Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Разработка программных систем»

Отчёт по лабораторной работе № 1

**Задание:** Вычислить среднее арифметическое значений всех элементов массива. Для отрицательных элементов использовать их абсолютные значения.

**Выполнил студент группы №485:**

Зобнин Илья Михайлович

**Проверили:**

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург

2020

# Постановка задачи

Составить программу для подсчёта среднего арифметического значений всех элементов массива. Для отрицательных элементов использовать их абсолютные значения.

В качестве исходных данных программа использует вводимый пользователем массив, либо считывает его из файла.

# Особые ситуации

* Если пользователь указал размер массива строго меньше единицы, то программа попросит пользователя ввести размер ещё раз.
* Если элементов массива в файле окажется меньше заявленного размера, программа попросит ввести путь к файлу заново.
* Если элементов массива в файле окажется больше заявленного размера, программа считает только те элементы, которые подпадают под размер.
* Если файл для чтения пуст, или не существует, или имеет недопустимое имя, программа попросит ввести путь к другому файлу.
* Если пользователь попытается сохранить результат работы программы в уже существующий файл с данными, программа оповестит пользователя и предложит перезаписать файл либо же указать другой файл.

# Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будут использоваться алгоритмы, блок-схемы которых представлены на рисунках 1 и 2.

С помощью первого алгоритма все отрицательные элементы массива преобразуются в свои абсолютные значения. Второй алгоритм занимается подсчётом среднего арифметического значения всех элементов массива.

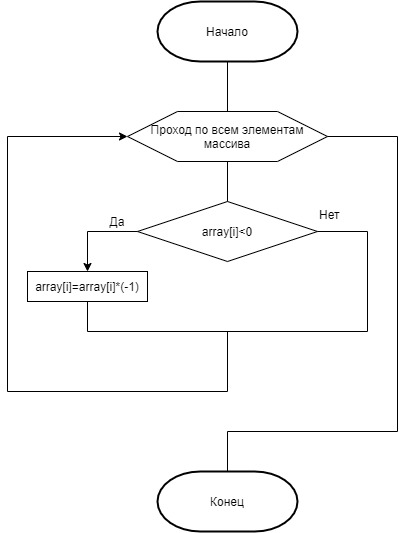


Рисунок 1 – Блок-схема алгоритма преобразования отрицательных элементов массива в положительные.

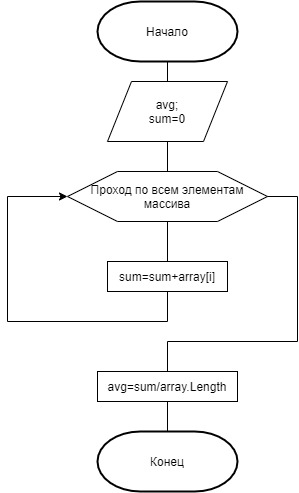


Рисунок 2 – Блок-схема расчёта среднего арифметического значения.

# Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| arr | int[] | Целочисленный массив значений, среднее арифметическое которого нужно рассчитать по заданию |
| result | double | Среднее арифметическое |
| menuItmes | MainMenuItems | Выбранный пользователем пункт меню |
| arrSize | int | Размер массива |
| rand | Random | Генератор случайных чисел |
| Продолжение таблицы 1 | | |
| path | string | Путь к файлу |
| color | ConsoleColor | Цвет текста в консоли |

# Структура программы

Программа разделена на пять классов, из которых один является основным и отвечает за запуск программы, а четыре оставшихся содержат в себе функции, необходимые для работы программы.

Класс Program.cs:

Таблица 2 – Функции, составляющие Класс Program.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| DoTask | Выполнение задания, то есть преобразование отрицательных элементов массива в положительные и расчёт среднего арифметического |
| Menu | Вывод пунктов меню в консоль |
| Main | Приветствие пользователя и цикличность программы |

Класс ArrayFilling.cs:

Таблица 3 – Функции, составляющие класс ArrayFilling.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| GetArrSize | Ввод и проверка размера массива |
| ManualInput | Ввод элементов массива пользователем вручную |
| RandomInput | Заполнение массива случайными значениями |
| FileFilling | Заполнение массива значениями из файла |
| GetArray | Вызов одного из трёх методов заполнения массива на основе выбранного пункта меню |

Класс ColorPrint.cs:

Таблица 4 – Функции, составляющие класс ColorPrint.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| PrintWithColor | Вывод цветного текста в консоль |
| ErrorPring | Вывод текста ошибки и предложения в консоль |

Класс Gets.cs:

