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)

Губский Иван Владимирович
(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

Цель данного проекта состоит в разработке программного комплекса, который будет демонстрировать работу алгоритмов сортировки массивов данных. Необходимо реализовать не менее четырех алгоритмов сортировки, которые будут выбраны самостоятельно.

Разработка программного обеспечения - это процесс создания программных продуктов, которые решают определенные задачи и удовлетворяют потребности пользователей. Этот процесс включает в себя несколько этапов, включая формулирование задачи, определение требований к программному продукту, проектирование, кодирование, тестирование и развертывание.

На этапе формулирования задачи и определения требований к программному продукту определяются функции, которые должен выполнять программный продукт, а также требования к его производительности, надежности, безопасности и другим параметрам.

Проектирование программного продукта включает в себя создание дизайна и архитектуры программного продукта, определение структуры данных, выбор используемых технологий и инструментов разработки.

На этапе кодирования разработчики пишут и отлаживают код программного продукта, используя выбранные технологии и инструменты разработки.

Тестирование программного продукта включает в себя проверку его работоспособности и соответствия требованиям, а также исправление ошибок и недостатков.

Наконец, на этапе развертывания программный продукт устанавливается на компьютеры пользователей и настраивается для их потребностей.

Вся работа по разработке программного обеспечения должна проводиться в соответствии с принципами программной инженерии, включая использование методов и инструментов управления проектами, тестирования и контроля качества.

Для успешного выполнения проекта необходимо выполнить следующие задачи:

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 интерфейс программного продукта в визуальной среде.
7. Выполнить интеграцию программных модулей в программный продукт.
8. Разработать процедуру тестирования программного продукта.
9. Выполнить тестирование программного продукта.
10. Разработать справочную систему программного продукта и руководства оператора (пользователя).

ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Разработка технического задания

ТЗ – основополагающий документ, которым руководствуются разработчики и проектировщики, приступая к разработке нового изделия. Оно определяет основные направления разработки: конструкции и принципа работы будущего изделия. ТЗ заявляет, с одной стороны, о потребностях общества в новых изделиях, с другой – о технических и технико-экономических характеристиках изделия.

Техническое задание является начальным этапом работ и составляется на все разработки и виды работ, необходимые для создания нового изделия. Оно может предшествовать научно-исследовательским и опытно-конструкторским работам (НИОКР) по разработке средств механизации и автоматизации, отдельных узлов и систем, технологии, измерительных средств, средств контроля и других изделий (выполнение работы, оказание услуги, промышленный комплекс, прибор, машина, аппарат, система управления, информационная система, нормативная документация (например, стандарт) и т. д.).

Требования, включаемые в ТЗ, должны основываться на современных достижениях науки и техники, на итогах выполненных научно-исследовательских и экспериментальных работ. ТЗ должно устанавливать следующие показатели разрабатываемого изделия:

- основное назначение, технические и тактико-технические характеристики, уровень стандартизации и унификации;
- технико-экономические показатели;
- патентно-правовые показатели;
- специальные требования к изделию и др.

В технических заданиях оговариваются этапы разработки и сроки выполнения каждого этапа, сроки разработки в целом. Качество ТЗ обеспечивается объемом и полнотой сбора материалов, необходимых для разработки. При разработке используются следующие материалы:

- научно-техническая информация;
- патентная информация;
- характеристика рынка сбыта;
- характеристика производства, на котором изделие будет изготавливаться (технологическая оснащенность, квалификация кадров, технологическая дисциплина, уровень организации труда и др.).

При разработке ТЗ разработчик учитывает информацию об аналогичной продукции, содержащуюся в базах данных (общероссийской и региональных), созданных в Госстандарте России на основе каталожных листов продукции.

Техническое задание разрабатывается, как правило, организацией-разработчиком изделия. Сформулировать задачу максимально полно и грамотно, обосновать необходимость её решения – главная цель ТЗ. Исполнитель выполняет его в контакте с заказчиком. Обязанность заказчика – предъявить разработчику исходные данные для разработки изделия.

ТЗ разрабатывают и утверждают в порядке, установленном заказчиком и разработчиком. К разработке ТЗ могут привлекаться другие заинтересованные организации (предприятия): изготовитель, торговая (посредническая) организация, страховая организация, организация-проектировщик, монтажная организация и др.

Для подтверждения отдельных требований к продукции, в том числе требований безопасности, охраны здоровья и окружающей среды, а также оценки технического уровня продукции, ТЗ может быть направлено разработчиком или заказчиком на экспертизу (заключение) в сторонние организации. Решение по полученным заключениям принимают разработчик и заказчик до утверждения ТЗ.

К техническому заданию прилагаются схемы и эскизы по конструкции будущего изделия, а для технологических разработок – технологические и технико-экономические показатели существующего производства. Техническое задание должно содержать максимум информации, облегчающей работу над изделием и сокращающей сроки разработки.

Техническое задание на разработку программного обеспечения:

1. Описание функциональных требований:

Программное обеспечение должно иметь способность принимать данные различных форматов.

Программное обеспечение должно иметь способность выводить данные на экран, и автоматически в буфер обмена.

Программное обеспечение должно содержать алгоритмы сортировки данных.

2. Описание нефункциональных требований:

Программное обеспечение должно иметь простой и понятный пользовательский интерфейс с возможностью выбора метода сортировки.

Программное обеспечение должно обладать высокой производительностью и эффективностью работы с большими объемами данных.

Программное обеспечение должно быть совместимо с операционными системами типа Windows.

3. Технические требования:

Программное обеспечение должно быть написано на языке программирования C#.

Программное обеспечение должно использовать интегрируемые модули для работы с данными.

Программное обеспечение должно использовать эффективные алгоритмы сортировки данных.

4. Требования к тестированию:

Провести модульное тестирование каждой функции программы.

Провести интеграционное тестирование на разных операционных системах.

Провести функциональное тестирование для проверки соответствия программы функциональным требованиям.

Провести тестирование производительности для оценки скорости работы программы при обработке больших объемов данных.

Провести тестирование безопасности для проверки защищенности программы от несанкционированного доступа.

Предоставить документацию на программу, включающую описание функций, алгоритмов, и результаты всех тестов.

Наименование программы: "Губский Иван4ПКС18".

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:

1.2 Разработка спецификаций

Разработка спецификации (спецификации требований) - это процесс определения требований к программному продукту и их описания в документе. Этот документ определяет, что должно быть разработано и каким образом, а также является основой для оценки качества и результативности проекта. Вот основные шаги для разработки спецификации:

Спецификация требований на разработку программного обеспечения:

Общие требования

Функциональные требования

1) Программное обеспечение должно иметь возможность принимать данные.

2) Программное обеспечение должно иметь возможность обрабатывать полученные данные с использованием алгоритмов сортировки.

3) Программное обеспечение должно иметь возможность выводить обработанные данные на экран.

4) Программное обеспечение должно иметь собственный интерфейс, который позволит пользователям взаимодействовать с программой и управлять ее функциями.

Нефункциональные требования

1) Программное обеспечение должно быть разработано на языке программирования C# с использованием технологии .Net и Windows Form.

2) Программное обеспечение должно быть совместимо с операционными системами типа Windows.

3) Программное обеспечение должно обладать высокой производительностью и эффективностью в использовании ресурсов компьютера.

4) Программное обеспечение должно иметь документацию, включающую инструкции по установке, настройке, использованию и техническую поддержку.

