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)

Тейфуков Руслан Наримановович
(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

Я, Тейфуков Руслан Наримановичб 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):

1.1 Разработать техническое задание на программный продукт.

1.2 Разработать спецификацию на программный продукт.

1.3 Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.

1.4 Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.

1.5 Разработать коды программных модулей программного продукта.

1.6 Разработать пользовательский интерфейса программного продукта в визуальной среде.

1.7 Выполнить интеграцию программных модулей в программный продукт.

1.8 Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.

1.9 Разработать справочную систему программного продукта.

1.10 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: "ArraySorting".

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

Программный комплекс предоставляет пользовательский интерфейс, модуль сортировки пузырьков, модуль быстрой сортировки, модуль сортировки вставкой, модуль гномьей сортировки.

Программный комплекс реализован на языке программирования C#, в визуальной среде разработки Visual Studio Code, для операционной системы Windows 10.

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:

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.

Этапы разработки:

1. Разработка программы
2. Разработка программной документации.
3. Испытание программы.

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

1.2 Разработка спецификаций

Программа “ArraySorting” реализована на языке программирования C#, предназначена для выполнения задач, связанных с сортировкой различного размера массивов разными методами, среди них: метод пузырьком, сортировка вставками, гномья сортировка, и быстрая сортировка, в программе представлены следующие модули: Модуль интерфейса, модуль BubbleSorting, модуль MergeSorting, модуль QuickSorting, модуль SelectionSorting. Предназначена для людей, у которых стоит потребность в сортировке массивов различных размеров.

1.3 Разработка диаграмм

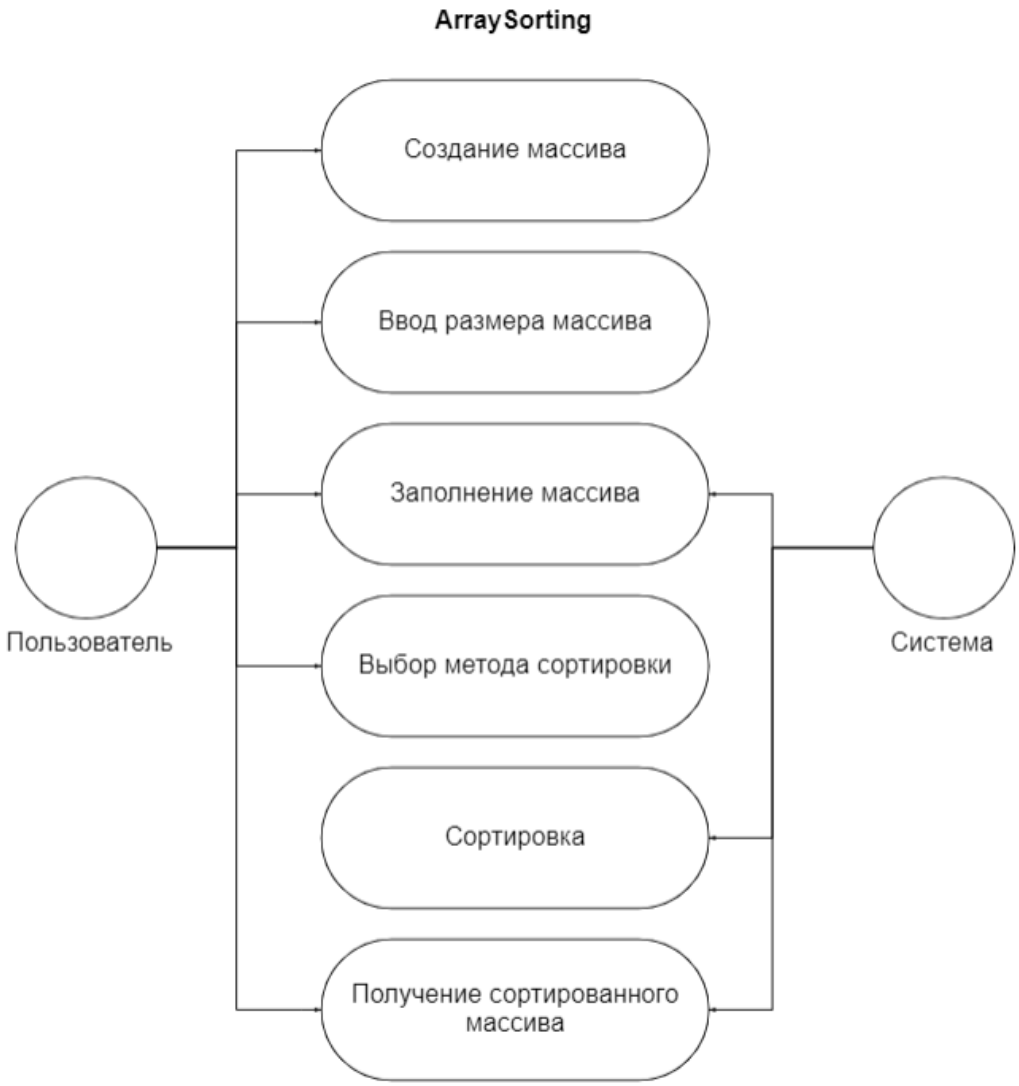


Рисунок 1 – UML диаграмма

На данном рисунке представлена UML диаграмма программного комплекса. Субъекты диаграммы – пользователь и система. Пользователь задает размер массива и его наполнение, а также выбирает метод сортировки. Система считывает массив и сортирует его с помощью выбранного метода, после чего выводит пользователю.

