МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Крымский федеральный университет имени В.И. Вернадского»

Таврический колледж

(структурное подразделение)

**ОТЧЕТ ПО ПРАКТИКЕ**

**Учебная практика по профессиональному модулю**

**ПМ.03 Участие в интеграции программных модулей**

Специальность **09.02.03 Программирование в компьютерных системах**

Обучающийся 4 курса группы **4ПКС18**

форма обучения⠀⠀⠀⠀очная⠀⠀⠀⠀

(очная, заочная)

⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀Беловоденко Вениамин Евгеньевич⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀  
(фамилия, имя, отчество)

Место практики

Таврический колледж (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского»

(наименование организации)

Срок практики с **16 марта 2023 г. по 22 марта 2023 г.**

Руководитель практики

от колледжа

преподаватель\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Руденко А.В. /

должность подпись (Ф.И.О.)

Зам директора

по учебно-производственной

практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Малюга Г.Г. /

подпись (Ф.И.О.)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отлично, хорошо, удовлетворительно)

МП

г. Симферополь, 2023 г.

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ 3

ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5

1.1 Разработка технического задания 5

1.2 Разработка спецификаций 6

1.3 Разработка диаграмм 6

ГЛАВА 2. МОДУЛИ ПРОГРАММЫ 11

2.1 Модуль интерфейса 11

2.2 Модуль BubbleSort 11

2.3 Модуль InsertionSorter 12

2.4 Модуль ShakerSorter 13

2.5 Модуль ShellSorter 15

ЗАКЛЮЧЕНИЕ 19

СПИСОК ЛИТЕРАТУРЫ 21

ПРИЛОЖЕНИЕ А 22

**ВВЕДЕНИЕ**

Я, Беловоденко Вениамин Евгеньевич, проходил учебную практику на базе: Таврического колледжа (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского».

Дата начала практики: 16 марта 2023 г.

Дата окончания практики: 22 марта 2023 г.

Дата сдачи отчёта по практике: 22 марта 2023 г.

Цель практики:

Формирование и развитие общих и профессиональных компетенций по модулю ПМ.03 Участие в интеграции программных модулей.

Задачи учебной практики:

Закрепление навыков разработки программного обеспечения;

Использование методов для получения кода с заданной функциональностью и степенью качества;

Разработка документации на программный продукт;

Знание моделей процесса разработки программного обеспечения, основных принципов процесса разработки программного обеспечения;

Знание основных подходов к интегрированию программных модулей, основных методов и средств эффективной разработки ПО;

Задание для выполнения:

1. Необходимо разработать программный комплекс по демонстрации работы алгоритмов сортировки массивов данных (реализовать не менее 4 алгоритмов сортировки, которые выбрать самостоятельно):
   1. Разработать техническое задание на программный продукт.
   2. Разработать спецификацию на программный продукт.
   3. Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.
   4. Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.
   5. Разработать коды программных модулей программного продукта.
   6. Разработать пользовательский интерфейса программного продукта в визуальной среде.
   7. Выполнить интеграцию программных модулей в программный продукт.
   8. Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.
   9. Разработать справочную систему программного продукта.
   10. Разработать руководства оператора (пользователя).
2. Создать аккаунт в GitHub. Создать папку проекта. В папку загрузить разработанный программный комплекс, всю разработанную документацию к проекту (п.п.1.1 – 1.10).
3. Составить отчет о выполнении.

**ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

1.1 Разработка технического задания

Наименование программы: "Belovodenko\_Program".

Программа предназначена для сортировки массивов различных размеров 4-мя разными методами сортировки.

Программа предоставляет свой пользовательский интерфейс.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.   
На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

Этапы разработки:  
1. Разработка программы

2. Разработка программной документации.

3. Испытание программы.

На стадии внедрения должен быть выполнен этап разработки подготовка и передача программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:   
1. постановка задачи;   
2. определение и уточнение требований к техническим средствам;   
3. определение требований к программе;  
4. определение стадий, этапов и сроков разработки программы и документации на неё;   
5. согласование и утверждение технического задания.   
На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.   
На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.   
На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:   
1. разработка, согласование и утверждение и методики испытаний;   
2. проведение приемо-сдаточных испытаний;   
3. корректировка программы и программной документации по результатам испытаний.   
На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

1.2 Разработка спецификаций

Программа “Belovodenko\_Program” реализована на языке программирования C#, предназначена для выполнения задач, связанных с сортировкой различного размера массивов разными методами, среди них: метод пузырьком, метод выборки, метод перестановки, и метод вставки, в программе представлены следующие модули: Модуль интерфейса, модуль BubbleSort, модуль InsertionSorter, модуль ShakerSorter модуль ShellSorter.