Таблица 5 – Функции, составляющие класс Gets.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| MyParse | Преобразование текста в целое число с выводом сообщений об ошибках |
| GetInt | Получение целого числа из консоли |
| GetMenuItem | Получение пункта меню из консоли |
| AskYesNo | Получение «y» или «n» из консоли |
| ExtensionCheck | Проверка корректности расширения при вводе пользователем пути к файлу |
| GetFilePathForRead | Получение пути к файлу для чтения массива |
| GetFilePathForWrite | Получение пути к файлу для сохранения данных |

Класс ResultOutput.cs:

Таблица 6 – Функции, составляющие класс ResultOutput.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SaveArr | Запись в файл массива |
| SaveResult | Запись в файл среднего арифметического |
| SaveArrAndResult | Запись в файл массива и среднего арифметического |
| OutputInFile | Вывод меню по сохранению в файл и вызов соответствующей функции |
| OutputResult | Вывод среднего арифметического в консоль и вывод меню по сохранению в файл |

# Описание хода выполнения лабораторной работы

* При работе программы с файлами нужно было добавить проверки на валидность имени файла, а также на то, создан ли файл или нет при сохранении.
* При получении пользовательского ввода необходимо было добавить проверку, на его корректность.
* При выполнении лабораторной работы были созданы тесты для проверки правильности работы алгоритма.

# Результат работы программы

В результате работы программа выводит местоположение искомого слова или же сообщение об отсутствии данного слова в исходном тексте.

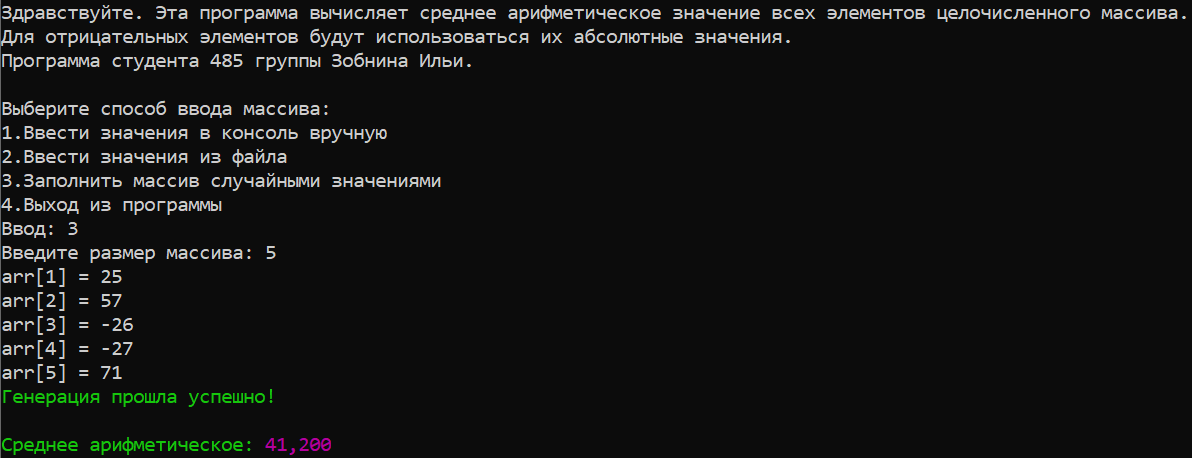


Рисунок 3 – Работа программы при заполнении массива случайными числами.