5) Программное обеспечение должно быть простым в использовании и иметь интуитивно понятный пользовательский интерфейс.

Требования к производительности

1) Программное обеспечение должно быть способно обрабатывать большие объемы данных без замедления производительности.

2) Программное обеспечение должно быть оптимизировано для использования ресурсов компьютера, включая процессор, оперативную память и диск.

3) Программное обеспечение должно иметь возможность обрабатывать данные в реальном времени.

Требования к тестированию

1) Программное обеспечение должно проходить обязательное тестирование перед выпуском в эксплуатацию.

2) Тестирование должно включать проверку функциональности, производительности и безопасности программного обеспечения.

3) Результаты тестирования должны быть документированы и доступны для ознакомления.

1.3 Разработка диаграмм

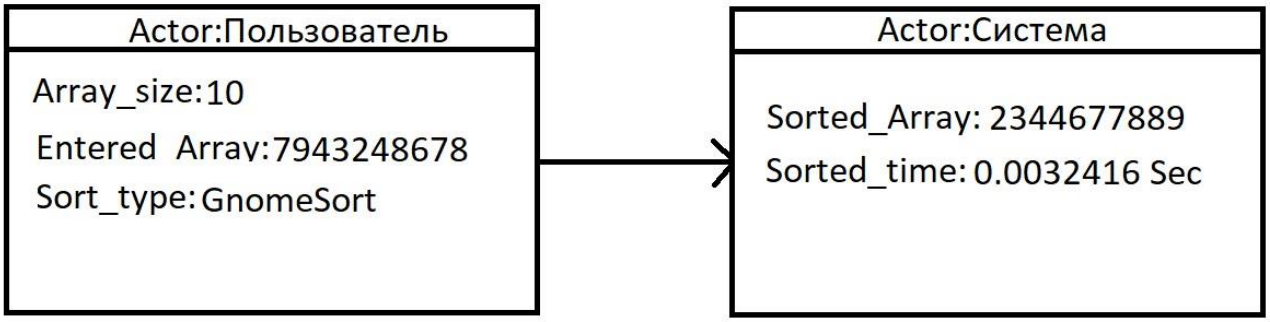


Рисунок 1 – Первая UML диаграмма

На данном рисунке представлена первая UML диаграмма с выполнением алгоритма сортировки массива методом Гномья сортировка с размерностью массива 10 и скоростью выполнения 0.0032 сек.

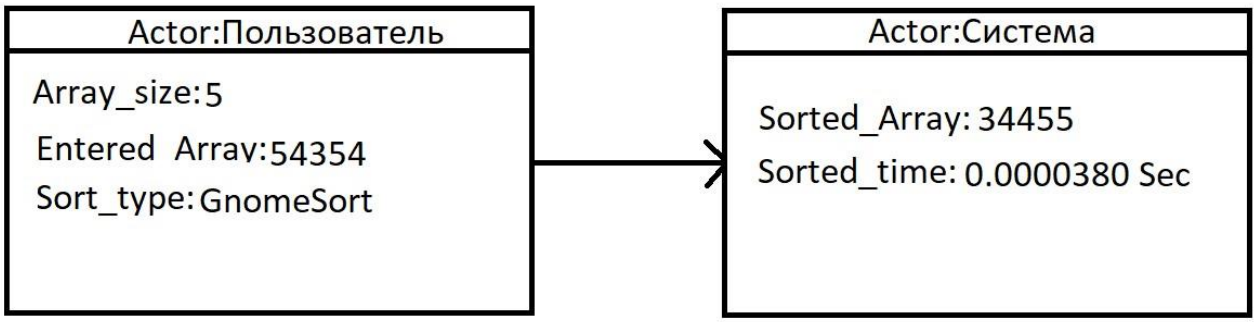


Рисунок 2 – Вторая UML диаграмма

а данном рисунке представлена вторая UML диаграмма с выполнением алгоритма сортировки массива методом Гномья сортировка с размерностью массива 5 и скоростью выполнения 0.000038 сек.

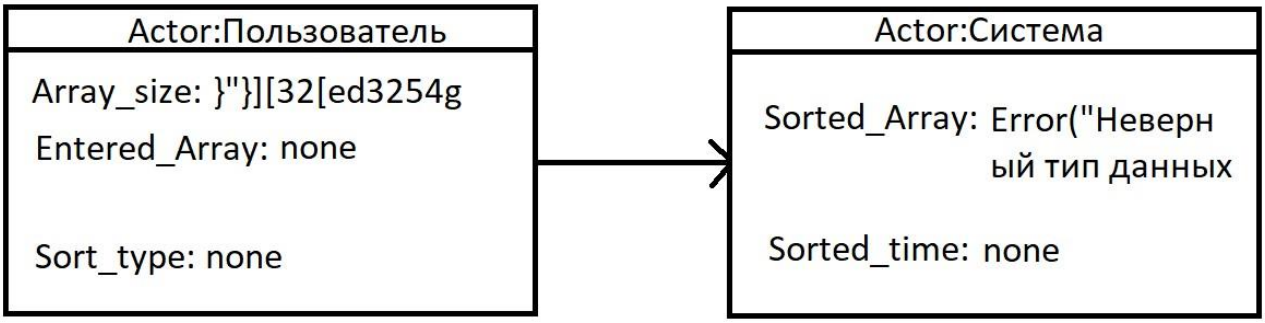


Рисунок 3 – Третья UML диаграмма

на данном рисунке представлена третья UML диаграмма с выполнением алгоритма с заведомо некорректными данными.

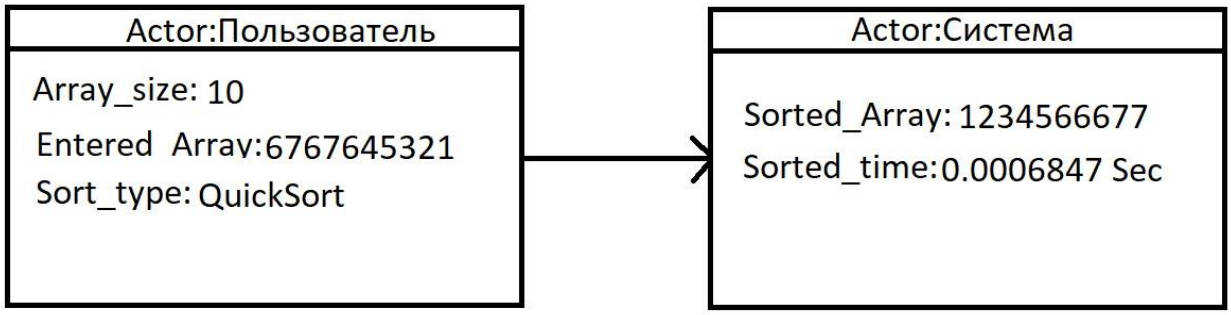


Рисунок 4 – Четвёртая UML диаграмма

на данном рисунке представлена четвёртая UML диаграмма с выполнением алгоритма сортировки массива методом Быстрой сортировки с размерностью массива 10 и скоростью выполнения 0.0006847 сек.