Предназначена для людей, у которых стоит потребность в сортировке массивов различных размеров.

1.3 Разработка диаграмм

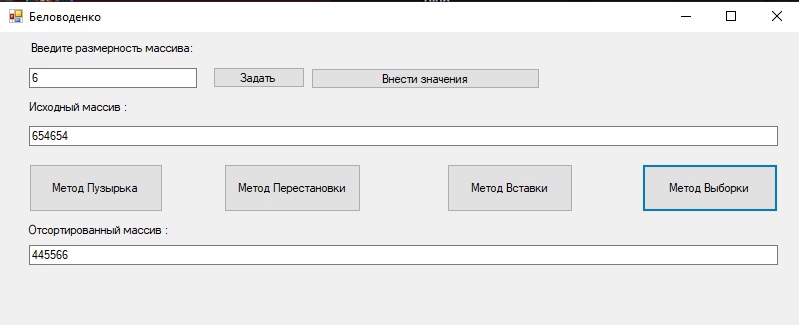


Рисунок 1 – Первая UML диаграмма 1

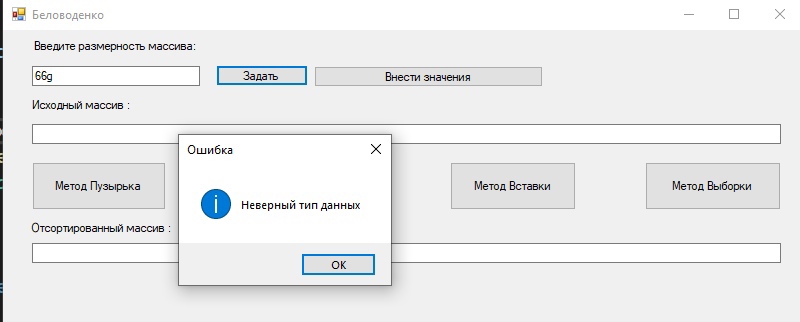
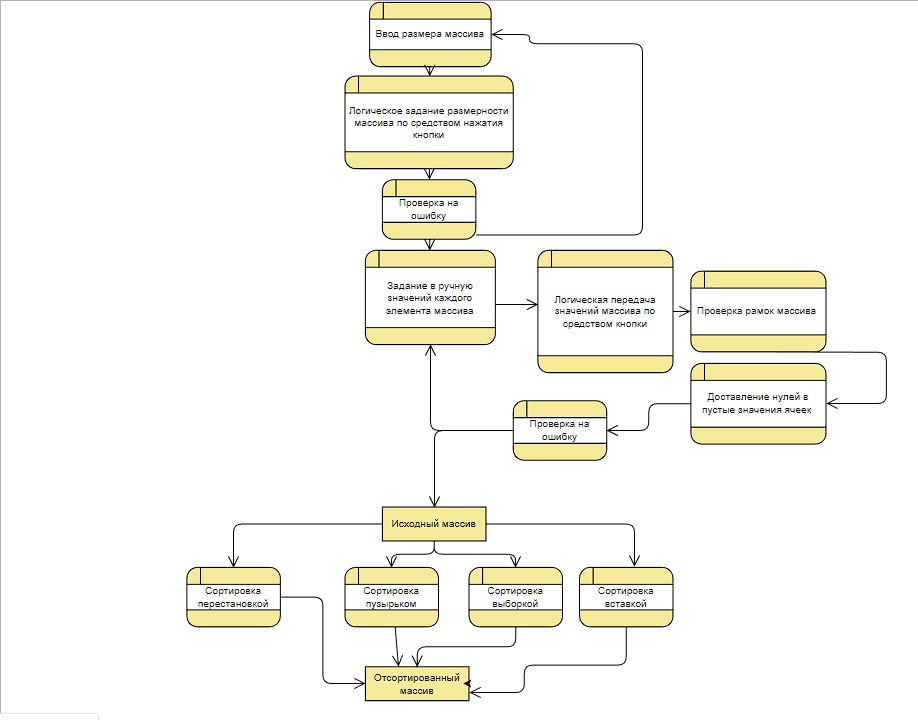
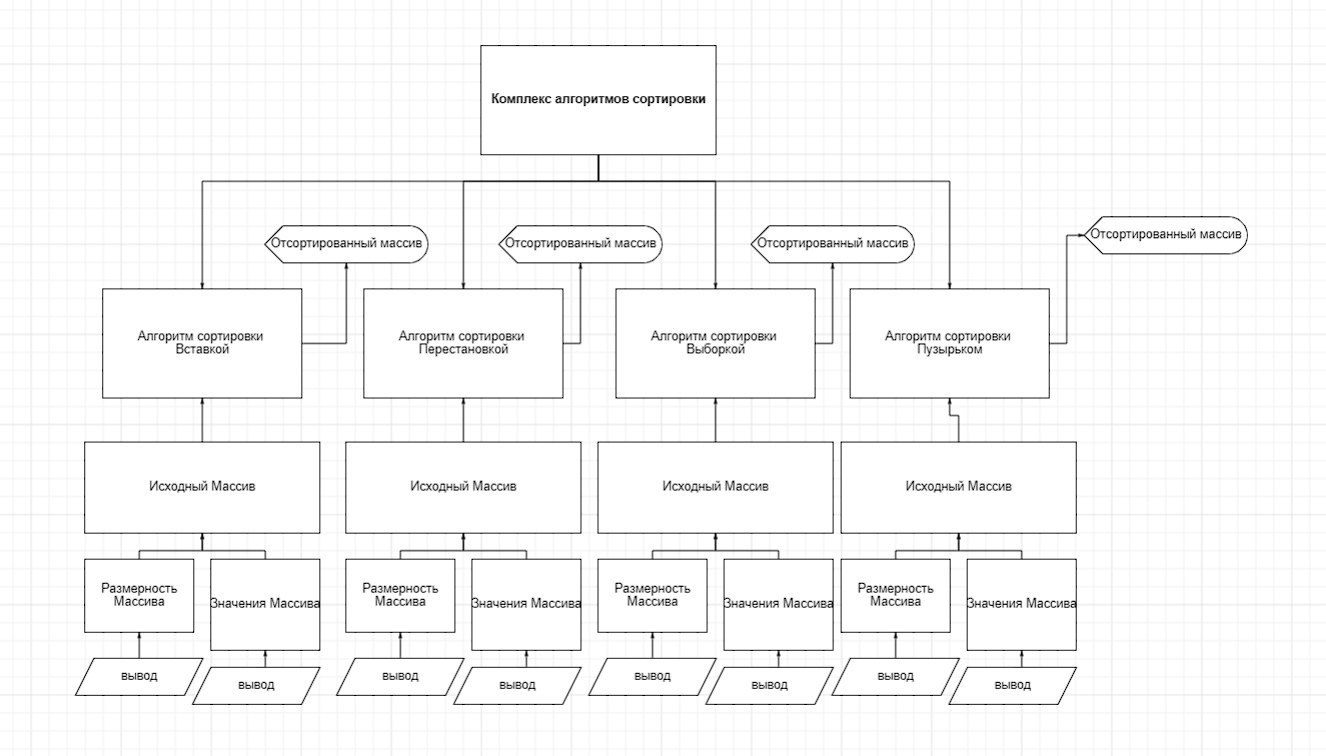
На данном рисунке представлена первая UML диаграмма.

