**Лабораторная работа №17**

**ФОРМАТИРОВАННЫЙ ВЫВОД. ВВОД-ВЫВОД В ФАЙЛЫ**

**Цели:**

1.Научиться выполнять форматированный вывод в окно консоли.

2.Научиться работать с текстовыми файлами.

**Приложение Lab17\_01 Вывод таблицы умножения**

***Задание*:** Добавить в класс **Helper** процедуру **MultiplicationTable()**, выводящую в окно консоли таблицу умножения чисел от 1 до 9. Сопроводить разрабатываемую процедуру соответствующими XML-комментариями.

Разработать консольное приложение, которое будет использовать процедуру **MultiplicationTable()**.

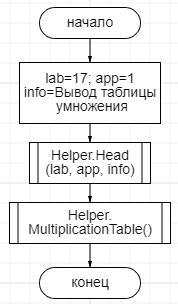
***Константы:*** отсутствуют.

***Переменные:*** lab - № Лабораторной работы; app - № Приложения; info – информация о задаче;

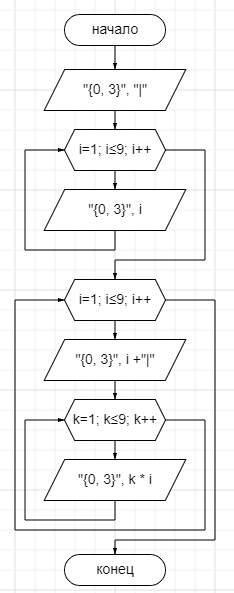
***Исходные данные:*** отсутствуют.

***Результат:*** Программа выводит таблицу умножения.

***Блок-схема:***



***Блок-схема MultiplicationTable():***



***Код приложения:***

**Листинг 1.1 – Код консольного приложения Lab17\_01**

namespace PuzanovVE.OP.Lab17\_01

{

class Program

{

static void Main(string[] args)

{

//Стандартная шапка

int lab = 17; int app = 1;

//информация о задаче

string info = "Вывод таблицы умножения";

//Вывод шапки

Helper.Head(lab, app, info);

//основной код программы

Helper.MultiplicationTable();

Console.ReadLine();

}

}

}

**Листинг 1.2 – Код функции MultiplicationTable()**

/// <summary>

/// Вывод таблицы умножения

/// </summary>

public static void MultiplicationTable()

{

Console.Write("{0, 3}", "|");

for (int i = 1; i <= 9; i++)

{

Console.Write("{0, 3}", i);

}

Console.WriteLine("\n------------------------------");

for (int i = 1; i <= 9; i++)

{

Console.Write("{0, 3}", i +"|");

for (int k = 1; k <= 9; k++)

{

Console.Write("{0, 3}", k \* i);

}

Console.WriteLine();

}

}

**Листинг 1.3 – Код функции Head(int lab, int app, string info)**

/// <summary>

/// Вывод стандартной шапки

/// </summary>

/// <param name="app">

/// № приложения

/// </param>

/// <param name="info">

/// Информация о задаче

/// </param>

/// <param name="lab">

/// № лабараторной

/// </param>

public static void Head(int lab, int app, string info)

{

Console.WriteLine("Лабораторная работа №" + lab);

Console.WriteLine("Приложение Lab" + lab + "\_0" + app);

Console.WriteLine("Разработчик: Пузанов В. Е.");

Console.WriteLine("------------------------------------------------------");

//информация о задаче

Console.WriteLine(info);

Console.WriteLine("------------------------------------------------------");

}

**Тестирование:**

Контрольный пример:

Исходные данные: отсутствуют.

Результат: Программа выводит таблицу умножения.

Результат работы программы для указанных исходных данных приведён на рисунке 1.1

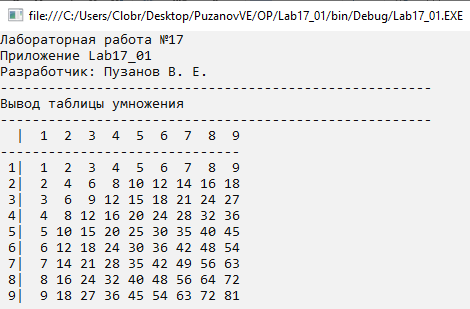


Рисунок 1.1 –Результат работы программы Lab17\_01

**Приложение Lab17\_02. Табулирование функции.**

***Задание*:** Добавить в класс Helper процедуру TableFuncArctg(), выводящую в окно консоли таблицу значений функции f (x)  arctg(x) на отрезке [a; b] с шагом h. Сопроводить разрабатываемую процедуру соответствующими XML- комментариями.

