

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 423 |

Отчёт по контрольной работе №3

Вариант № 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 423 |  |  |  | Ефремов Иван Андреевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Макарук Роман Валерьевич |
|  |  |  |  | Федин Алексей Константинович |

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc134647499)

[2 Исходные данные 3](#_Toc134647500)

[3 Особые ситуации 3](#_Toc134647501)

[4 Математические методы и алгоритмы решения задач 4](#_Toc134647502)

[5 Форматы представления данных 4](#_Toc134647503)

[6 Структура программы 5](#_Toc134647504)

[7 Блок-схема алгоритма решения задачи 9](#_Toc134647505)

[8 Описание хода выполнения 13](#_Toc134647506)

[9 Результаты работы программы 14](#_Toc134647507)

[10 Выводы по заданию 16](#_Toc134647508)

[11 Исходный код полученного программного решения 16](#_Toc134647509)

## 1 Постановка задачи

Необходимо написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты (при их наличии). При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение.

Заданная функция:

## 2 Исходные данные

Исходные данные состоят из границ построения функции, коэффициентов функции и шага, введенных пользователем с клавиатуры в форму.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

-если при запросе числа пользователь ввёл другие символы, то программа укажет на необходимость ввода целых чисел и попросит повторить ввод.

## 4 Математические методы и алгоритмы решения задач

Согласно постановке задачи, для получения необходимых результатов будут использоваться операторы “>”, “≥”, “<”, “≤”, “=”, “+”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 Форматы представления данных Таблица 1 - Основные переменные, используемые в программе   |  |  |  | | --- | --- | --- | | Имя | Тип | Описание | | array | double[] | Массив элементов | | requestedElement | double | Элемент, индекс которого программа должна найти в процессе работы | | requestedIndex | int | Индекс запрошенного элемента в массиве |    6 Структура программы Таблица 2 - Модули программы   |  |  | | --- | --- | | Название модуля | Описание | | Program | Зацикленное меню и основное тело программы | | Information | Вывод сообщений о работе программы в консоль | | DataInputAndOutput | Ввод данных с файла/консоли/случайными значениями, вывод в файл | | ArrayManipulation | Алгоритм сортировки массива и бинарный поиск | |  |

Таблица 3 – Основные функции модуля Information

|  |  |
| --- | --- |
| Имя | Описание |
| Greeting | Вывод приветственного сообщения |
| Options | Вывод опций программы |
| AskToSave | Вывод сообщения, хочет ли пользователь сохранить данные в файл. |
| FileInputInfo | Вывод сообщения с требованиями к расположению информации в файле |
| AskToOverwrite | Вывод сообщения, хочет ли пользователь перезаписать данные в файл. |
| SaveSuccess | Вывод сообщения об успешном сохранении данных в файл |

Таблица 4 – Функции модуля DataInputAndOutput

|  |  |
| --- | --- |
| Имя | Описание |
| ReadDouble | Считывание рационального числа из консоли |
| ReadInt | Считывание целого числа из консоли |
| ReadPath | Считывание пути к файлу |
| SplitString | Разделение строки по пробелам при ввода с файла |
| SaveData | Сохранение данных в файл |
| ConsoleInput | Ввод данных с консоли |
| RandomInput | Генерация случайных данных |
| FileInput | Ввода данных с файла |

Таблица 5 – Функции модуля ArrayManipulation

|  |  |
| --- | --- |
| Имя | Описание |
| IsArraySorted | Проверка, отсортирован ли массив |
| Swap | Перестановка двух элементов в массиве |
| InsertionSort | Алгоритм сортировки вставками |
| BinarySearch | Алгоритм бинарного поиска |

## 7 Блок-схема алгоритма решения задачи

На рисунке 1 представлена блок-схема алгоритма бинарного поиска элемента.



Рисунок 1 – Блок-схема алгоритма

## 8 Описание хода выполнения

Во время работы, программа предлагает пользователю три варианта ввода данных (из консоли, из файла и генерация случайных значений). Если пользователь выбрал любой вариант ввода, кроме ввода из файла. То после ввода/генерации значений программа предложит пользователю сохранить эти значения в файл. После этого программа, если это необходимо, отсортирует массив по возрастанию и запросит элемент, индекс которого необходимо найти. После ввода элемента, программа найдет его позицию в массиве и выведет сообщение. Если элемент находится в массиве, то программа выведет «Requested element of {запрошенный элемент} is located at position {позиция элемента} in ascending sorted array». Если массив не содержит такого элемента, то программа выведет «Array does not contain requested element of {запрошенный элемент. После этого программа предложит пользователю сохранить результат работы в файл.