Рисунок 2 – Вторая UML диаграмма

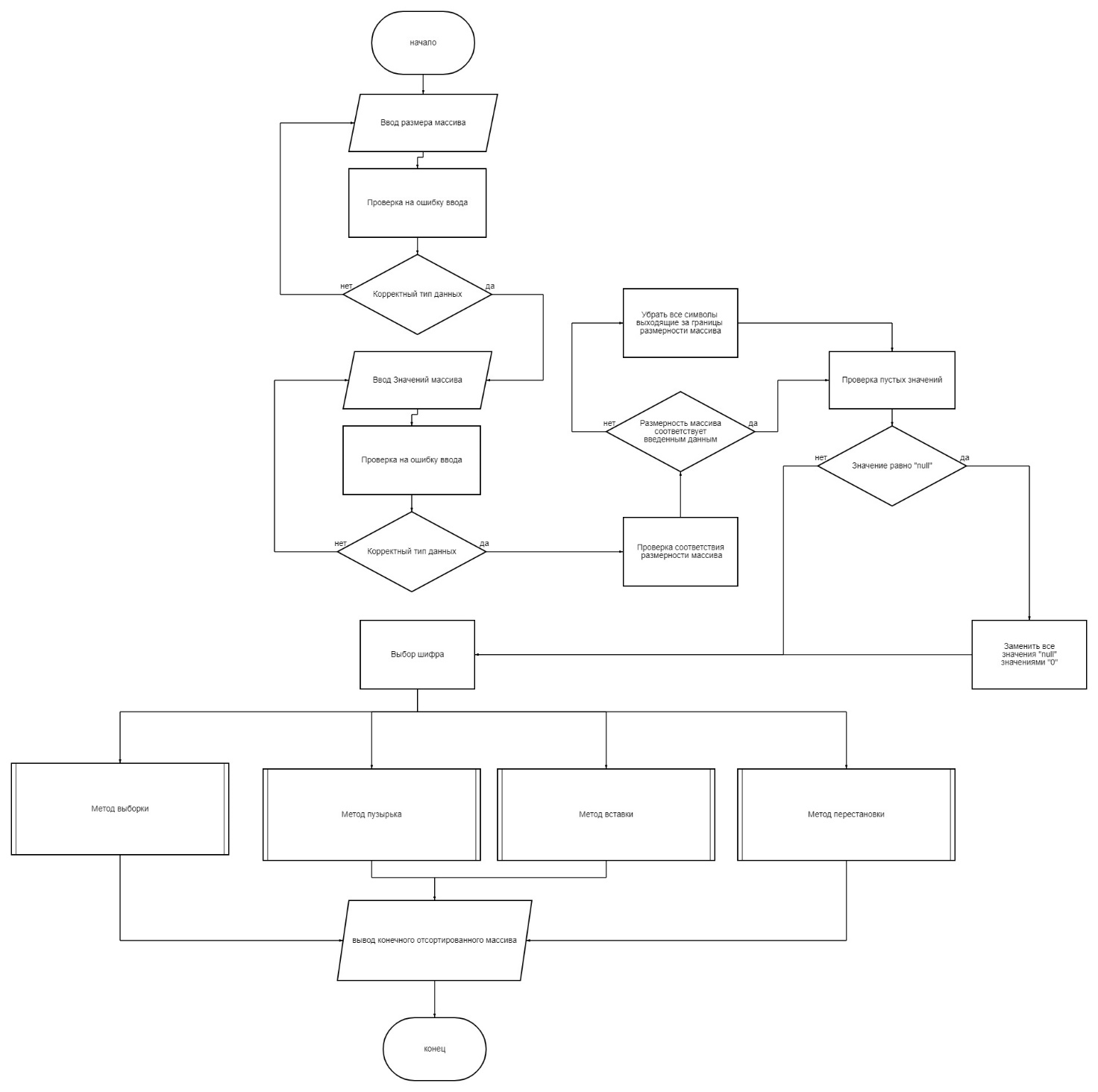
На данном рисунке представлена вторая UML диаграмма.

Рисунок 3 – Диаграмма потока данных

На данном рисунке представлена диаграмма потока данных.

Рисунок 4 – Функциональная схема продукта

На данном рисунке представлена функциональная схема продукта.

Рисунок 5 – Блок схема программы

На данном рисунке представлена блок схема реализованной программы.