Разработать консольное приложение, которое будет использовать процедуру TableFuncArctg().

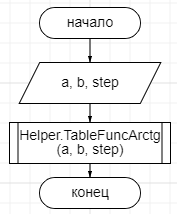
***Константы:*** отсутствуют.

***Переменные:*** a – переменная типа double – левая граница отрезка; b – переменная типа double – правая граница отрезка; step – переменная типа double – шаг;

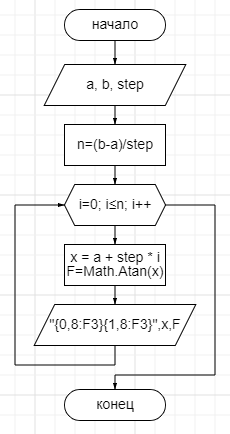
***Исходные данные:*** a; b; step;

***Результат:*** Программа табулирует функцию.

***Блок-схема:***



***Блок-схема TableFuncArctg(double a, double b, double step):***



***Код приложения:***

**Листинг 2.1 – Код консольного приложения Lab17\_02**

namespace PuzanovVE.OP.Lab17\_02

{

class Program

{

static void Main(string[] args)

{

//Стандартная шапка

int lab = 17; int app = 2;

//информация о задаче

string info = "Табулирование функции";

//Вывод шапки

Helper.Head(lab, app, info);

//основной код программы

Console.WriteLine("Введите левую границу отрезка.");

double a = double.Parse(Console.ReadLine());

Console.WriteLine("Введите правую границу отрезка.");

double b = double.Parse(Console.ReadLine());

Console.WriteLine("Введите шаг.");

double step = double.Parse(Console.ReadLine());

Helper.TableFuncArctg(a, b, step);

Console.ReadLine();

}

}

}

**Листинг 2.2 – Код функции TableFuncArctg(a, b, step)**

/// <summary>

/// Табулирование функции

/// </summary>

/// <param name="a">

/// Левая граница отрезка

/// </param>

/// <param name="b">

/// Правая граница отрезка

/// </param>

/// <param name="step">

/// Шаг

/// </param>

public static void TableFuncArctg(double a, double b, double step)

{

double n = (b - a) / step;

Console.Write("{0,8}", "x ");

Console.Write("|");

Console.Write("{0,8}", "f(x)");

Console.WriteLine("\n------------------");

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

{

double x = a + step \* i;

double F = Math.Atan(x);

Console.WriteLine("{0,8:F3}{1,8:F3}", x, F);

}

}

**Тестирование:**

Контрольный пример 1:

Исходные данные:

a = 1; b = 10; step = 5;

Результат:

Программа табулирует функцию.

Результат работы программы для указанных исходных данных приведён на рисунке 2.1

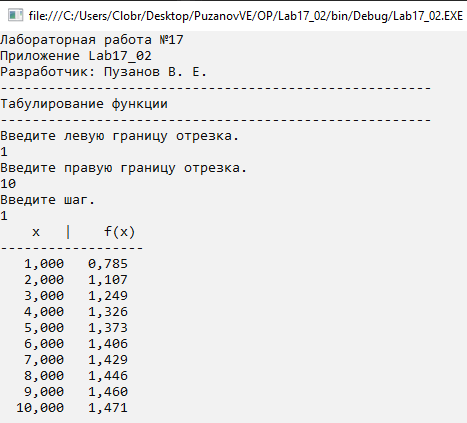


Рисунок 2.1 –Результат работы программы Lab17\_02

Контрольный пример 2:

Исходные данные:

a = 1; b = 5; step = 0,5;

Результат:

Программа табулирует функцию.