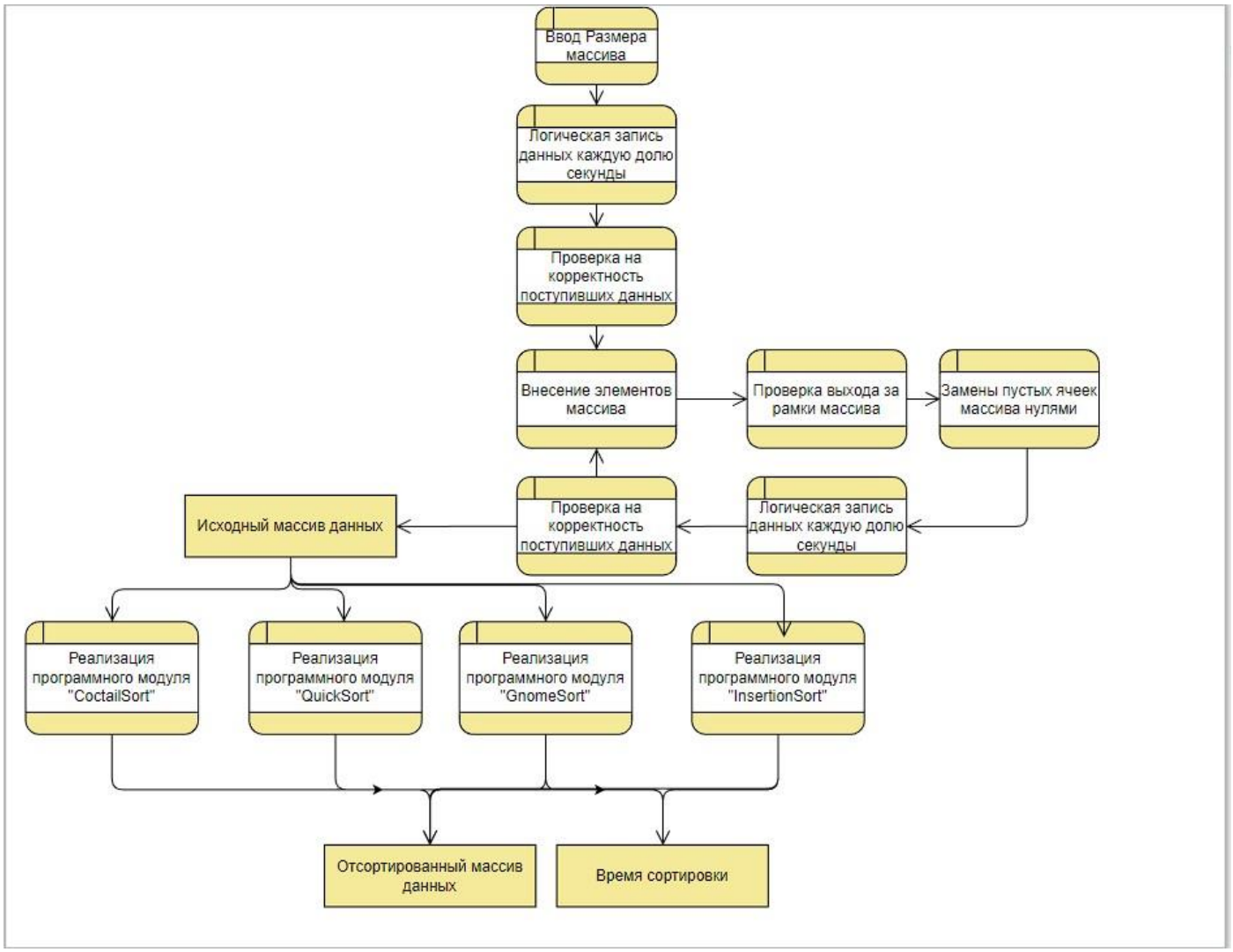


Рисунок 5 – Диаграмма потока данных

На данном рисунке представлена диаграмма потока данных в которой показывается какой именно путь данные проходят от начала программы до конечного результата, а именно: Отсортированного массива чисел.

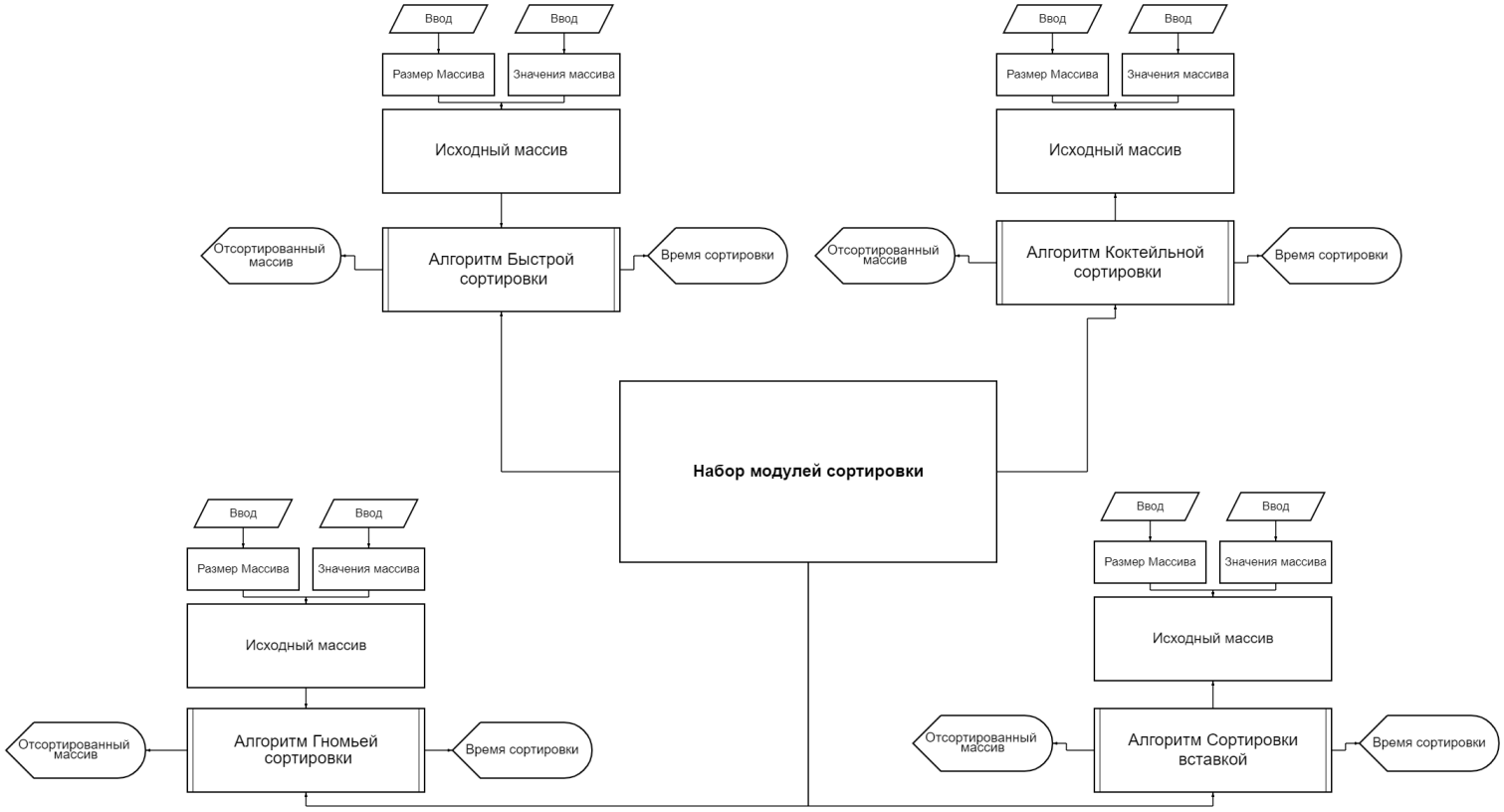


Рисунок 6 – Функциональная схема продукта

На данном рисунке представлена функциональная схема продукта где наглядно описана функциональная составляющая программного продукта