## 9 Результаты работы программы

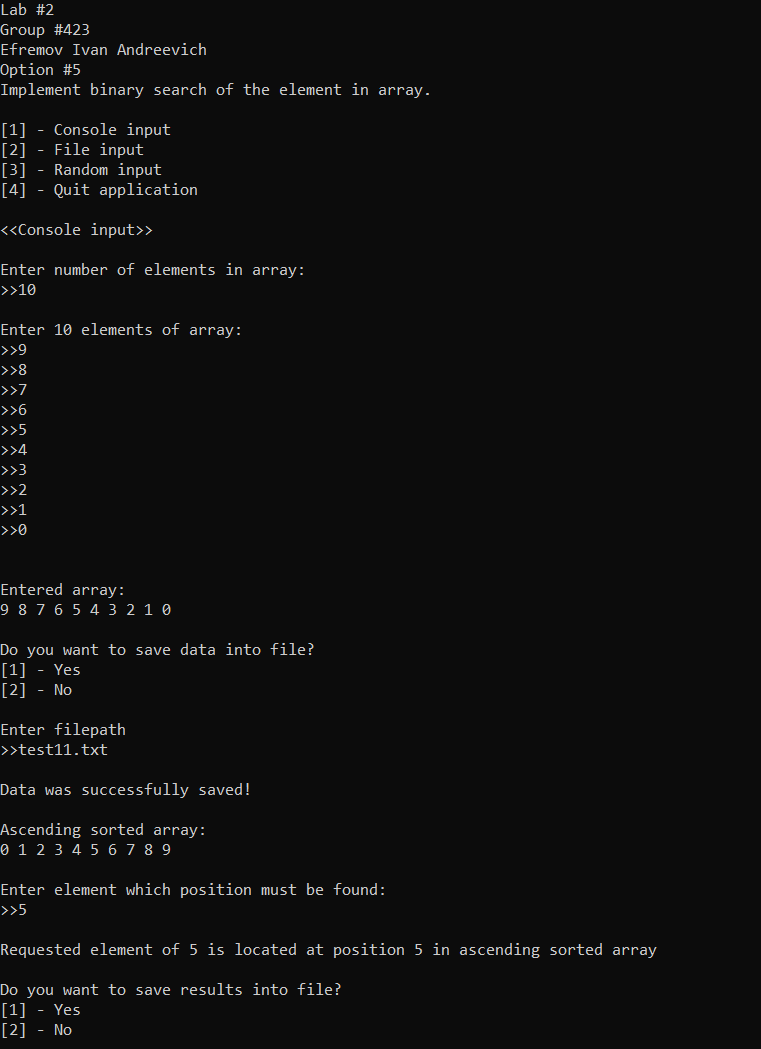


Рисунок 2 – Экранная копия результата работы разработанной программы

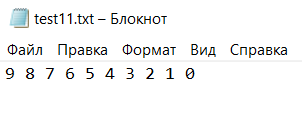


Рисунок 3 – Данные, сохраненные в файл

## 10 Выводы по заданию

В ходе выполнения задания были изучены основы работы с языком С# и реализацией классов, а также были получены навыки работы с многофайловыми проектами.

## 11 Исходный код полученного программного решения

Файл Program.cs:

using Lab1;

using System.Collections.Generic;

namespace Lab2

{

internal class Program

{

static void Main()

{

Information.Greeting();

Information.Options();

double[] array = new double[1];

while (true)

{

switch (Console.ReadKey(true).Key)

{

case ConsoleKey.D1:

DataInputAndOutput.ConsoleInput(out array);

break;

case ConsoleKey.D2:

DataInputAndOutput.FileInput(out array);

break;

case ConsoleKey.D3:

DataInputAndOutput.RandomInput(out array);

break;

case ConsoleKey.D4:

Console.WriteLine($"{Environment.NewLine}Programm finished it's work!");

Environment.Exit(0);

break;

default:

continue;

}

ArrayManipulation.InsertionSort(ref array);

Console.WriteLine("Ascending sorted array:");

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

Console.Write($"{array[i]} ");

Console.WriteLine();

Console.WriteLine($"{Environment.NewLine}Enter element which position must be found:");

double requestedElement = DataInputAndOutput.ReadInt(">>");

Console.WriteLine();

int requestedPosition = ArrayManipulation.BinarySearch(array, requestedElement);

if (requestedPosition == -1)