Результат работы программы для указанных исходных данных приведён на рисунке 2.2

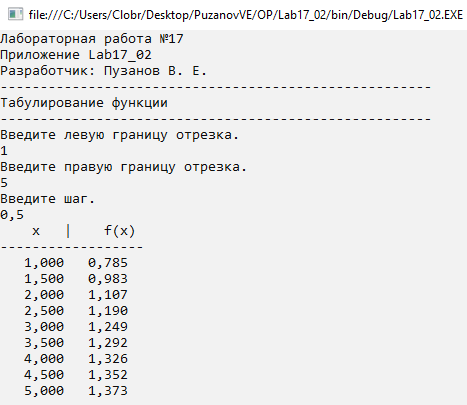


Рисунок 2.2 –Результат работы программы Lab17\_02

Контрольный пример 3:

Исходные данные:

Исходные данные:

a = -2; b = 1; step = 0,25;

Результат:

Программа табулирует функцию.

Результат работы программы для указанных исходных данных приведён на рисунке 2.3

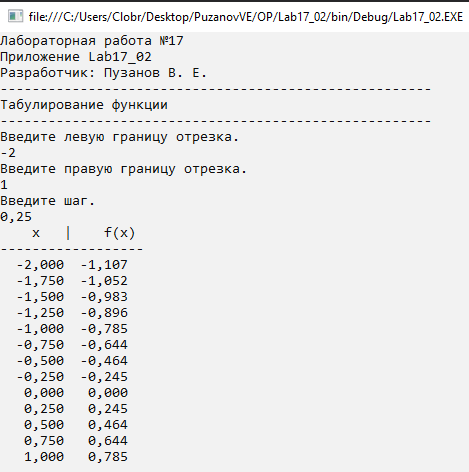


Рисунок 2.3 –Результат работы программы Lab17\_02

**Приложение Lab17\_03.Разработка и использование метода записи одномерного массива в текстовый файл**

***Задание*:** Добавить в класс **Helper** метод **WriteArrayFile()**, выводящий одномерный массив в указанный текстовый файл. Метод должен принимать в качестве параметров путь к файлу и одномерный массив вещественных чисел. Сопроводить разрабатываемый метод XML-комментариями. Разработать консольное приложение, которое, используя метод **WriteArrayFile()**, записывало бы в файл «H:\массив.txt», введенный в консоль, одномерный массив.

***Константы:*** path – переменная типа string – хранит в себе ссылку на txt файл.

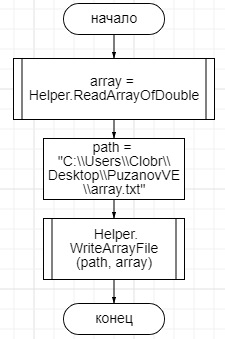
***Переменные:*** array – массив;

***Исходные данные:*** отсутствуют.

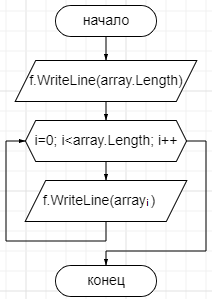
***Результат:***

Программа записывает массив в txt файл.

***Блок-схема:***



***Блок-схема WriteArrayFile(string path, double[] array):***



***Код приложения:***

**Листинг 3.1 – Код консольного приложения Lab17\_03**

namespace PuzanovVE.OP.Lab17\_03

{

class Program

{

static void Main(string[] args)

{

//Стандартная шапка

int lab = 17; int app = 3;

//информация о задаче

string info = "Запись массива в txt файл";

//Вывод шапки

Helper.Head(lab, app, info);

//основной код программы

double[] array = Helper.ReadArrayOfDouble();

string path = "C:\\Users\\Clobr\\Desktop\\PuzanovVE\\array.txt";

Helper.WriteArrayFile(path, array);

Console.ReadLine();

}

}

}

**Листинг 3.2 – WriteArrayFile(string path, double[] array)**

/// <params>

/// Выводит одномерный массив вещественных чисел в текстовый файл

/// </params>

/// <param name="array">

/// Массив, выводимый в текстовый файл

/// </param>

/// <param name="path">

/// Путь к файлу

/// </param>

public static void WriteArrayFile(string path, double[] array)

{

StreamWriter f = new StreamWriter(path, false);

f.WriteLine(array.Length);

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

{

f.WriteLine(array[i]);

}

f.Close();

}

**Тестирование:**

Контрольный пример 1:

Исходные данные:

Массив.