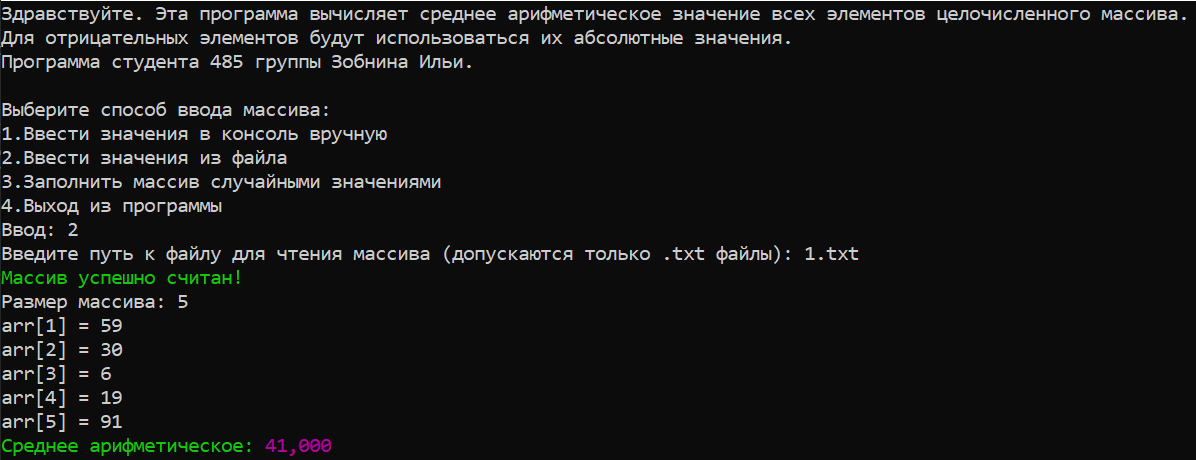


Рисунок 4 – Работа программы при заполнении массива значениями из файла.

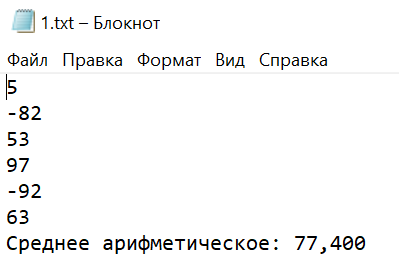


Рисунок 5 – Результат работы программы, записанный в файл.

# Текст программы

**[--- Начало программы ---]**

**// Program.cs**

**// Лабораторная работа №1.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Linq;

namespace Lab1

{

public static class Program

{

public enum MainMenuItems

{

manualFilling = 1,

fileFilling,

randomFilling,

exitProgram

}

static string NL = Environment.NewLine;

static MainMenuItems Menu()

{

Console.Write("Выберите способ ввода массива:" + NL +

"1.Ввести значения в консоль вручную" + NL +

"2.Ввести значения из файла" + NL +

"3.Заполнить массив случайными значениями" + NL +

"4.Выход из программы" + NL +

"Ввод: ");

return Gets.GetMenuItem<MainMenuItems>();

}

public static double DoTask(int[] arr)

{

int[] arrCopy = new int[arr.Length];

Array.Copy(arr, arrCopy, arr.Length);

try

{

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

if (arrCopy[i] < 0)

arrCopy[i] \*= -1;

return arrCopy.Average();

}

catch

{

ColorPrint.ErrorPrint("Вам как-то удалось дойти до этой части программы" +

"и сломать её. ", "Поздравляю, вы профессиональный хакер!");

return double.NaN;

}

}

static void Main()

{

Console.WriteLine("Здравствуйте. Эта программа вычисляет среднее арифметическое значение всех " +

"элементов целочисленного массива." + NL +

"Для отрицательных элементов будут использоваться их абсолютные значения." + NL +

"Программа студента 485 группы Зобнина Ильи." + NL);

MainMenuItems menuChoice;

int[] arr;

double result;

while (true)

{

menuChoice = Menu();

if (menuChoice == MainMenuItems.exitProgram)

break;

arr = ArrayFilling.GetArray(menuChoice);

result = DoTask(arr);

ResultOutput.OutputResult(arr, result);

}

}

}

}

**// ArrayFilling.cs**

**// Лабораторная работа №1.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.IO;

namespace Lab1

{

static class ArrayFilling

{

static string NL = Environment.NewLine;

static int GetArrSize(Program.MainMenuItems wayToFillArray)

{

const int minArrSize = 0;

const int manualInputLim = 15;

const string newValueAsk = "Введите размер массива ещё раз: ";

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

int arrSize;

while (true)

{

arrSize = Gets.GetInt(newValueAsk);

if (arrSize <= minArrSize)

{

ColorPrint.ErrorPrint("Размер массива должен быть больше нуля! ", newValueAsk);

}

else

{

try

{

int[] memLimTest = new int[arrSize];

}

catch (OutOfMemoryException)

{

ColorPrint.ErrorPrint("Слишком большой размер массива! ", newValueAsk);

continue;

}

if (arrSize > manualInputLim && wayToFillArray == Program.MainMenuItems.manualFilling)

{

ColorPrint.ErrorPrint("Размер массива очень большой для ввода вручную. ",

"Вы точно хотите заполнять массив введённой длины вручную?" + NL +

"y - да, n - нет: ");

if (Gets.AskYesNo())

break;

else

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

}

else

break;

}

}

return arrSize;

}

static int[] ManualFilling()

{

int arrSize = GetArrSize(Program.MainMenuItems.manualFilling);

int[] arr = new int[arrSize];

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

{

Console.Write($"arr[{i + 1}] = ");

arr[i] = Gets.GetInt($"arr[{i + 1}] = ");