Console.WriteLine($"Array does not contain requested element of {requestedElement} :(");

else

Console.WriteLine($"Requested element of {requestedElement} is located at " +

$"position {requestedPosition} in ascending sorted array");

Information.AskToSave("results");

if (Console.ReadKey(true).Key == ConsoleKey.D1)

DataInputAndOutput.SaveData([double.NaN, requestedElement, requestedPosition]);

Information.Options();

}

}

}

}

Файл DataInputAndOutput.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1

{

static class DataInputAndOutput

{

private static double ReadDouble(string message)

{

bool isCorrectInput = false;

double value = 0;

while (!isCorrectInput)

{

Console.Write(message);

string? temp = Console.ReadLine();

if (!double.TryParse(temp, out value))

Console.WriteLine("Invalid input! Numbers only allowed!");

else isCorrectInput = true;

}

return value;

}

public static int ReadInt(string message)

{

bool isCorrectInput = false;

int value = 0;

while (!isCorrectInput)

{

Console.Write(message);

string? temp = Console.ReadLine();

if (!int.TryParse(temp, out value))

Console.WriteLine("Invalid input! Integers only allowed!");

else if (value < 1)

Console.WriteLine("Array size must be > 0!");

else isCorrectInput = true;

}

return value;

}

private static string ReadPath()

{

string? path = "";

while (string.IsNullOrWhiteSpace(path))

{

Console.Write($"Enter filepath{Environment.NewLine}>>");

path = Console.ReadLine();

if (string.IsNullOrWhiteSpace(path))

Console.WriteLine($"{Environment.NewLine}Path must not be empty!{Environment.NewLine}");

}

return path;

}

private static double[] SplitString(string input)

{

string[] separated = input.Split(' ', StringSplitOptions.RemoveEmptyEntries);

int numberOfElements = separated.Length;

double[] values = new double[numberOfElements];

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

if (!double.TryParse(separated[i], out values[i])) return [double.NaN, i + 1];

return values;

}

public static void SaveData(double[] values)

{

bool isCorrect = false;

while (!isCorrect)

{

string path = ReadPath();

if (File.Exists(path))

{

Console.WriteLine($"{Environment.NewLine}File exists! Do you want to overwrite it?{Environment.NewLine}");

Information.AskToOverwriteFile();

if(Console.ReadKey(true).Key == ConsoleKey.D2) continue;

}

else

{

try

{

FileStream fs = File.Open(path, FileMode.OpenOrCreate);

fs.Close();

}

catch(IOException)

{

Console.WriteLine($"{Environment.NewLine}Path contains forbidden values!{Environment.NewLine}");

continue;

}

}

if(values.Length == 3 && double.IsNaN(values[0]))

{

if (values[1] == -1)

File.WriteAllText(path, $"Array does not contain requested element of {values[1]}!");

else

File.WriteAllText(path, $"Requested element of {values[1]} is located at " +

$"position {values[2]} in ascending sorted array");

Information.SaveSuccess("Results");

}

else

{

string data = "";

for(int i = 0; i < values.Length - 1; i++)

{

data += $"{values[i]} ";

}

data += $"{values.Last()}";

File.WriteAllText(path, data);

Information.SaveSuccess("Data");

}

isCorrect = true;

}

}

public static void ConsoleInput(out double[] array)

{

Console.WriteLine($"<<Console input>>{Environment.NewLine}");

Console.WriteLine("Enter number of elements in array:");

int arraySize = ReadInt(">>");

array = new double[arraySize];

Console.WriteLine();

Console.WriteLine($"Enter {arraySize} elements of array:");

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

array[i] = ReadDouble(">>");

Console.WriteLine();

Console.WriteLine($"{Environment.NewLine}Entered array:");

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

Console.Write($"{array[i]} ");

Console.WriteLine();

Information.AskToSave("data");

if(Console.ReadKey(true).Key == ConsoleKey.D1) SaveData(array);

}

public static void RandomInput(out double[] array)

{

Console.WriteLine($"<<Random input>>{Environment.NewLine}");

Console.WriteLine("Enter number of elements in array:");

int arraySize = ReadInt(">>");

array = new double[arraySize];

int randomBorder = 999;

var random = new Random();

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

array[i] = random.Next(-randomBorder, randomBorder);

Console.WriteLine($"{Environment.NewLine}Generated array:");

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

Console.Write($"{array[i]} ");

Console.WriteLine();

Information.AskToSave("data");

if (Console.ReadKey(true).Key == ConsoleKey.D1) SaveData(array);

}

public static void FileInput(out double[] array)

{

Console.WriteLine($"<<File input>>{Environment.NewLine}");