**ГЛАВА 2. МОДУЛИ ПРОГРАММЫ**

2.1 Модуль интерфейса

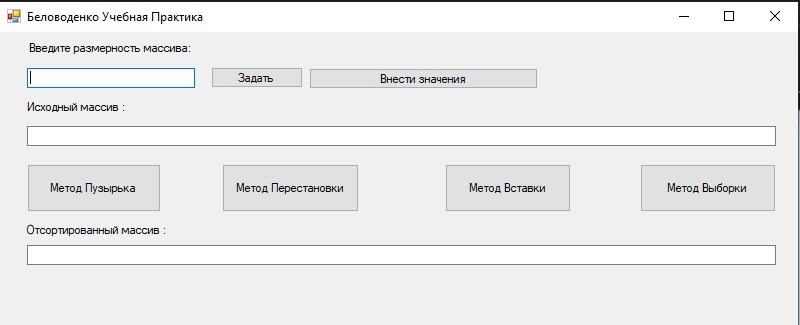
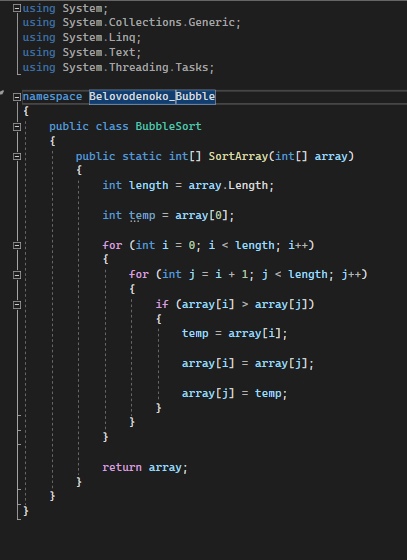


Рисунок 6 – Модуль интерфейса

На данном рисунке представлен общий интерфейс программы, в нем реализованы такие кнопки как: “Задать”, ”Внести значения”, ”Метод Пузырька”, ”Метод Перестановки”, ”Метод Вставки”, “Метод Выборки”, а также окна: “Введите размерность массива::”, “Исходный массив:”, “Отсортированный массив:”.

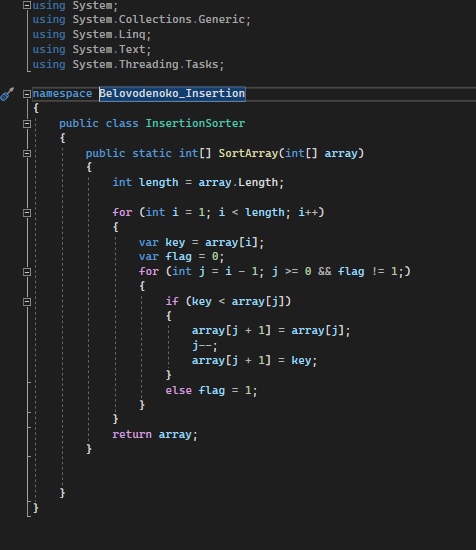
2.2 Модуль BubbleSort

Рисунок 7 – Модуль BubbleSort

На данном рисунке представлена реализация метода сортировки пузырьком, весь необходимый для реализации внутри программы код.

Пузырьковая сортировка, иногда называемая потопающей сортировкой, представляет собой простой алгоритм сортировки, который многократно перебирает входной список элемент за элементом, сравнивая текущий элемент с последующим, при необходимости меняя местами их значения. Эти проходы по списку повторяются до тех пор, пока во время прохода не потребуется выполнять замену, что означает, что список стал полностью отсортированным. [1]

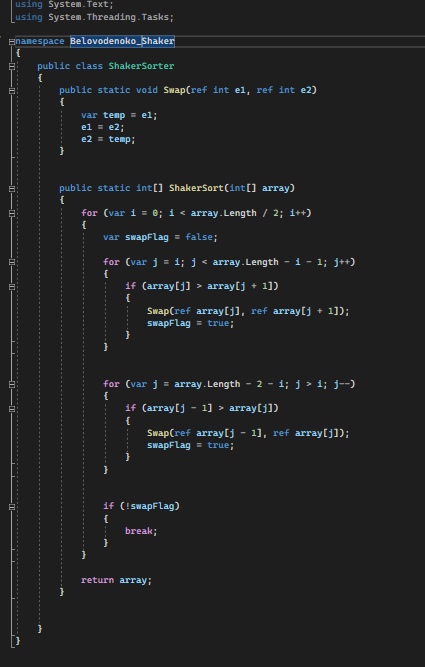
2.3 Модуль InsertionSorter

Рисунок 8 – Модуль InsertionSorter

На данном рисунке представлена реализация метода сортировки вставками, весь необходимый для реализации внутри программы код.