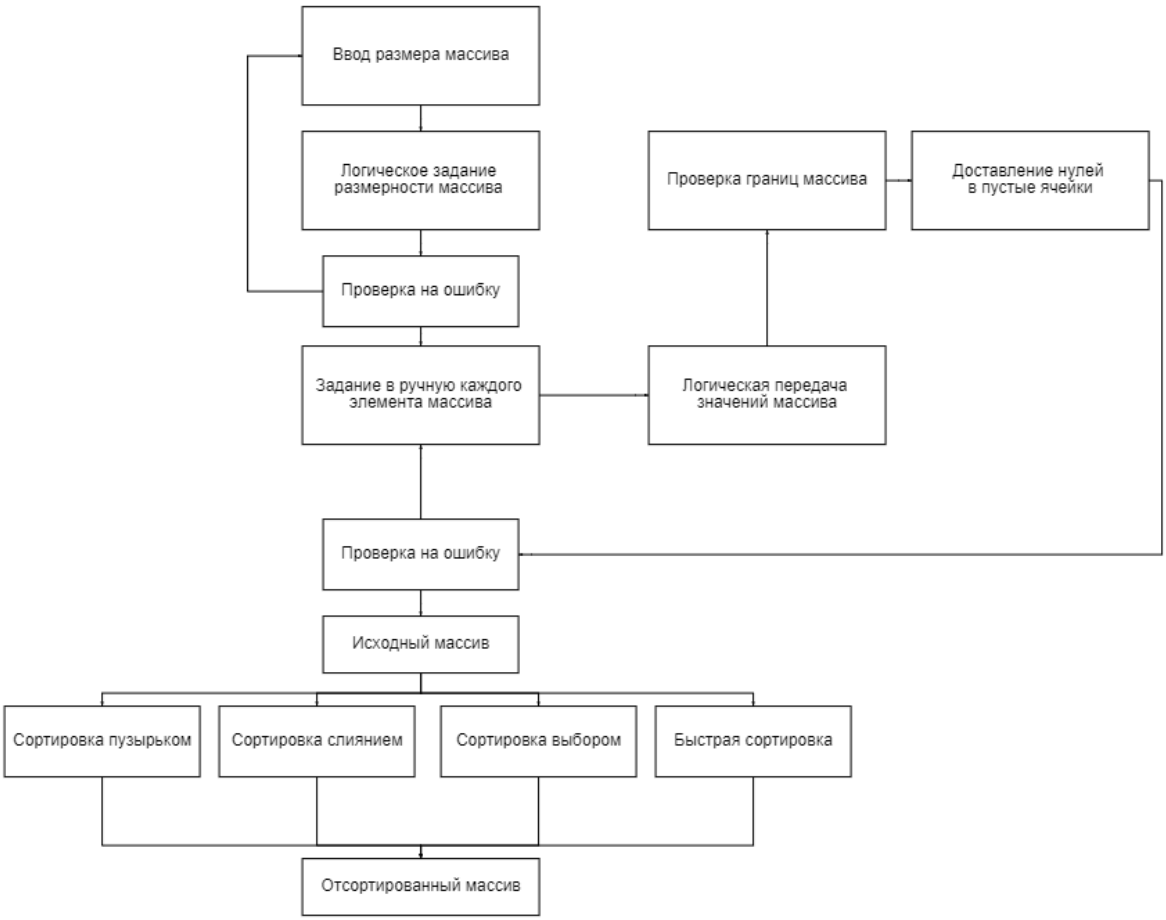


Рисунок 2 – Диаграмма потока данных

На данном рисунке представлена диаграмма потока данных программного комплекса ArraySorting.

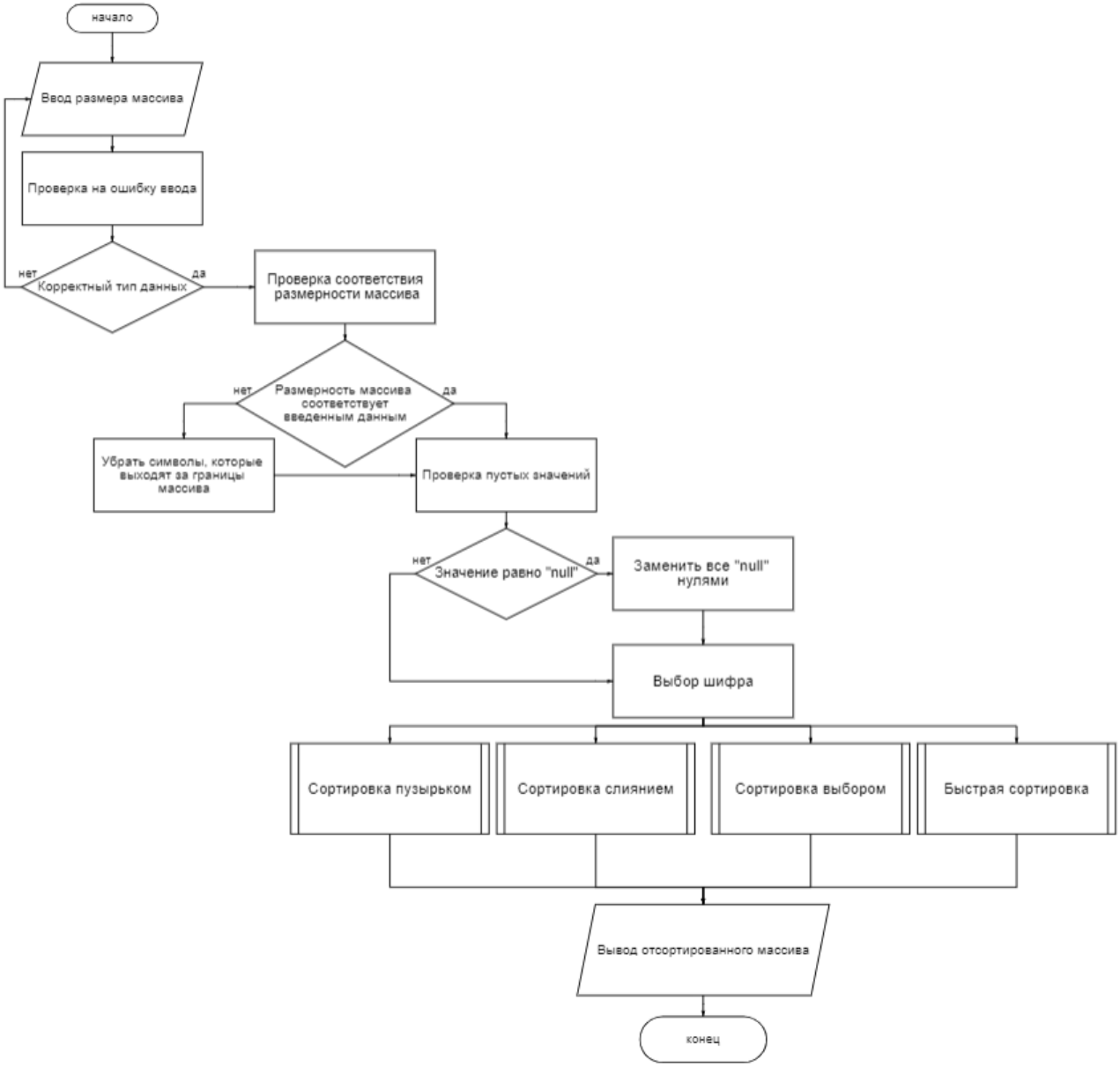


Рисунок 3 – Блок-схема программы

На данном рисунке представлена блок-схема программы.