Information.FileInputInfo();

bool isCorrect = false;

array = new double[1];

while(!isCorrect)

{

string path = ReadPath();

if (!File.Exists(path))

{

Console.WriteLine($"{Environment.NewLine}There is no such file!{Environment.NewLine}");

continue;

}

string? input = File.ReadAllText(path);

if (string.IsNullOrWhiteSpace(input))

{

Console.WriteLine($"File is empty!{Environment.NewLine}");

continue;

}

array = SplitString(input);

if(array.Length == 2 && double.IsNaN(array[0]))

{

Console.WriteLine($"File contains invalid value at position {array[1]}!{Environment.NewLine}");

continue;

}

isCorrect = true;

}

Console.WriteLine($"{Environment.NewLine}Entered array:");

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

Console.Write($"{array[i]} ");

Console.WriteLine($"{ Environment.NewLine}");

}

}

}

Файл Information:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1

{

static class Information

{

public static void Greeting()

{

Console.WriteLine("Lab #2");

Console.WriteLine("Group #423");

Console.WriteLine("Efremov Ivan Andreevich");

Console.WriteLine("Option #5");

Console.WriteLine($"Implement binary search of the element in array.{Environment.NewLine}");

}

public static void Options()

{

Console.WriteLine($"[1] - Console input");

Console.WriteLine($"[2] - File input");

Console.WriteLine($"[3] - Random input");

Console.WriteLine($"[4] - Quit application");

Console.WriteLine();

}

public static void AskToSave(string message)

{

Console.WriteLine($"{Environment.NewLine}Do you want to save {message} into file?");

Console.WriteLine("[1] - Yes");

Console.WriteLine("[2] - No");

Console.WriteLine();

}

public static void FileInputInfo()

{

Console.WriteLine("Data in file must be located in one line. Elements must be separated with spaces!");

Console.WriteLine($"Example:{Environment.NewLine}a1 a2 a3 ... an{Environment.NewLine}");

}

public static void AskToOverwriteFile()

{

Console.WriteLine("[1] - Yes");

Console.WriteLine("[2] - No");

Console.WriteLine();

}

public static void SaveSuccess(string message)

{

Console.WriteLine();

Console.WriteLine($"{message} was successfully saved!");

Console.WriteLine();

}

}

}

Файл ArrayManipulation.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

public static class ArrayManipulation

{

public static bool IsArraySorted(double[] array)

{

for(int i = 0; i < array.Length - 1; i++)

{

if (array[i] > array[i + 1]) return false;

}

return true;

}

private static void Swap(ref double[] array, int i, int j)

{

double tmp = array[i];

array[i] = array[j];

array[j] = tmp;

}

public static void InsertionSort(ref double[] array)

{

if (IsArraySorted(array)) return;

for(int i = 1; i < array.Length; i++)

{

double elem = array[i];

int tmpIndex = i - 1;

while(tmpIndex >= 0 && elem < array[tmpIndex])

{

Swap(ref array, tmpIndex + 1, tmpIndex);

tmpIndex--;

}

}

}

public static int BinarySearch(double[] array, double requestedElement)

{

const double epsilon = 0.0001;

int left = 0, right = array.Length - 1;

while(left < right)

{

int mid = left + (right - left) / 2;

if (Math.Abs(array[mid] - requestedElement) < epsilon) return mid;

if (array[mid] < requestedElement) left = mid + 1;

else right = mid - 1;

}

if (Math.Abs(array[left] - requestedElement) < epsilon) return left;

return -1;

}

}

}

Файл Lab2Tests.cs:

using Lab2;

namespace Lab2\_Tests

{

[TestClass]

public class Lab2Tests

{

[TestMethod]

public void TestArraySort()

{

double[] array = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0];

ArrayManipulation.InsertionSort(ref array);

Assert.IsTrue(ArrayManipulation.IsArraySorted(array));

}

[TestMethod]

public void TestSortedArray()

{

double[] array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

Assert.IsTrue(ArrayManipulation.IsArraySorted(array));

}

[TestMethod]

public void ElementIsInArray()

{

double[] array = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0];

ArrayManipulation.InsertionSort(ref array);

int position = ArrayManipulation.BinarySearch(array, 5);

int expected = 5;

Assert.AreEqual(expected, position);

}

[TestMethod]

public void ElementIsNotInArray()

{

double[] array = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0];

ArrayManipulation.InsertionSort(ref array);

int position = ArrayManipulation.BinarySearch(array, 100);

int expected = -1;

Assert.AreEqual(expected, position);

}

}

}