Сортировка вставками (Insertion sort) – простой алгоритм сортировки, при которой очередной элемент добавляется всегда в конец списка, а затем перемещается в начало списка до тех пор, пока он меньше предыдущего элемента. Таким образом, формально, находится правильное место для вставки (например, сортировка карточек в библиотеке). [2]

2.4 Модуль ShakerSorter

Рисунок 9 – Модуль ShakerSort

На данном рисунке представлена реализация метода сортировки перемешиванием, весь необходимый для реализации внутри программы код.

Термин Shakersort описывает стабильный алгоритм сортировки, который сортирует набор линейно расположенных элементов (например, чисел) по размеру. Другие названия этого алгоритма - Cocktailsort, Ripplesort, Shearsort или BiDiBubbleSort (двунаправленная пузырьковая сортировка). [3]

2.5 Модуль ShellSorter

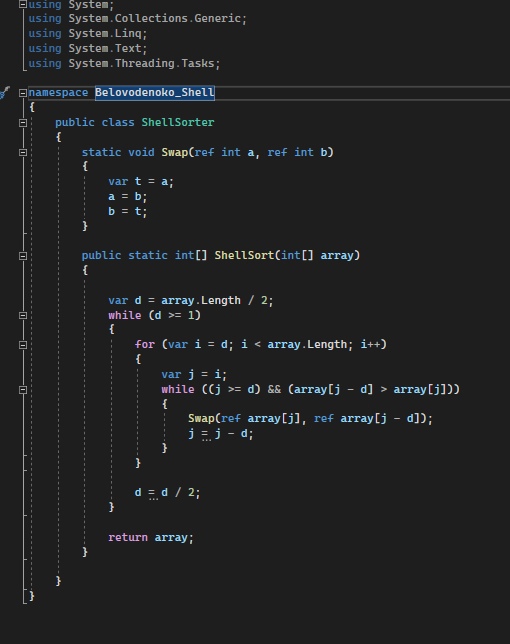


Рисунок 10 – Модуль ShellSorter

На данном рисунке представлена реализация метода сортировки Шелла, весь необходимый для реализации внутри программы код.