ГЛАВА 2. МОДУЛИ ПРОГРАММЫ

2.1 Модуль интерфейса

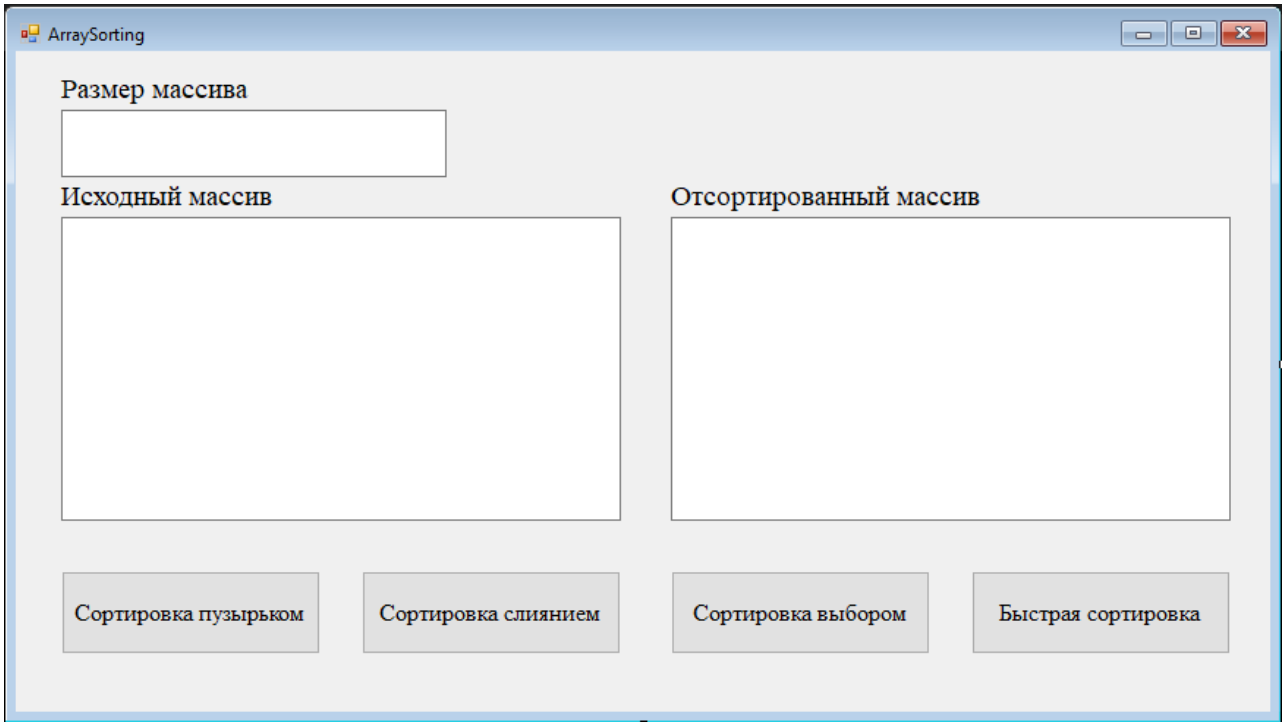
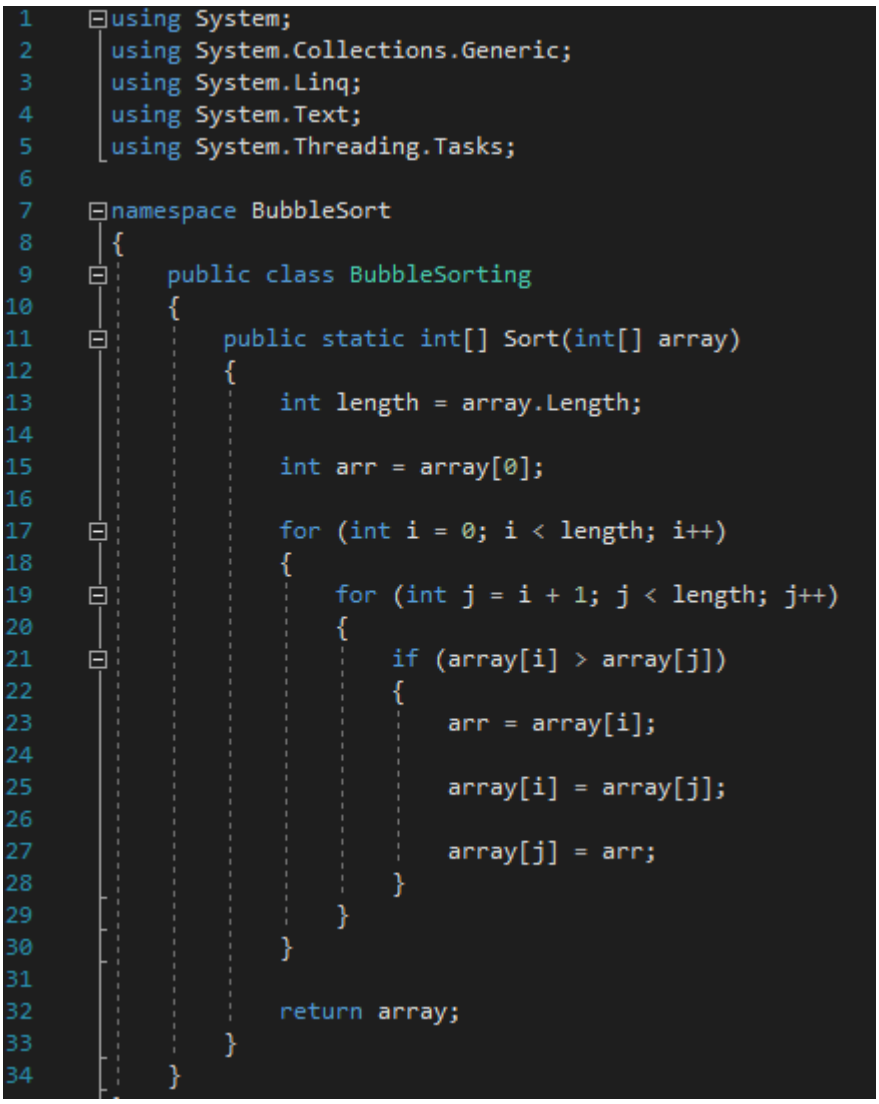


Рисунок 4 – Модуль интерфейса

На данном рисунке представлен общий интерфейс программы, в нем реализованы такие кнопки как: “Сортировка пузырьком”, “Сортировка слиянием”, “Сортировка выбором”, “Быстрая сортировка”, а также окна: “Размер Массива:”, “Исходный массив:”, “Отсортированный массив:”.