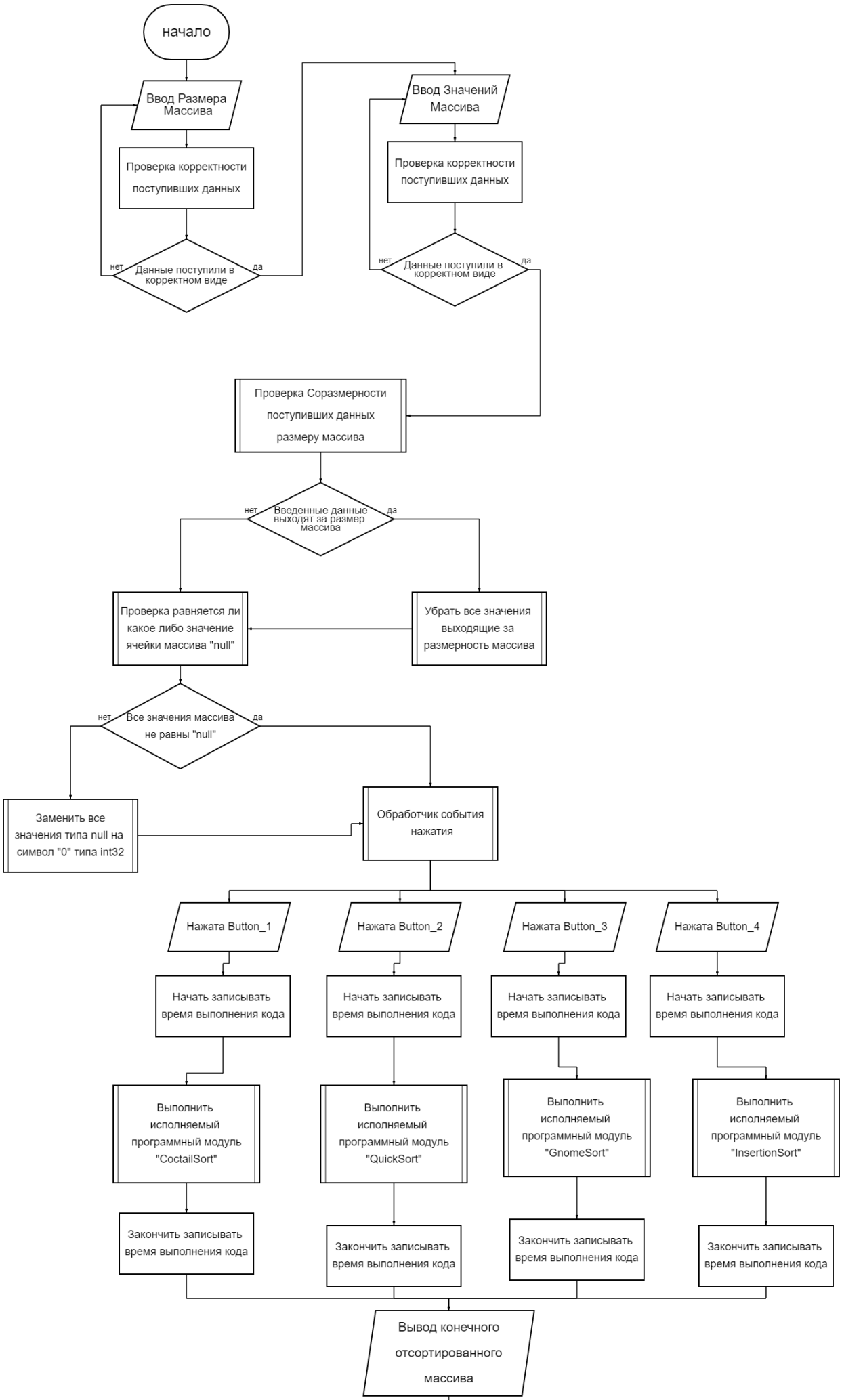

начало
Ввод Размера Массива
Проверка корректности поступивших данных
нет
Данные поступили в корректном виде
да
Ввод Значений Массива
Проверка корректности поступивших данных
нет
Данные поступили в корректном виде
да
Проверка Соразмерности поступивших данных размеру массива
нет
Введенные данные выходят за размер массива
да
Проверка равняется ли какое либо значение ячейки массива "null"
Убрать все значения выходящие за размерность массива
нет
Все значения массива не равны "null"
да
Заменить все значения типа null на символ "0" типа int32
Обработчик события нажатия
Нажата Button_1
Нажата Button_2
Нажата Button_3
Нажата Button_4
Начать записывать время выполнения кода
Начать записывать время выполнения кода
Начать записывать время выполнения кода
Начать записывать время выполнения кода
Выполнить исполняемый программный модуль "CoctailSort"
Выполнить исполняемый программный модуль "QuickSort"
Выполнить исполняемый программный модуль "GnomeSort"
Выполнить исполняемый программный модуль "InsertionSort"
Закончить записывать время выполнения кода
Закончить записывать время выполнения кода
Закончить записывать время выполнения кода
Закончить записывать время выполнения кода
Вывод конечного отсортированного массива

Рисунок 7 – Блок схема программы

На данном рисунке представлена блок схема реализованной программы где наглядно описана логика программного продукта со всеми необходимыми этапами.