}

ColorPrint.PrintWithColor(ConsoleColor.Green, "Все элементы введены успешно!" + NL + NL);

return arr;

}

static int[] RandomFilling()

{

Random rand = new Random();

const int minRand = -100;

const int maxRand = 100;

int arrSize = GetArrSize(Program.MainMenuItems.randomFilling);

int[] arr = new int[arrSize];

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

{

arr[i] = rand.Next(minRand, maxRand);

Console.WriteLine($"arr[{i + 1}] = {arr[i]}");

}

ColorPrint.PrintWithColor(ConsoleColor.Green, "Генерация прошла успешно!" + NL + NL);

return arr;

}

static int[] FileFilling()

{

int[] arr = null;

bool valueReadSuccess;

string path;

int arrSize;

do

{

path = Gets.GetFilePathForRead();

using (StreamReader streamReader = new StreamReader(path))

{

valueReadSuccess = Gets.MyParse(streamReader.ReadLine(), "Исправьте 1-ю строку с размером " +

"массива." + NL, out arrSize);

if (!valueReadSuccess)

continue;

arr = new int[arrSize];

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

{

valueReadSuccess = Gets.MyParse(streamReader.ReadLine(), $"Исправьте {i + 2} строку в " +

"файле." + NL, out arr[i]);

if (!valueReadSuccess)

break;

}

}

} while (!valueReadSuccess);

ColorPrint.PrintWithColor(ConsoleColor.Green, "Массив успешно считан!" + NL);

Console.WriteLine($"Размер массива: {arrSize}");

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

Console.WriteLine($"arr[{i + 1}] = {arr[i]}");

return arr;

}

public static int[] GetArray(Program.MainMenuItems menuItem)

{

int[] array = null;

switch (menuItem)

{

case Program.MainMenuItems.manualFilling:

array = ManualFilling();

break;

case Program.MainMenuItems.fileFilling:

array = FileFilling();

break;

case Program.MainMenuItems.randomFilling:

array = RandomFilling();

break;

}

return array;

}

}

}

**// ColorPrint.cs**

**// Лабораторная работа №1.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

namespace Lab1

{

static class ColorPrint

{

public static void PrintWithColor(ConsoleColor color, string message)

{

Console.ForegroundColor = color;

Console.Write(message);

Console.ResetColor();

}

public static void ErrorPrint(string errorMessage, string askMessage)

{

PrintWithColor(ConsoleColor.Red, errorMessage);

PrintWithColor(ConsoleColor.Yellow, askMessage);

}

}

}

**// Gets.cs**

**// Лабораторная работа №1.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.IO;

namespace Lab1

{

static class Gets

{

public static bool MyParse(string stringValue, string askMessage, out int value)

{

try

{

value = int.Parse(stringValue);

return true;

}

catch (ArgumentNullException)

{

ColorPrint.ErrorPrint("Число не было введено! ", askMessage);

}

catch (FormatException)

{

ColorPrint.ErrorPrint("Неверный формат! ", askMessage);

}

catch (OverflowException)

{

ColorPrint.ErrorPrint("Слишком большое или маленькое число! ", askMessage);

}

value = 0;

return false;

}

public static int GetInt(string askMessage)

{

int value;

string stringValue;

bool parseSuccess;

do

{

stringValue = Console.ReadLine();

parseSuccess = MyParse(stringValue, askMessage, out value);

} while (!parseSuccess);

return value;

}

public static EnumType GetMenuItem<EnumType>()

{

int value = GetInt("Введите пункт меню ещё раз: ");

while (!Enum.IsDefined(typeof(EnumType), value))

{

ColorPrint.ErrorPrint("Такого пункта меню нет! ", "Введите пункт меню ещё раз: ");

value = GetInt("Введите пункт меню ещё раз: ");

}

return (EnumType)Enum.ToObject(typeof(EnumType), value);

}

public static bool AskYesNo()

{

char yn;

while (true)

{

yn = Console.ReadKey().KeyChar;

Console.WriteLine();

if (yn != 'y' && yn != 'n')

{

ColorPrint.ErrorPrint("Неверный формат! ", "Введите y/n: ");

}

else

break;

}

return true ? yn == 'y' : false;

}

static bool ExtensionCheck(string filePath)

{

const int minPathLength = 4;

const string newValueAsk = "Введите путь ещё раз: ";

if (filePath.Length > minPathLength)

{

if (filePath.EndsWith(".txt"))

{

return true;

}

else

{

ColorPrint.ErrorPrint("Неверное расширение! ", newValueAsk);

return false;