2.2 Модуль BubbleSorting



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BubbleSort
{
    public class BubbleSorting
    {
        public static int[] Sort(int[] array)
        {
            int length = array.Length;

            int arr = array[0];

            for (int i = 0; i < length; i++)
            {
                for (int j = i + 1; j < length; j++)
                {
                    if (array[i] > array[j])
                    {
                        arr = array[i];

                        array[i] = array[j];

                        array[j] = arr;
                    }
                }
            }

            return array;
        }
    }
```

Рисунок 5 – Модуль BubbleSorting

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

Сортировка пузырьком — простой алгоритм сортировки. Для понимания и реализации этот алгоритм — простейший, но эффективен он лишь для небольших массивов.

Алгоритм считается учебным и практически не применяется вне учебной литературы, вместо него на практике применяются более эффективные алгоритмы сортировки.

Алгоритм состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более $n-1$ проходов, где n размер массива, чтобы отсортировать массив.

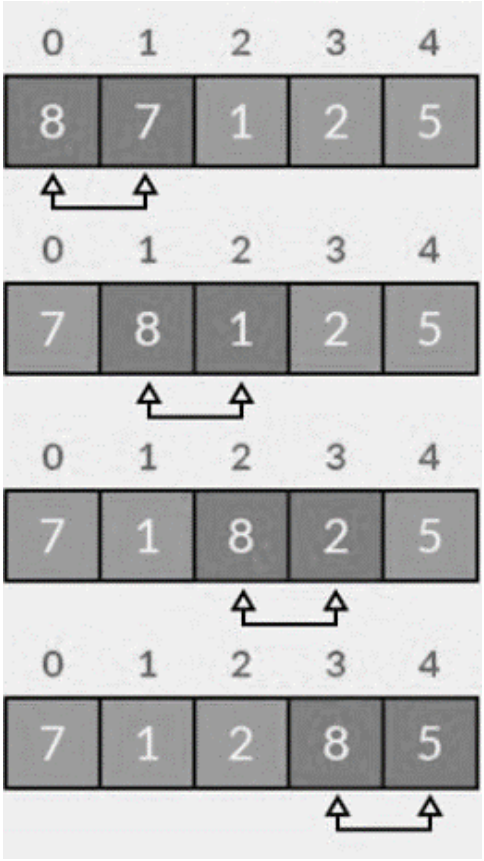
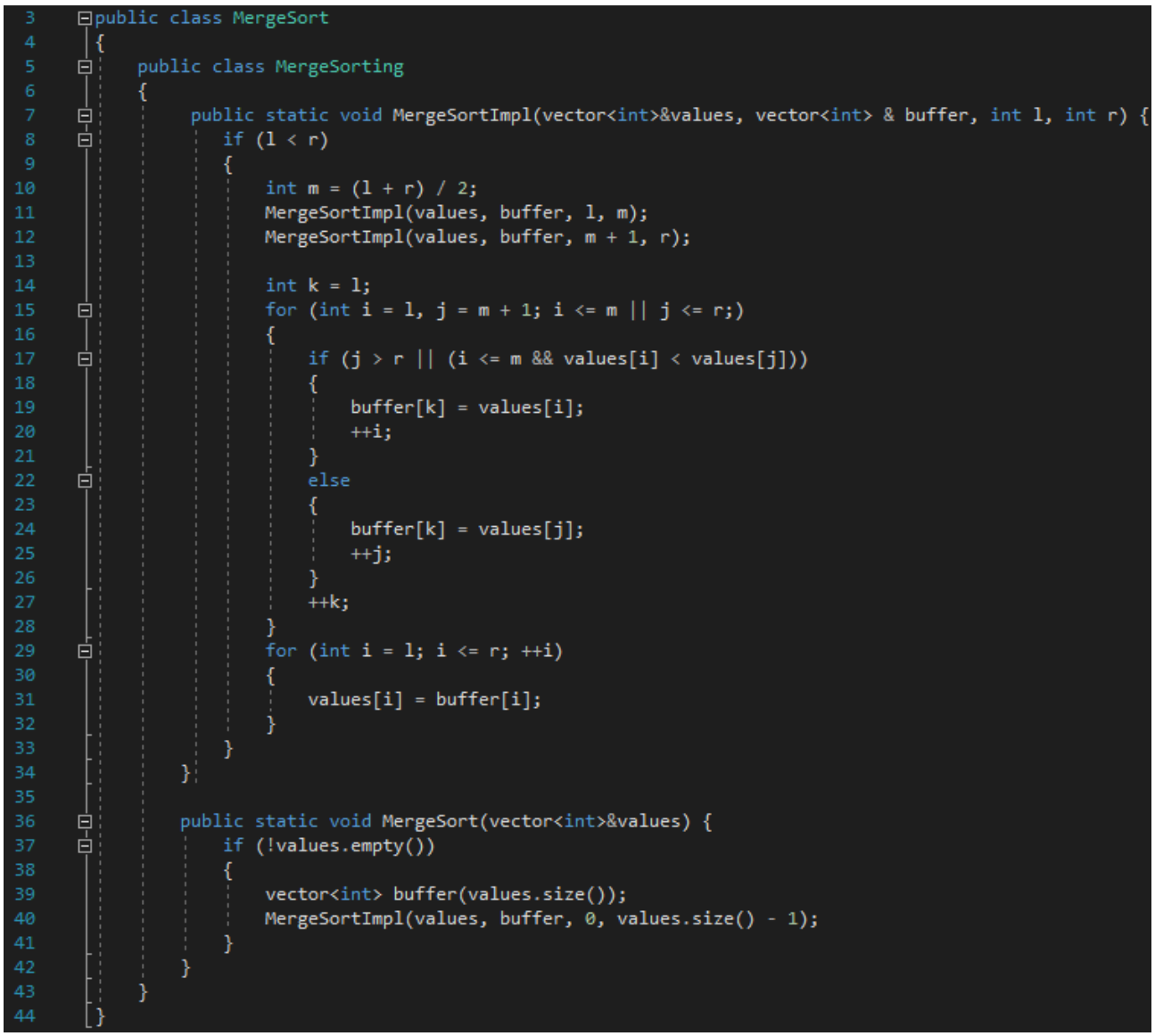


Рисунок 6 – Пример реализации сортировки пузырьком

2.3 Модуль MergeSorting



