}

}

}

tmp = massMove; // Присваивание к временному массиву

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 5: // Пятый пункт / Поиск элемента

Console.Clear();

counterCompare = 0;

okNeeded = false;

Console.WriteLine("Введите элемент, который надо будет найти в массиве, элемент должен являться целым числом:");

string strNeeded = Console.ReadLine();

Console.WriteLine();

okNeeded = (int.TryParse(strNeeded, out needed));

while (!okNeeded) // Проверка ввода искомого числа

{

Console.WriteLine("Введите элемент, который надо будет найти в массиве, элемент должен являться целым числом:");

strNeeded = Console.ReadLine();

Console.WriteLine();

okNeeded = (int.TryParse(strNeeded, out needed));

}

Console.Clear();

do // Поиск элемента

{

found = (tmp[counterCompare] == needed) && (counterCompare <= counter);

counterCompare = ++counterCompare;

if (found)

{

Console.WriteLine($"Найденный элемент находится на {counterCompare} позиции.");

Console.WriteLine($"Количество сравнений = {counterCompare}.");

break;

}

if (counterCompare >= counter) //Если прошёлся по массиву,а числа не нашёл

{

Console.WriteLine($"Элемент не найден в данном массиве!");

Console.WriteLine($"Количество сравнений = {counterCompare}.");

break;

}

} while (!found);

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

case 6: // Шестой пункт / Сортировка массива

Console.Clear();

Console.Write("Ваш массив: ");

if ((tmp.Length == 1 & tmp[0] == 0) || (counter == 1 & tmp[0] == 0)) // Пустой массив

{

Console.WriteLine("пуст");

Console.WriteLine("");

}

else

{

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

{

minimal = i;

for (int j = i; j < length; j++)

{

if (tmp[j] < tmp[minimal])

{

minimal = j;

}

tempNum = tmp[i];

tmp[i] = tmp[minimal];

tmp[minimal] = tempNum;

}

if (tmp[i] != 0)

{

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

}

}

}

Console.WriteLine();

Console.WriteLine("[1] - Вернуться в меню");

Console.WriteLine("");

Console.WriteLine("[2] - Завершить работу программы");

do

{

string strReturn = Console.ReadLine();

returnOk = int.TryParse(strReturn, out returnMenu);

switch (returnMenu)

{

case 1:

returnOk = true;

Console.Clear();

break;

case 2:

returnOk = true;

exitMenu = true;

break;

default:

returnOk = false;

exitMenu = false;

break;

}

} while (returnOk == false);

break;

}

}

else

{

Console.Clear();

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

}

} while (exitMenu == false);

exit = true;

break;

default:

Console.WriteLine("Такого пункта меню не существует, попробуйте ещё раз!");

break;

}

}

else

{

Console.WriteLine("Вы ввели некорректные данные, попробуйте ещё раз!");

}

} while (exit == false);

// break;

} while (exit == false);