ГЛАВА 2. МОДУЛИ ПРОГРАММЫ

2.1 Модуль интерфейса

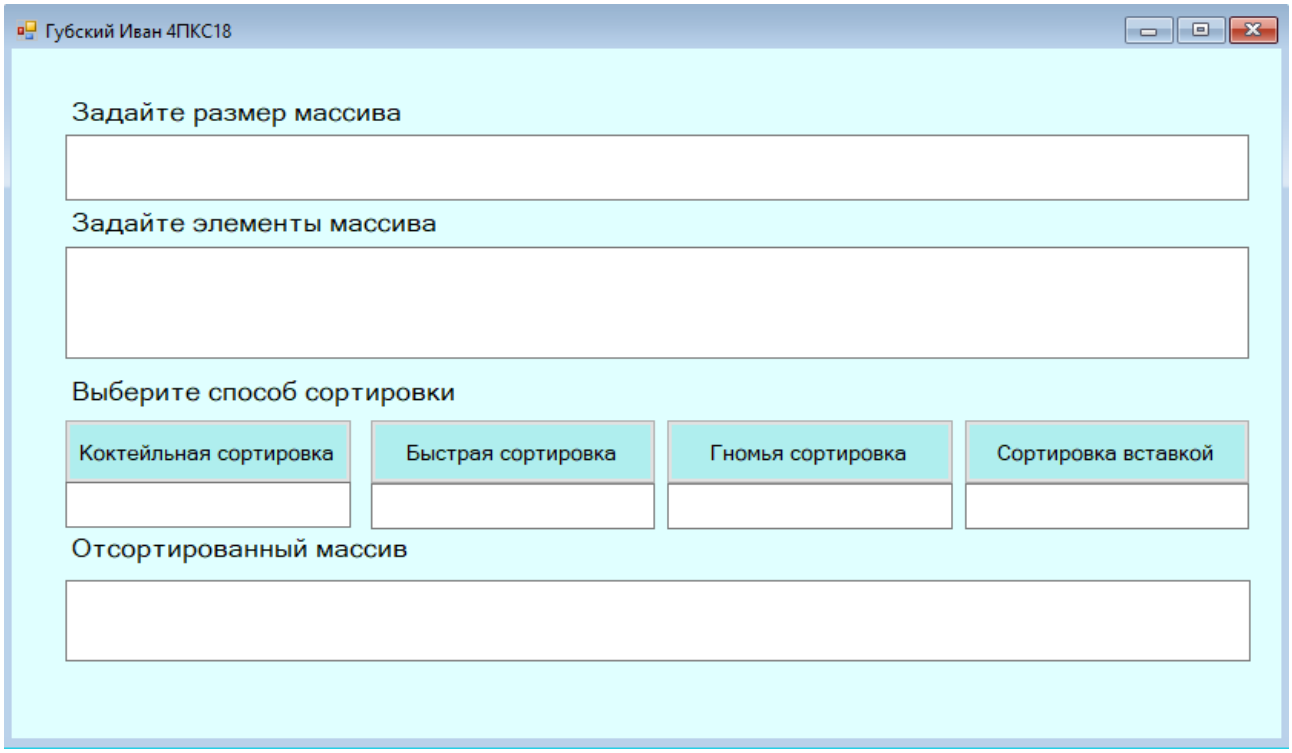
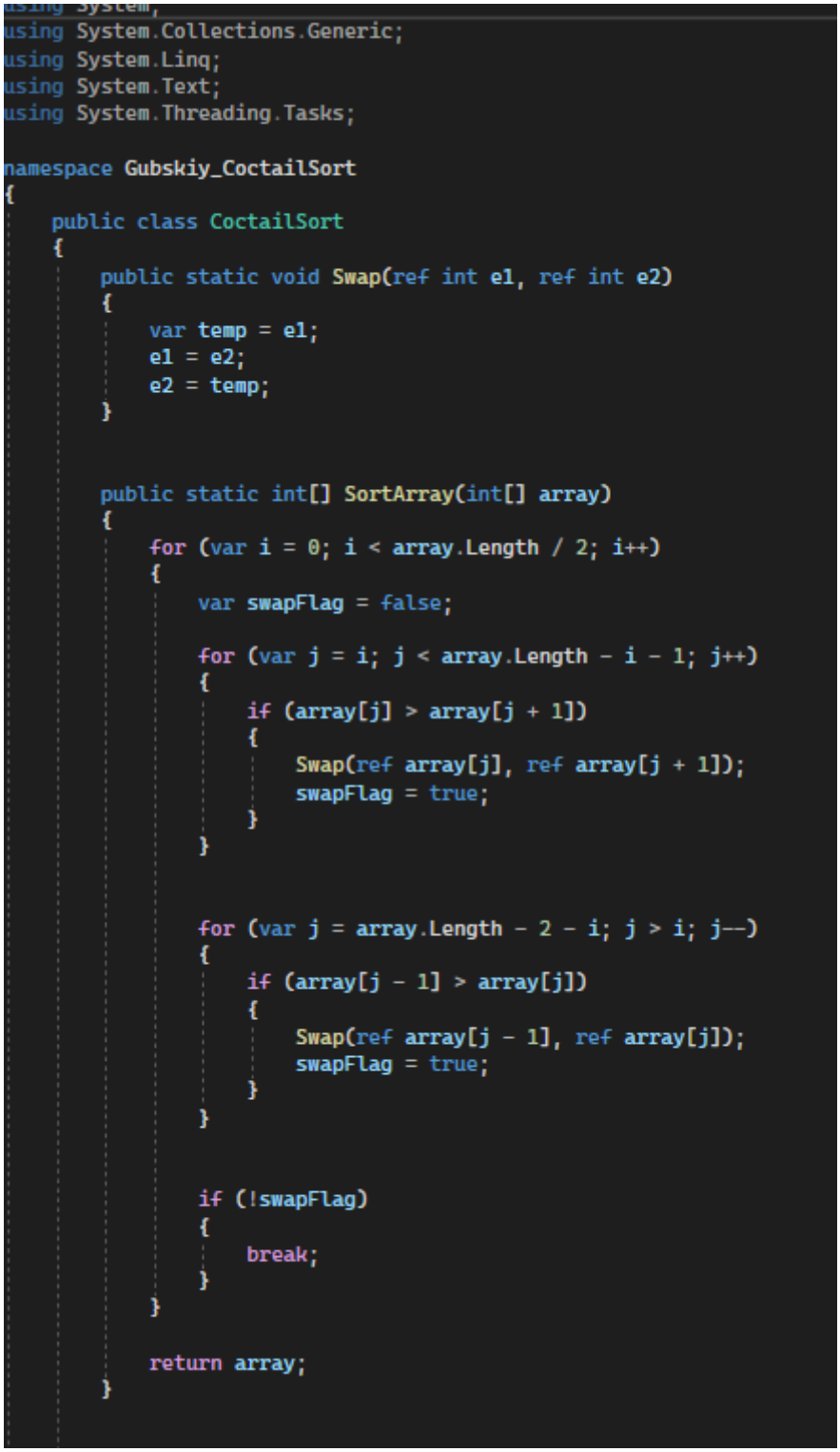


Рисунок 8 – Модуль интерфейса

На данном рисунке представлен общий интерфейс программы, в нем реализованы такие кнопки как:” Коктейльная Сортировка”,” Быстрая сортировка”,” Гномья сортировка», «Сортировка вставкой”, а также окна: “Задайте размер массива”, “Задайте элементы массива», «” Выберите способ сортировки”, “Отсортированный массив”.

2.2 Модуль CoctailSort



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

namespace Gubskiy_CoctailSort
{
    public class CoctailSort
    {
        public static void Swap(ref int e1, ref int e2)
        {
            var temp = e1;
            e1 = e2;
            e2 = temp;
        }


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

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


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


                if (!swapFlag)
                {
                    break;
                }
            }