```
public class MergeSort
{
    public class MergeSorting
    {
        public static void MergeSortImpl(vector<int>&values, vector<int> & buffer, int l, int r) {
            if (l < r)
            {
                int m = (l + r) / 2;
                MergeSortImpl(values, buffer, l, m);
                MergeSortImpl(values, buffer, m + 1, r);

                int k = l;
                for (int i = l, j = m + 1; i <= m || j <= r;)
                {
                    if (j > r || (i <= m && values[i] < values[j]))
                    {
                        buffer[k] = values[i];
                        ++i;
                    }
                    else
                    {
                        buffer[k] = values[j];
                        ++j;
                    }
                    ++k;
                }
                for (int i = l; i <= r; ++i)
                {
                    values[i] = buffer[i];
                }
            }
        }

        public static void MergeSort(vector<int>&values) {
            if (!values.empty())
            {
                vector<int> buffer(values.size());
                MergeSortImpl(values, buffer, 0, values.size() - 1);
            }
        }
    }
}
```

Рисунок 7 – Модуль MergeSorting

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

Сортировка слиянием пригодится для таких структур данных, в которых доступ к элементам осуществляется последовательно (например, для потоков). Главное преимущество сортировки слиянием — она работает всегда с одной и той же скоростью на любых массивах. [2]

Конкретно процедуру сортировки слиянием можно описать следующим образом:

1. Если в рассматриваемом массиве один элемент, то он уже отсортирован — алгоритм завершает работу.

2. Иначе массив разбивается на две части, которые сортируются рекурсивно.

3. После сортировки двух частей массива к ним применяется процедура слияния, которая по двум отсортированным частям получает исходный отсортированный массив.

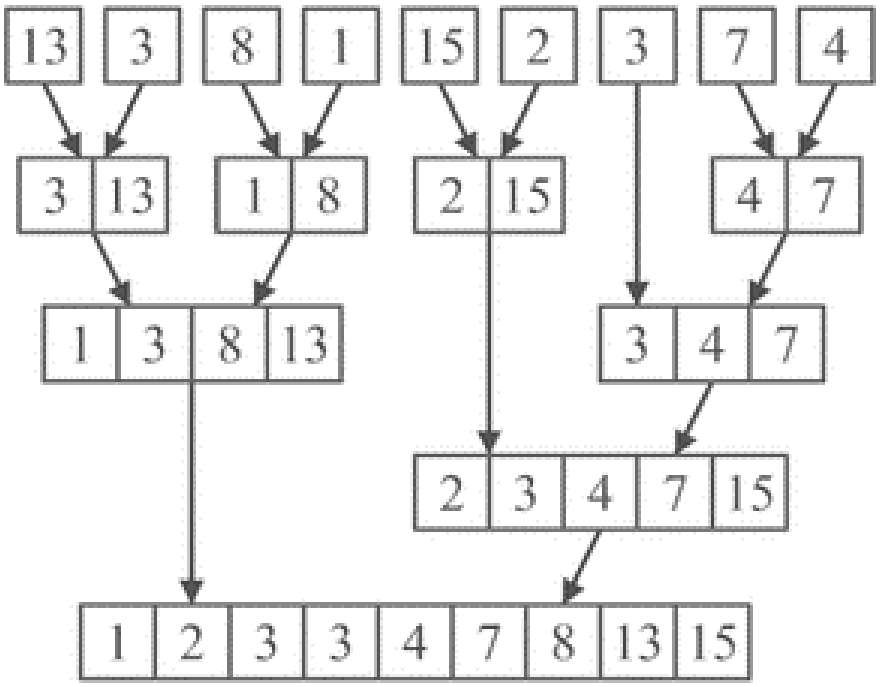
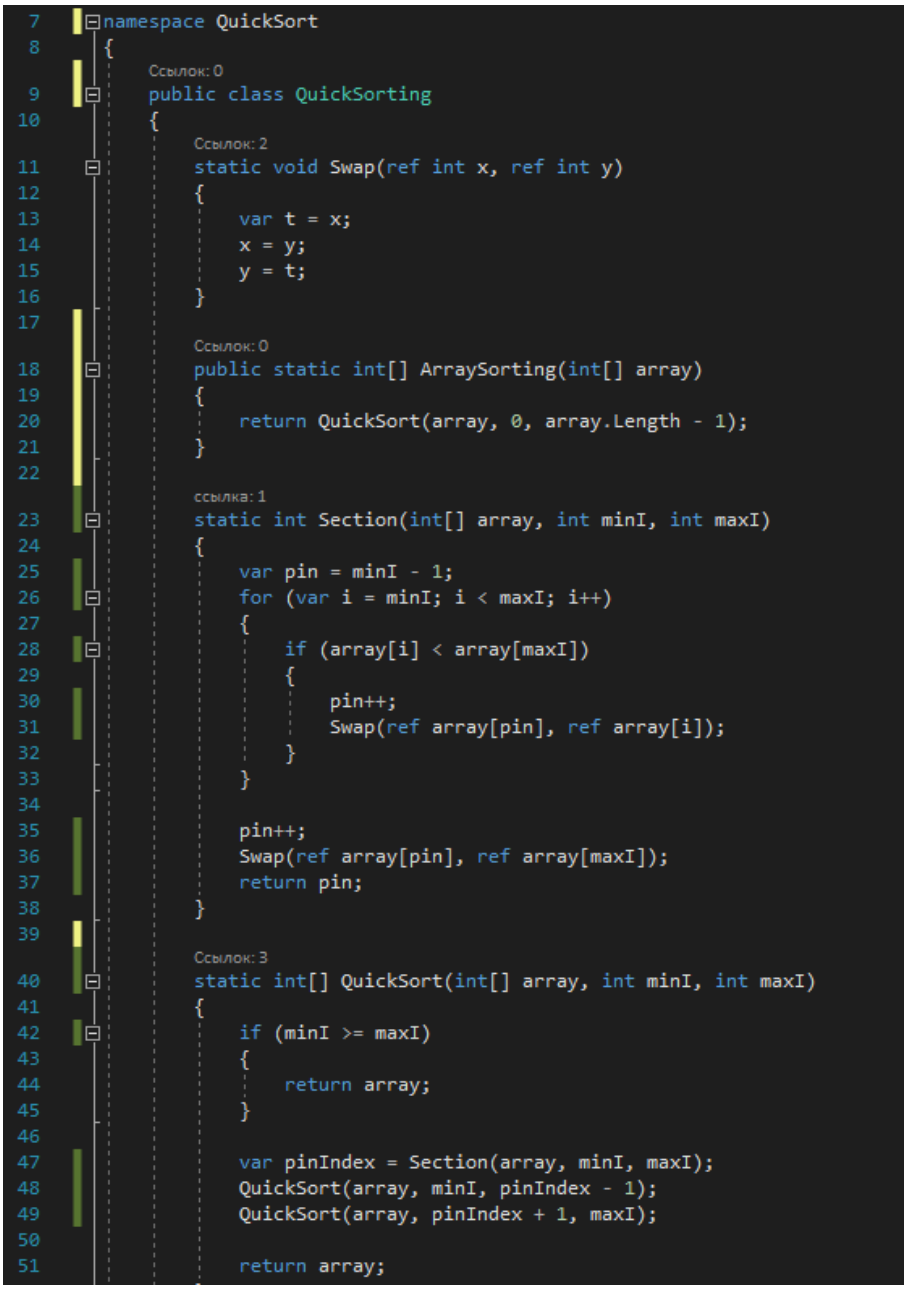


Рисунок 8 – Пример реализации сортировки слиянием

2.4 Модуль QuickSorting