Результат:

Запись массива в txt файл.

Результат работы программы для указанных исходных данных приведён на рисунках (3.1.1 - 3.1.2)

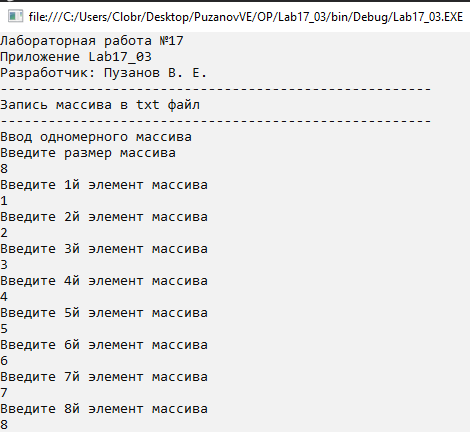


Рисунок 3.1.1 –Результат работы программы Lab17\_03

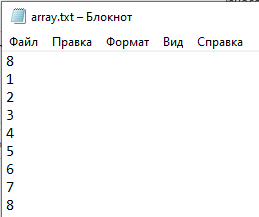


Рисунок 3.1.2 –txt файл после выполнения программы Lab17\_03

Контрольный пример 2:

Исходные данные:

Массив.

Результат:

Запись массива в txt файл.

Результат работы программы для указанных исходных данных приведён на рисунках (3.2.1 - 3.2.2)

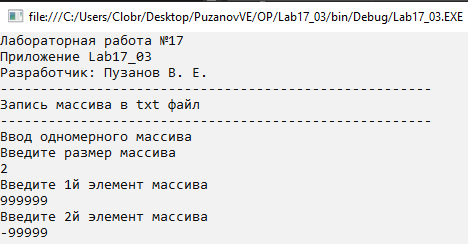


Рисунок 3.2.1 –Результат работы программы Lab17\_03

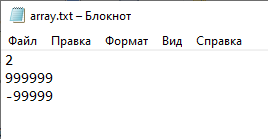


Рисунок 3.2.2 –txt файл после выполнения программы Lab17\_03

Контрольный пример 3:

Исходные данные:

Массив.

Результат:

Запись массива в txt файл.

Результат работы программы для указанных исходных данных приведён на рисунках (3.3.1 - 3.3.2)

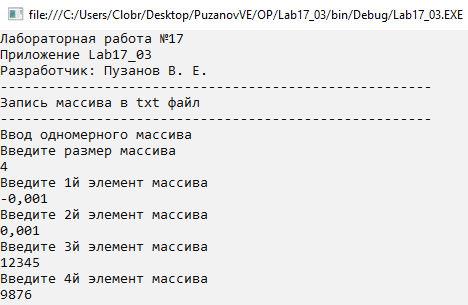


Рисунок 3.3.1 –Результат работы программы Lab17\_03

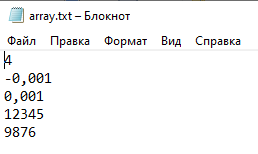


Рисунок 3.2.2 –txt файл после выполнения программы Lab17\_03

**Приложение Lab17\_04.Разработка и использование метода чтения одномерного массива из текстового файла**

***Задание*:** Реализовать в классе **Helper** функцию **ReadArrayFile()**, читающую одномерный массив из указанного текстового файла. Функция должна принимать путь к файлу в качестве параметра и возвращать в точку вызова считанный одномерный массив. Сопроводить метод-функцию XML-комментариями.

Разработать консольное приложение, которое, используя разработанный метод **ReadArrayFile()**, читало бы из файла «H:\массив.txt» одномерный массив и выводило его на консоль.

***Константы:*** path – переменная типа string – хранит в себе ссылку на txt файл.

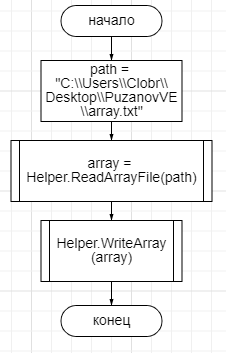
***Переменные:*** array – массив;

***Исходные данные:*** отсутствуют.

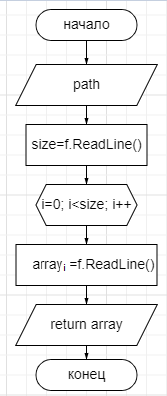
***Результат:***

Программа считывает массив из txt файла.