            return array;
        }
```

Рисунок 9 – Модуль CoctailSort

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

Коктейльная сортировка - это улучшение и вариация барботажной сортировки, также известная как «двусторонняя барботажная сортировка». Коктейльная сортировка - это сортировка от низкого до высокого, а затем от высокого к низкому (выберите самые большие и самые маленькие элементы), чем Эффективность сортировки пузырьков несколько лучше, потому что сортировка пузырьков сравнивается только в одном направлении (от низкого до высокого), и только один элемент перемещается за цикл.

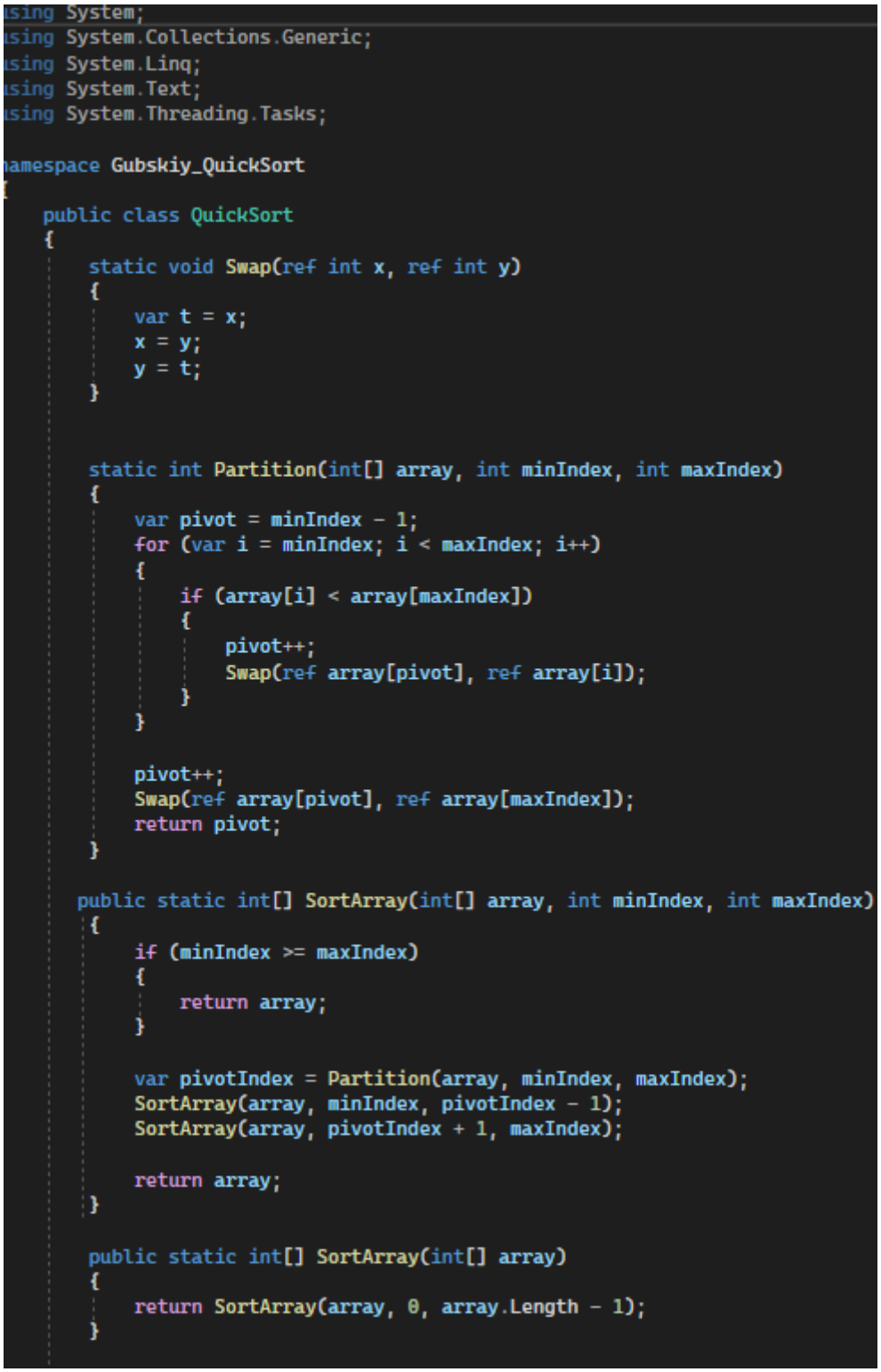
2 Алгоритм анализа：

1. Сначала отсортируйте массив слева направо в пузырьковом порядке (в порядке возрастания), затем самый большой элемент перейдет в крайнее правое положение.

2. Затем отсортируйте массив справа налево в пузырьковом порядке (в порядке убывания), затем наименьший элемент перейдет в крайнее левое положение.

3. Цикл 1, 2 шага, последовательное изменение направления пузырьков и продолжение сужения диапазона несортированных элементов до конца последнего элемента [1]

2.3 Модуль QuickSort



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

namespace Gubskiy_QuickSort
{
    public class QuickSort
    {
        static void Swap(ref int x, ref int y)
        {
            var t = x;
            x = y;
            y = t;
        }


        static int Partition(int[] array, int minIndex, int maxIndex)
        {
            var pivot = minIndex - 1;
            for (var i = minIndex; i < maxIndex; i++)
            {
                if (array[i] < array[maxIndex])
                {
                    pivot++;
                    Swap(ref array[pivot], ref array[i]);
                }
            }

            pivot++;
            Swap(ref array[pivot], ref array[maxIndex]);
            return pivot;
        }

       public static int[] SortArray(int[] array, int minIndex, int maxIndex)
        {
            if (minIndex >= maxIndex)
            {
                return array;
            }

            var pivotIndex = Partition(array, minIndex, maxIndex);
            SortArray(array, minIndex, pivotIndex - 1);
            SortArray(array, pivotIndex + 1, maxIndex);

            return array;
        }

        public static int[] SortArray(int[] array)
        {
            return SortArray(array, 0, array.Length - 1);
        }
```

Рисунок 10 – Модуль QuickSort

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

Алгоритм быстрой сортировки является рекурсивным, поэтому для простоты процедура на вход будет принимать границы участка массива от l включительно и до r не включительно. Понятно, что для того, чтобы отсортировать весь массив, в качестве параметра l надо передать 0, а в качестве r — n, где по традиции n обозначает длину массива.

В основе алгоритма быстрой сортировке лежит процедура partition. Partition выбирает некоторый элемент массива и переставляет элементы участка массива таким образом, чтобы массив разбился на 2 части: левая часть содержит элементы, которые меньше этого элемента, а правая часть содержит элементы, которые больше или равны этого элемента. Такой разделяющий элемент называется пивотом.

Реализация partiion'a:

```
partition(l, r):
    pivot = a[random(l ... r - 1)]
    m = l
    for i = l ... r - 1:
        if a[i] < pivot:
            swap(a[i], a[m])
            m++
    return m
```

Пивот в нашем случае выбирается случайным образом. Такой алгоритм называется рандомизированным. На самом деле пивот можно выбирать самым разным образом: либо брать случайный элемент, либо брать первый / последний элемент участка, либо выбирать его каким-то «умным» образом. Выбор пивота является очень важным для итоговой сложности алгоритма сортировки, но об этом несколько позже. Сложность же процедуры partition — O(n), где n = r — l — длина участка.

Теперь используем partition для реализации сортировки:

Реализация partiion'a:

```
sort(l, r):
    if r - l = 1:
        return
    m = partition(l, r)
    sort(l, m)
    sort(m, r)
```

Крайний случай — массив из одного элемента обладает свойством упорядоченности. Если массив длинный, то применяем partition и вызываем процедуру рекурсивно для двух половин массива.

Если прогнать написанную сортировку на примере массива 1 2 2, то можно заметить, что она никогда не закончится. Почему так получилось?