}

}

else

{

ColorPrint.ErrorPrint("Слишком короткий путь к файлу! ", newValueAsk);

return false;

}

}

public static string GetFilePathForRead()

{

string path;

const string newValueAsk = "Введите путь ещё раз: ";

Console.Write("Введите путь к файлу для чтения массива (допускаются только .txt файлы): ");

while (true)

{

path = Console.ReadLine();

if (ExtensionCheck(path))

{

try

{

using (StreamReader streamReader = new StreamReader(path)) { }

break;

}

catch (FileNotFoundException)

{

ColorPrint.ErrorPrint("Файла по указанному пути не существует! ", newValueAsk);

}

catch (UnauthorizedAccessException)

{

ColorPrint.ErrorPrint("Доступ к файлу запрещён! ", newValueAsk);

}

catch (System.NotSupportedException)

{

ColorPrint.ErrorPrint("Запрещённое имя файла! ", newValueAsk);

}

}

}

return path;

}

public static string GetFilePathForWrite()

{

string path;

Console.Write("Введите путь для сохранения результата в файл (допускаются только .txt файлы): ");

while (true)

{

path = Console.ReadLine();

if (ExtensionCheck(path))

{

try

{

if (File.Exists(path))

{

ColorPrint.ErrorPrint("Файл уже существует. ", "y - перезаписать, n - указать другой путь: ");

if (AskYesNo())

{

using (StreamWriter streamWriter = new StreamWriter(path)) { }

break;

}

else

{

Console.Write("Введите путь к файлу: ");

continue;

}

}

else

{

using (StreamWriter streamWriter = new StreamWriter(path)) { }

return path;

}

}

catch

{

ColorPrint.ErrorPrint("Доступ к файлу запрещён или было использовано запрещённое имя! ", "Введите путь ещё раз: ");

}

}

}

return path;

}

}

}

**// ResultOutput.cs**

**// Лабораторная работа №1.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.IO;

namespace Lab1

{

static class ResultOutput

{

enum SaveFileMenu

{

saveOnlyArr = 1,

saveOnlyResult,

saveArrAndResult,

saveNothing

}

static string NL = Environment.NewLine;

static void SaveArr(int[] arr)

{

using (StreamWriter streamWriter = new StreamWriter(Gets.GetFilePathForWrite()))

{

streamWriter.WriteLine(arr.Length.ToString());

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

streamWriter.WriteLine(arr[i].ToString());

}

ColorPrint.PrintWithColor(ConsoleColor.Green, "Успешно сохранено!" + NL + NL);

}

static void SaveResult(double result)

{

using (StreamWriter streamWriter = new StreamWriter(Gets.GetFilePathForWrite()))

{

streamWriter.Write("Среднее арифметическое: " + result.ToString("F3"));

}

ColorPrint.PrintWithColor(ConsoleColor.Green, "Успешно сохранено!" + NL + NL);

}

static void SaveArrAndResult(int[] arr, double result)

{

using (StreamWriter streamWriter = new StreamWriter(Gets.GetFilePathForWrite()))

{

streamWriter.WriteLine(arr.Length.ToString());

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

streamWriter.WriteLine(arr[i].ToString());

streamWriter.Write("Среднее арифметическое: " + result.ToString("F3"));

}

ColorPrint.PrintWithColor(ConsoleColor.Green, "Успешно сохранено!" + NL + NL);

}

static void OutputInFile(int[] arr, double result)

{

Console.Write("Сохранение в файл:" + NL +

"1.Сохранить исходный массив" + NL +

"2.Сохранить среднее арифметическое" + NL +

"3.Сохранить исходный массив и среднее арифметическое" + NL +

"4.Ничего не сохранять" + NL +

"Ввод: ");

SaveFileMenu menuChoice = Gets.GetMenuItem<SaveFileMenu>();

switch (menuChoice)

{

case SaveFileMenu.saveOnlyArr:

SaveArr(arr);

break;

case SaveFileMenu.saveOnlyResult:

SaveResult(result);

break;

case SaveFileMenu.saveArrAndResult:

SaveArrAndResult(arr, result);

break;

case SaveFileMenu.saveNothing:

Console.WriteLine();

break;

}

}

public static void OutputResult(int[] arr, double result)

{

ColorPrint.PrintWithColor(ConsoleColor.Green, "Среднее арифметическое: ");

ColorPrint.PrintWithColor(ConsoleColor.Magenta, result.ToString("F3") + NL + NL);

OutputInFile(arr, result);

}

}

}

**[--- Конец программы ---]**