В 1959 году американский ученый Дональд Шелл опубликовал алгоритм сортировки, который впоследствии получил его имя – «Сортировка Шелла». Этот алгоритм может рассматриваться и как обобщение пузырьковой сортировки, так и сортировки вставками. Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или N/2, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии N/2; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на d/2, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на d=1 проход по массиву происходит в последний раз сортировка данных в массиве является одной из наиболее распространенных задач в информатике. Но повышение эффективности алгоритмов сортировки очень важно не только по этой причине. Неослабевающий интерес к их оптимизации объясняется тем, что объемы информационных массивов непрерывно и стремительно растут, соответственно возрастают и требования к скорости сортировки. Нами была поставлена цель проследить эволюцию алгоритмов сортировки данных с первых методов, используемых при машинной обработке информации, да настоящего времени, выделив основные этапы и направления их развития. Начать хотелось бы с того, что важное практическое значение проблема сортировки данных в больших массивах впервые приобрела в США в середине XIX века. В 1840 году там был создан центральный офис переписи населения, куда стекались первичные данные из всех штатов. В ходе переписи было опрошено 17 069 453 человек, каждая анкета состояла из 13 вопросов. Объем полученных данных был столь велик, что их обработка традиционным ручным способом потребовала непомерных затрат труда и времени. Ситуация усугублялось необходимостью проведения постоянных сверок и пересчетов из-за допускаемых при ручной сортировке данных ошибок. С каждой новой переписью, которая проводилась раз в десять лет, объем обрабатываемой информации, а вместе с ним стоимость и длительность обработки данных возрастали. Так, ручная обработка данных переписи населения 1880 года (50 189 209 человек) потребовала привлечения сотен служащих и длилась семь с половиной лет [1]. Перед переписью 1890 года для решения проблемы сортировки данных в очень больших массивах информации по инициативе бюро переписи был проведен конкурс на лучшее электромеханическое сортировочное оборудование, которое сделало бы сортировку данных более эффективной — более быстрой, точной и дешевой. Конкурс выиграл американский инженер и изобретатель немецкого происхождения Герман Холлерит (Herman Hollerith), разработавший оборудование для работы с перфокартами — электрическую табулирующую систему, ставшую известной как Hollerith Electric Tabulating System. Использование этого оборудования при переписях населения США в 1890 и 1900 годах было признано очень успешным. Вот как были описаны преимущества машины Холлерита в русском журнале «Вестник Опытной Физики и Элементарной Математики» в 1895 году: «Преимущества машины Голлерита заключаются: а) в значительном ускорении и удешевлении работы. При ручном способе можно разложить и подсчитать за час не более 400 карточек. Если принять, что в Российской Империи 120 миллионов жителей, то для изготовления одной только сводной таблицы потребуется не менее … 300 000 часов… Машина сокращает работу почти в 5 раз. б) в большей точности результатов… в) в большей легкости получения сложных сводных таблиц… После немногих пропусках через машину всех счетных карточек получаются столь полные и разнообразные таблицы, составление которых было почти немыслимо при прежнем способе». [2, с. 200–201]. Использование способа табулирования при сортировке данных оказалось настолько эффективным, что предварительные подсчеты результатов переписи потребовали всего шесть недель, а полный статистический анализ данных занял два с половиной года, Таким образом, обработка данных с помощью машины Холлерита потребовала в три раза меньше времени, чем вручную, причем точность сводных таблиц значительно возросла. Таким образом, в конце XIX века на смену ручной сортировке данных в массивах пришла сортировка с помощью статистических табуляторов. При этом использовался алгоритм поразрядной сортировки. Следующий этап развития способов и алгоритмов сортировки начался в начале 1940-х годов с появлением первых электронных вычислительных машин. Фантастическое по тем временам быстродействие ЭВМ вызвало рост интереса к новым, приспособленным для машинной обработки алгоритмам сортировки. В 1946 году вышла первая статья об алгоритмах сортировки данных, автором которой был Джон Уильям Мочли (John William Mauchly) — американский физик и инженер, один из создателей первого в мире электронного цифрового компьютера общего назначения ENIAC. В статье рассматривался целый ряд новых алгоритмов сортировки, в том числе метод бинарных вставок. До середины 1950-х годов наиболее распространенными были модификации сортировки слиянием и вставками сложности O (n log n) для п элементов. Еще одним следствием перехода к сортировке данных с помощью ЭВМ стало разделение сортировки на два типа — внешнюю и внутреннюю, то есть на использующую и не использующую данные, расположенные на периферийных устройствах. В середине 1950-х годов с разработкой ЭВМ второго поколения началось активное развитие алгоритмов сортировки. Основными предпосылками для этого стали, во-первых, значительное упрощение и ускорение написания программ для компьютеров в результате разработки первых языков программирования высокого уровня (Фортран, Алгол, Кобол); во-вторых, значительное повышение доступности компьютеров в результате резкого уменьшения их габаритов и стоимости и, как следствие, достаточно широкое их распространение; в-третьих, увеличение производительности компьютеров до 30 тысяч операций в секунду.

**ЗАКЛЮЧЕНИЕ**

Я, Беловоденко Вениамин Евгеньевич, проходил учебную практику на базе: Таврического колледжа (структурное подразделение) ФГАОУ «КФУ им. В.И. Вернадского».

Дата начала практики: 16 марта 2023 г.

Дата окончания практики: 22 марта 2023 г.

Дата сдачи отчёта по практике: 22 марта 2023 г.

Была выполнена следующая цель практики:

Формирование и развитие общих и профессиональных компетенций по модулю ПМ.03 Участие в интеграции программных модулей.