```
namespace QuickSort
{
    Ссылок: 0
    public class QuickSorting
    {
        Ссылок: 2
        static void Swap(ref int x, ref int y)
        {
            var t = x;
            x = y;
            y = t;
        }

        Ссылок: 0
        public static int[] ArraySorting(int[] array)
        {
            return QuickSort(array, 0, array.Length - 1);
        }

        ссылка: 1
        static int Section(int[] array, int minI, int maxI)
        {
            var pin = minI - 1;
            for (var i = minI; i < maxI; i++)
            {
                if (array[i] < array[maxI])
                {
                    pin++;
                    Swap(ref array[pin], ref array[i]);
                }
            }

            pin++;
            Swap(ref array[pin], ref array[maxI]);
            return pin;
        }

        Ссылок: 3
        static int[] QuickSort(int[] array, int minI, int maxI)
        {
            if (minI >= maxI)
            {
                return array;
            }

            var pinIndex = Section(array, minI, maxI);
            QuickSort(array, minI, pinIndex - 1);
            QuickSort(array, pinIndex + 1, maxI);

            return array;
```

Рисунок 9 – Модуль QuickSorting

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

Быстрая сортировка — это сортировка сравнения, означающая, что она может сортировать элементы любого типа, для которых определено отношение "меньше" (формально, общий порядок). Большинство реализаций быстрой сортировки нестабильны, что означает, что относительный порядок одинаковых элементов сортировки не сохраняется. [3]