***Блок-схема:***



***Блок-схема ReadArrayFile(string path):***



***Код приложения:***

**Листинг 3.1 – Код консольного приложения Lab17\_04**

namespace PuzanovVE.OP.Lab17\_04

{

class Program

{

static void Main(string[] args)

{

//Стандартная шапка

int lab = 17; int app = 4;

//информация о задаче

string info = "Чтение массива из txt файла";

//Вывод шапки

Helper.Head(lab, app, info);

//основной код программы

string path = "C:\\Users\\Clobr\\Desktop\\PuzanovVE\\array.txt";

double[] array = Helper.ReadArrayFile(path);

Helper.WriteArray(array);

Console.ReadLine();

}

}

}

**Листинг 3.2 – ReadArrayFile(string path)**

/// <params>

/// Выводит одномерный массив вещественных чисел в текстовый файл

/// </params>

/// <param name="path">

/// Путь к файлу

/// </param>

public static double[] ReadArrayFile(string path)

{

StreamReader f = new StreamReader(path);

int size = int.Parse(f.ReadLine());

double[] array = new double[size];

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

{

array[i] = double.Parse(f.ReadLine());

}

f.Close();

return array;

}

**Тестирование:**

Контрольный пример 1:

Исходные данные:

Массив.

Результат:

Чтение массива из txt файла.

Результат работы программы для указанных исходных данных приведён на рисунках (4.1.1 - 4.1.2)

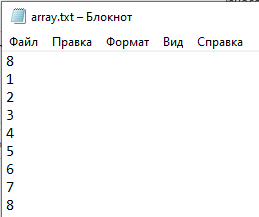


Рисунок 4.1.1 – txt файл перед выполнением программы Lab17\_04

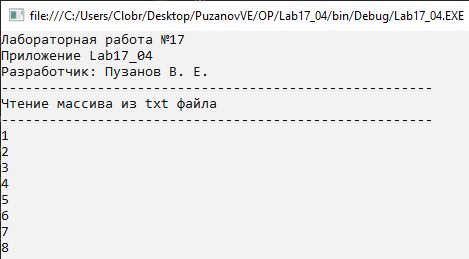


Рисунок 4.1.1 –Результат работы программы Lab17\_04

Контрольный пример 2:

Исходные данные:

Массив.

Результат:

Чтение массива из txt файла.

Результат работы программы для указанных исходных данных приведён на рисунках (4.2.1 - 4.2.2)

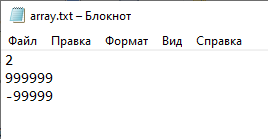


Рисунок 4.2.1 – txt файл перед выполнением программы Lab17\_04

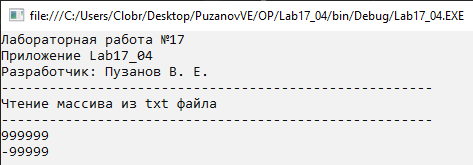


Рисунок 4.2.1 –Результат работы программы Lab17\_04

Контрольный пример 3:

Исходные данные:

Массив.

Результат:

Чтение массива из txt файла.

Результат работы программы для указанных исходных данных приведён на рисунках (4.3.1 - 4.3.2)

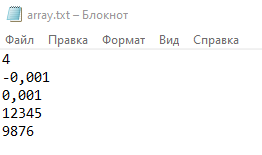


Рисунок 4.3.1 – txt файл перед выполнением программы Lab17\_04

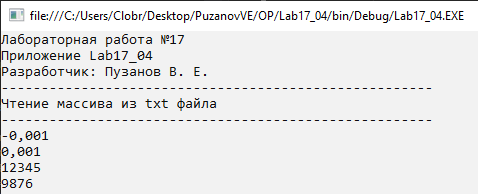


Рисунок 4.3.1 –Результат работы программы Lab17\_04

Выполнил студент Пузанов В. Е., ФИТУ 010304-КМСб-о22

Проверил ст. преподаватель каф. ПМ Черноиван Д.Н.