При написании partition мы сделали допущение — все элементы массива должны быть уникальны. В противном случае возвращаемое значение m будет равно l и рекурсия никогда не закончится, потому как sort(l, m) будет вызывать sort(l, l) и sort(l, m). Для решения данной проблемы надо массив разделять не на 2 части (< pivot и >= pivot), а на 3 части (< pivot, = pivot, > pivot) и вызывать рекурсивно сортировку для 1-ой и 3-ей частей. [2]

2.4 Модуль GnomeSort

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

namespace Gubskiy_GnomeSort
{
    public class GnomeSort
    {


        public static void Swap(ref int item1, ref int item2)
        {
            var temp = item1;
            item1 = item2;
            item2 = temp;
        }


        public static int[] SortArray(int[] unsortedArray)
        {
            var index = 1;
            var nextIndex = index + 1;

            while (index < unsortedArray.Length)
            {
                if (unsortedArray[index - 1] < unsortedArray[index])
                {
                    index = nextIndex;
                    nextIndex++;
                }
                else
                {
                    Swap(ref unsortedArray[index - 1], ref unsortedArray[index]);
                    index--;
                    if (index == 0)
                    {
                        index = nextIndex;
                        nextIndex++;
                    }
                }
            }

            return unsortedArray;

        }
    }
```

Рисунок 11 – Модуль GnomeSort

Гномья сортировка (Gnome sort) – простой в реализации алгоритм сортировки массива, назван в честь садового гнома, который предположительно таким методом сортирует садовые горшки.

Принцип работы алгоритма сортировки Гнома

Алгоритм находит первую пару неотсортированных элементов массива и меняет их местами. При этом учитывается факт, что обмент приводит к неправильному расположению элементов примыкающих с обеих сторон к

только что переставленным. Поскольку все элементы массива после переставленных не отсортированны, необходимо перепроверить только элементы до переставленных. [3]

2.5 Модуль InsertionSort

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

namespace Gubskiy_InsertionSort
{
    public class InsertionSort
    {
        public static int[] SortArray(int[] array)
        {
            int length = array.Length;

            for (int i = 1; i < length; i++)
            {
                var key = array[i];
                var flag = 0;
                for (int j = i - 1; j >= 0 && flag != 1;)
                {
                    if (key < array[j])
                    {
                        array[j + 1] = array[j];
                        j--;
                        array[j + 1] = key;
                    }
                    else flag = 1;
                }
            }
            return array;
        }
```

Рисунок 12 – Модуль InsertionSort

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

Сортировка вставками - алгоритм, при котором каждый последующий элемент массива сравнивается с предыдущими элементами (отсортированными) и вставляется в нужную позицию.

Общая идея алгоритма: Сравниваем второй элемент с первым элементом массива и при необходимости меняем их местами. Условно эти элементы (первый и второй) будут являться отсортированным массивом, остальные элементы - неотсортированным. [4]

Сортировка массива — расположение его элементов в некотором заданном порядке. В отсортированном массиве поиск элемента можно осуществлять, не просматривая весь массив. Например, в случае сортировки в порядке возрастания минимальный элемент массива всегда будет находиться на первом месте. Задача сортировки, как и любая другая задача, может решаться множеством способов, каждый из которых имеет как достоинства, так и недостатки. [5]

2.6 Процедура тестирования

Требования и исходной системе:

- Операционная система (OS / ОС): Windows 7 32-bit SP1
- Центральный процессор (CPU / ЦПУ): 2.0GHz
- Оперативная память (RAM / ОЗУ): 2 ГБ RAM
- Видеокарта (GPU): DX10 compatible или лучше
- DirectX: Версия 10
- Сеть (интернет-подключение): Широкополосное подключение к интернету
- Накопитель (HDD / SSD): 2000 MB свободного места

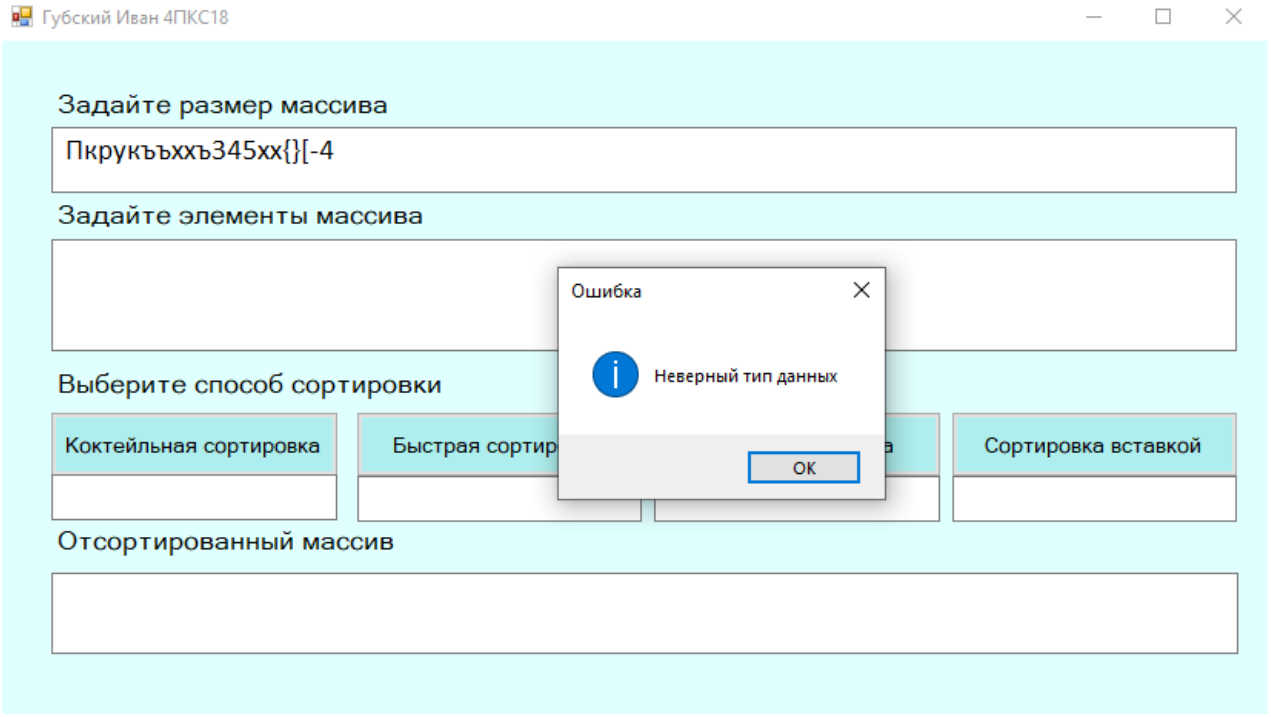


Рисунок 13 – Результат проведения первого теста

Данный рисунок представляет собой графическую схему тестирования алгоритмов сортировки и проверки введенных данных. На рисунке изображено несколько блоков, которые выполняют определенные функции.

Второй блок на рисунке представляет собой входные данные, которые были введены пользователем. В данном случае, алгоритм обнаружил в введенных данных символов "Пкрукъъччъхх{}[-", которые невозможно преобразовать в число.

Комплексом проверки данных отвечает за проверку введенных данных на наличие ошибок. Алгоритм производит анализ входных данных и обнаруживает символ " Пкрукъъччъхх{}[-", которые не могут быть преобразованы в число. В результате этой проверки алгоритм выдает ошибку.