Алгоритм состоит из трёх шагов:

1. Выбрать элемент из массива. Назовём его опорным.
2. Разбиение: перераспределение элементов в массиве таким образом, что элементы, меньшие опорного, помещаются перед ним, а большие или равные - после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

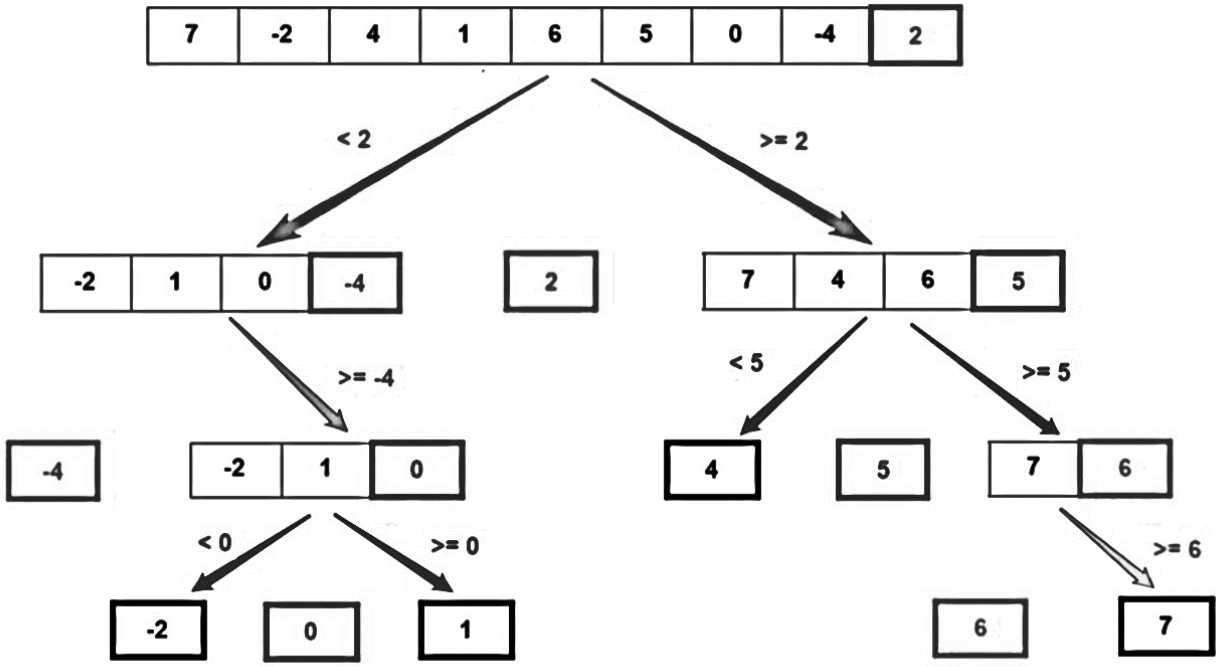
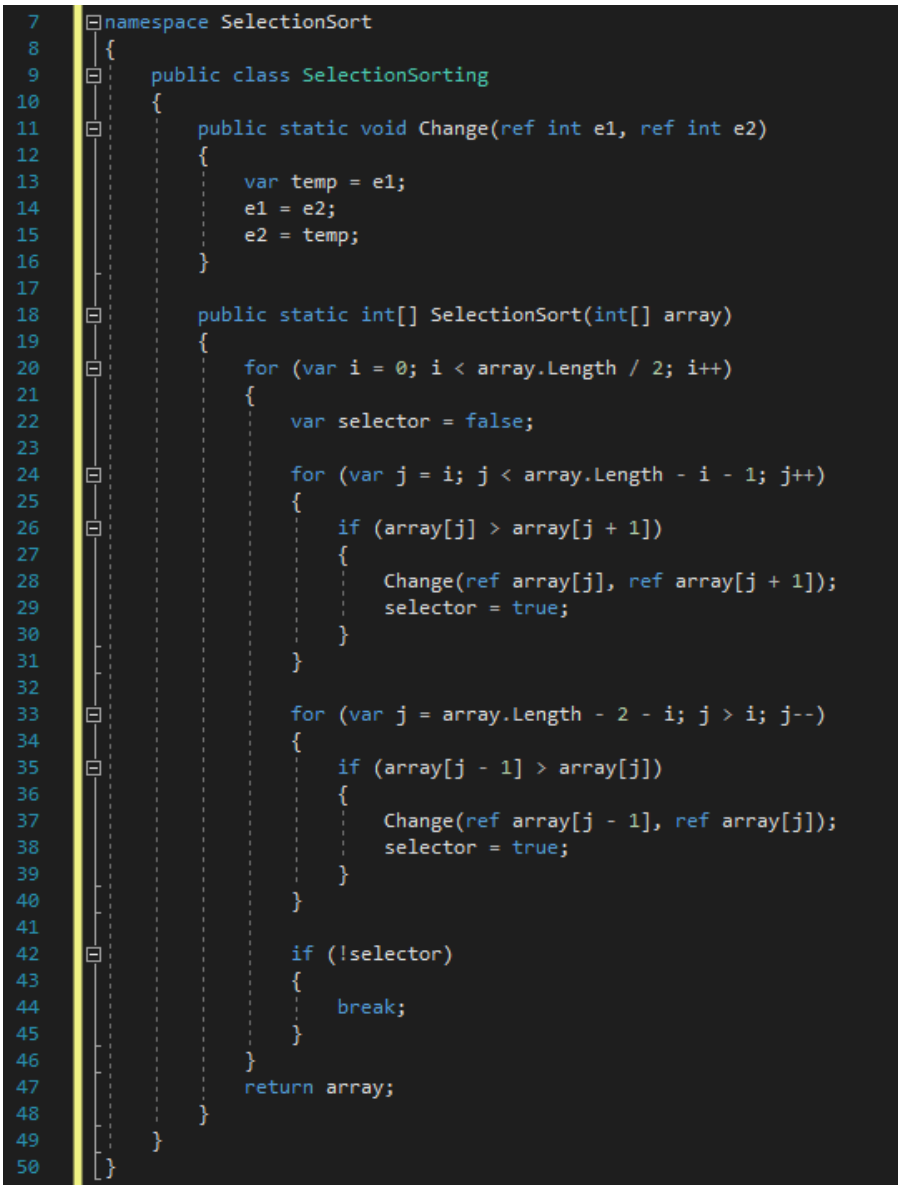


Рисунок 10 – Пример реализации быстрой сортировки

2.5 Модуль SelectionSorting



```
namespace SelectionSort
{
    public class SelectionSorting
    {
        public static void Change(ref int e1, ref int e2)
        {
            var temp = e1;
            e1 = e2;
            e2 = temp;
        }

        public static int[] SelectionSort(int[] array)
        {
            for (var i = 0; i < array.Length / 2; i++)
            {
                var selector = false;

                for (var j = i; j < array.Length - i - 1; j++)
                {
                    if (array[j] > array[j + 1])
                    {
                        Change(ref array[j], ref array[j + 1]);
                        selector = true;
                    }
                }

                for (var j = array.Length - 2 - i; j > i; j--)
                {
                    if (array[j - 1] > array[j])
                    {
                        Change(ref array[j - 1], ref array[j]);
                        selector = true;
                    }
                }