Были выполнены следующие задачи учебной практики:

Закрепление навыков разработки программного обеспечения;

Использование методов для получения кода с заданной функциональностью и степенью качества;

Разработка документации на программный продукт;

Знание моделей процесса разработки программного обеспечения, основных принципов процесса разработки программного обеспечения;

Знание основных подходов к интегрированию программных модулей, основных методов и средств эффективной разработки ПО;

Был выполнен весь план задания для выполнения:

1. Необходимо разработать программный комплекс по демонстрации работы алгоритмов сортировки массивов данных (реализовать не менее 4 алгоритмов сортировки, которые выбрать самостоятельно):
   1. Разработать техническое задание на программный продукт.
   2. Разработать спецификацию на программный продукт.
   3. Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.
   4. Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.
   5. Разработать коды программных модулей программного продукта.
   6. Разработать пользовательский интерфейса программного продукта в визуальной среде.
   7. Выполнить интеграцию программных модулей в программный продукт.
   8. Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.
   9. Разработать справочную систему программного продукта.
   10. Разработать руководства оператора (пользователя).
2. Создать аккаунт в GitHub. Создать папку проекта. В папку загрузить разработанный программный комплекс, всю разработанную документацию к проекту (п.п.1.1 – 1.10).
3. Составить отчет о выполнении.

Ссылка на репозиторий с выполненной работой: [https://github.com/DeXose/UP](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2FDeXose%2FUP&cc_key=)

**СПИСОК ЛИТЕРАТУРЫ**

1. Туполев А.В. Алгоритмы сортировки [электронный ресурс] studfile.net – URL: [https://studfile.net/preview/16573799/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fstudfile.net%2Fpreview%2F16573799%2F). – (дата обращения – 20.03.2023)

2. Евкова А.П. Методы сортировки данных: эволюция и сравнительный анализ. Примеры использования. [электронный ресурс] www.evkova.org – URL: https://www.evkova.org/kursovye-raboty/metodyi-sortirovki-dannyih-evolyutsiya-i-sravnitelnyij-analiz-primeryi-ispolzovaniya. – (дата обращения – 20.03.2023)

3. Павлов М.Ю. Алгоритмы и структуры данных [электронный ресурс] [www.aisd.kubsau.ru](http://www.aisd.kubsau.ru) – URL: <https://www.aisd.kubsau.ru/lections/lect_kurs_bi/lect9_bi.html>. – (дата обращения – 20.03.2023)

4. Швец А.Н. Идеи реализации. Глава 18. Сортировка [электронный ресурс] mech.math.msu.su – URL: <http://mech.math.msu.su/~shvetz/54/inf/perl-examples/PerlExamples_Sorting_Ideas.xhtml>. – (дата обращения – 20.03.2023)

5. Логинов М.И. Сортировки [электронный ресурс] cs.mipt.ru – URL: http://cs.mipt.ru/algo/lessons/lab8.html. – (дата обращения – 20.03.2023)

**ПРИЛОЖЕНИЕ А**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Belovodenoko\_Bubble;

using Belovodenoko\_Shaker;

using Belovodenoko\_Insertion;

using Belovodenoko\_Shell;

namespace ArraySorter

{

public partial class Form1 : Form

{

int n = 0;

public Form1()

{

InitializeComponent();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

int myArr = int.Parse(textBox4.Text);

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

string texter = string.Join("",BubbleSort.SortArray(result));

textBox3.Text = texter;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

textBox4.Text = "";

try

{

n = Convert.ToInt32(textBox1.Text);

}

catch

{

MessageBox.Show(

"Неверный тип данных",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1);

}

int[] x = new int[n];

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

{

textBox4.Text = textBox4.Text + x[i];

}

}

private void button2\_Click(object sender, EventArgs e)

{

int myArr = int.Parse(textBox4.Text);

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

string texter = string.Join("", ShakerSorter.ShakerSort(result));

textBox3.Text = texter;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

int[] x = new int[n];

if (x.Length > textBox4.TextLength)

{

int a = x.Length - textBox4.TextLength;

while (a > 0)

{

a--;

textBox4.Text = textBox4.Text + 0;

}

}

else

if (x.Length < textBox4.TextLength)

{

int a = textBox4.TextLength - x.Length;

textBox4.Text = textBox4.Text.Remove(textBox4.Text.Length - a);

a=0;

}

}

private void button3\_Click(object sender, EventArgs e)

{

int myArr = int.Parse(textBox4.Text);

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

int resl = result.Length;

string texter = string.Join("", InsertionSorter.SortArray(result));

textBox3.Text = texter;

}

private void button4\_Click(object sender, EventArgs e)

{

int myArr = int.Parse(textBox4.Text);

int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) - 48).ToArray();

int resl = result.Length;

string texter = string.Join("", ShellSorter.ShellSort(result));

textBox3.Text = texter;

}

}

}