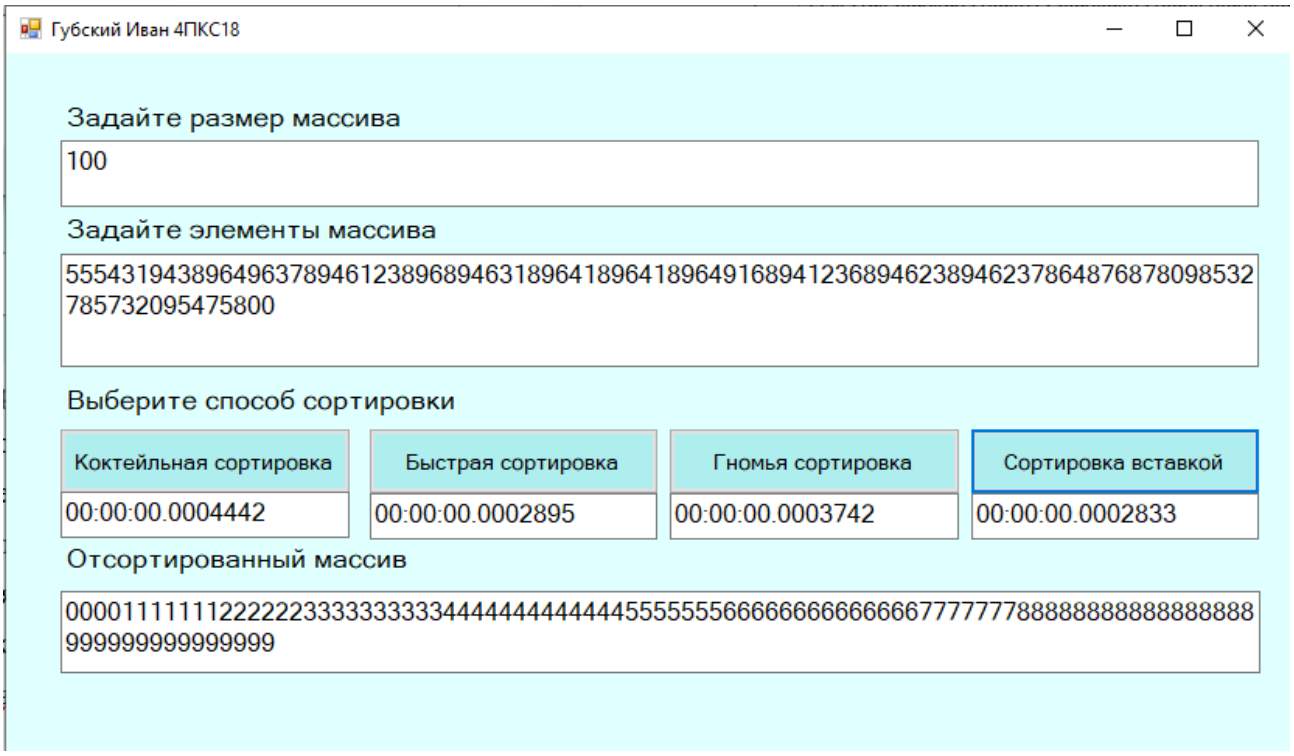


Рисунок 14 – Результат проведения второго теста

На данном рисунке представлено тестирование алгоритмов сортировок и проверки введённых данных.

После ввода данных, алгоритм проверил их на правильность и подтвердил, что введенные данные являются корректными. Затем алгоритм выполнил сортировку данных, используя все имеющиеся алгоритмы.

2.7 Руководство пользователя

Введение:

- Область применения: Программирование оперирование ЭВМ
- Описание возможностей: сортировка массива 4-мя методами, а также возможность задания размерности массива и его значений
- Уровень подготовки пользователя: умение работать с операционными системами типа Windows, умение печатать на клавиатуре и пользоваться компьютерной мышью.
- перечень документации для пользователя: текущее руководство пользователя.

Назначение и условия применения:

-Данный программный продукт предназначен для автоматизации процесса сортировки массива.

-Условиями применения являются задание конечного размера массива и ввод целых действительных чисел.

Подготовка к работе:

- Загрузочный пакет программы

- Порядок загрузки программы:

Распаковка загрузочного пакета, взаимодействие с корневым exe файлом по средству его открытия, загрузка приложения.

- порядок загрузки данных:

Ввод с клавиатуры значения размерности массива

Ручной ввод значений массива

Клик по одной из 4-х кнопок для выбора метода сортировки

Аварийные ситуации:

В случае какого-либо отклонения от правильной работы следует закрыть программу через диспетчера задач в главном его окне по средству по средству вызова диспетчера задач с помощью сочетания клавиш ctrl + alt + del и последующего открытия программы.

ЗАКЛЮЧЕНИЕ

Я, Губский Иван Владимирович, 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):

1.1 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.

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

СПИСОК ЛИТЕРАТУРЫ

1. Савельева А. С. Коктейльная сортировка [электронный ресурс] russianblogs.com – URL: https://russianblogs.com/article/8827516452/. – (дата обращения 20.03.2023)
2. Федоров М. В. Сортировка массивов методом быстрой сортировки [электронный ресурс] techiedelight.com – URL: https://www.techiedelight.com/ru/quicksort/. – (дата обращения 20.03.2023)
3. Казаков Л. В. Алгоритм Гномьей сортировки [электронный ресурс] programm.top – URL https://programm.top/c-sharp/algorithm/array-sort/gnome-sort/. – (дата обращения 20.03.2023)
4. Григорьев М. Д. Сортировка Вставкой [электронный ресурс] techiedelight.com – URL: https://www.techiedelight.com/ru/insertion-sort-iterative-recursive/. – (дата обращения 20.03.2023)
5. Смирнова С. М. Основные методы сортировки массивов [электронный ресурс] techiedelight.com – URL: https://www.techiedelight.com/ru/. – (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 System.Diagnostics;
using Gubskiy_CoctailSort;
using Gubskiy_GnomeSort;
using Gubskiy_InsertionSort;
using Gubskiy_QuickSort;
namespace Gubskiy_Ivan_4PCS18
{
   public partial class Form1 : Form
   {
      int n = 0;
      public Form1()
      {
         InitializeComponent();
      }
      private void textBox3_TextChanged(object sender, EventArgs e)
      {
         int[] x = new int[n];
         if (x.Length > textBox3.TextLength)
         {
```

```
        int a = x.Length - textBox3.TextLength;
        while (a > 0)
        {
            a--;
            textBox3.Text = textBox3.Text + 0;
        }
    }
    else
    if (x.Length < textBox3.TextLength)
    {
        int a = textBox3.TextLength - x.Length;

        textBox3.Text = textBox3.Text.Remove(textBox3.Text.Length - a);
        a = 0;

    }
}


private void textBox1_TextChanged(object sender, EventArgs e)
{

    textBox3.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++)
            {
                textBox3.Text = textBox3.Text + x[i];
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            var sw = new Stopwatch();
            sw.Start();
            string myArr = textBox3.Text;
            int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) -
48).ToArray();
            string texter = string.Join("", QuickSort.SortArray(result));
            textBox5.Text = texter;
            sw.Stop();
            textBox6.Text = sw.Elapsed.ToString();
        }

        private void button4_Click(object sender, EventArgs e)
```

```
        {
            var sw = new Stopwatch();
            sw.Start();
            string myArr = textBox3.Text;
            int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) -
48).ToArray();
            int resl = result.Length;
            string texter = string.Join("", InsertionSort.SortArray(result));
            sw.Stop();
            textBox2.Text = sw.Elapsed.ToString();
            textBox5.Text = texter;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var sw = new Stopwatch();
            sw.Start();
            string myArr = textBox3.Text;
            int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) -
48).ToArray();
            string texter = string.Join("", CoctailSort.SortArray(result));
            textBox5.Text = texter;
            sw.Stop();
            textBox7.Text = sw.Elapsed.ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var sw = new Stopwatch();
            sw.Start();
```

```
            string myArr = textBox3.Text;
            int[] result = myArr.ToString().Select(o => Convert.ToInt32(o) -
48).ToArray();
            int resl = result.Length;
            string texter = string.Join("", GnomeSort.SortArray(result));
            sw.Stop();
            textBox4.Text = sw.Elapsed.ToString();
            textBox5.Text = texter;
        }


    }
}
```