                if (!selector)
                {
                    break;
                }
            }
            return array;
        }
    }
}
```

Рисунок 11 – Модуль SelectionSorting

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

Алгоритм сортировки выбором заключается в поиске на необработанном срезе массива или списка минимального значения и в дальнейшем обмене этого значения с первым элементом необработанного среза. На следующем шаге необработанный срез уменьшается на один элемент. [4]

Алгоритм действий:

1. Найти наименьшее значение в массиве.
2. Записать его в начало массива, а первый элемент - на место, где раньше стоял наименьший.
3. Снова найти наименьший элемент в массиве. При этом в поиске не участвует первый элемент.
4. Второй минимум поместить на второе место списка. Второй элемент при этом перемещается на освободившееся место.
5. Продолжать выполнять поиск и обмен, пока не будет достигнут конец списка.

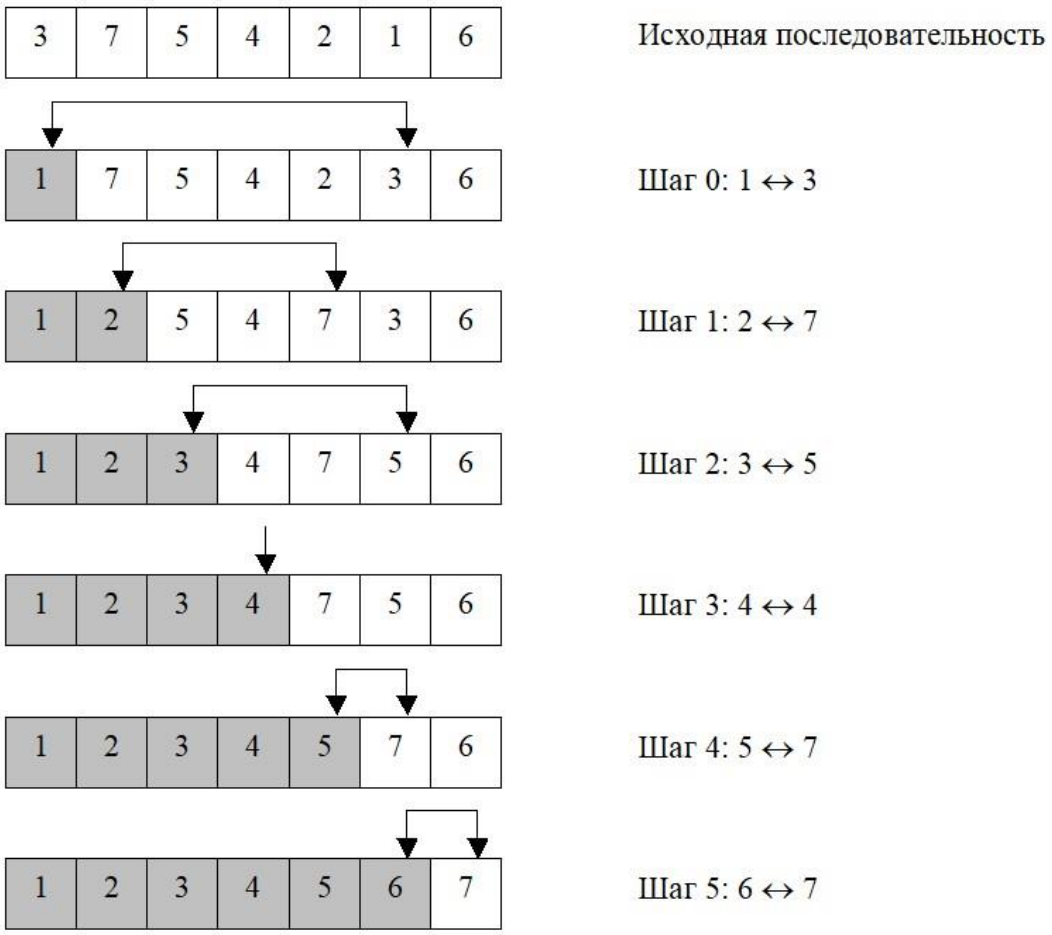


Рисунок 12 – Пример реализации сортировки методом выбора

ЗАКЛЮЧЕНИЕ

Я, Тейфуков Руслан Наримановичч, 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:

4. Необходимо разработать программный комплекс по демонстрации работы алгоритмов сортировки массивов данных (реализовать не менее 4 алгоритмов сортировки, которые выбрать самостоятельно):

4.1 Разработать техническое задание на программный продукт.

4.2 Разработать спецификацию на программный продукт.

4.3 Разработать функциональную диаграмму программного продукта, диаграмму потоков данных программных модулей продукта.

4.4 Разработать функциональную схему программного продукта, составить блок-схемы программных модулей программного продукта.

4.5 Разработать коды программных модулей программного продукта.

4.6 Разработать пользовательский интерфейса программного продукта в визуальной среде.

4.7 Выполнить интеграцию программных модулей в программный продукт.

4.8 Разработать процедуру тестирования программного продукта. Выполнить тестирование программного продукта. Результат тестирования оформить протоколом тестирования.

4.9 Разработать справочную систему программного продукта.

4.10 Разработать руководства оператора (пользователя).

5. Создать аккаунт в GitHub. Создать папку проекта. В папку загрузить разработанный программный комплекс, всю разработанную документацию к проекту (п.п.1.1 – 1.10).

6. Составить отчет о выполнении.

Ссылка на репозиторий с выполненной работой:
https://github.com/Co0kie444/ArraySorting

СПИСОК ЛИТЕРАТУРЫ

1. Котлин А.П. Алгоритмы сортировки [электронный ресурс] bimbilik.github.io – URL: https://bimlibik.github.io/posts/sorting-algorithm/. – (дата обращения 20.03.2023)
2. Баргузин В.В. Алгоритмы сортировки массивов. [электронный ресурс] study.muctr.ru – URL: https://study.muctr.ru/pluginfile.php/263589/mod_resource/content/1/сортировка.pdf. – (дата обращения 20.03.2023)
3. Липачёв Е.К. Технология программирования. Методы сортировки данных [электронный ресурс] kpfu.ru – URL: https://kpfu.ru/staff_files/F1066117885/Metody_sortirovki.pdf. – (дата обращения 20.03.2023)
4. Граф А.Е. Алгоритмы сортировки и поиска [электронный ресурс] prog-cpp.ru – URL: https://prog-cpp.ru/algorithm-sort/. – (дата обращения 20.03.2023)
5. Сенегалович А.В. Основные виды сортировок и примеры их реализации [электронный ресурс] academy.yandex.com – URL: https://academy.yandex.com/journal/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii. – (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 QuickSort;
using BubbleSort;
using SelectionSort;
using MergeSort;



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)
        {

            string myArr = textBox4.Text;
```

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

        string texter = string.Join("",BubbleSorting.ArraySorting(result));

        textBox3.Text = texter;
    }


    private void textBox1_TextChanged(object sender, EventArgs e)
    {

        textBox4.Text = "";
        try
        {
            n = Convert.ToInt32(textBox1.Text);
        }
        catch
        {
            textBox1.Text = "";
            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)
    {
```

```
        string myArr = textBox4.Text;

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

        string texter = string.Join("", MergeSorting.ArraySorting(result));

        textBox3.Text = texter;

    }



    private void textBox4_TextChanged(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)
    {
```

```
            string myArr = textBox4.Text;

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

            int resl = result.Length;

            string texter = string.Join("", SelectionSorting.ArraySorting(result));

            textBox3.Text = texter;
        }


        private void button4_Click(object sender, EventArgs e)
        {
            string myArr = textBox4.Text;

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

            int resl = result.Length;

            string texter = string.Join("", QuickSorting.ArraySorting(result));

            textBox3.Text = texter;
        }


        private void label2_Click(object sender, EventArgs e)
        {

        }

    }
}
```