ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Существует множество различных средств разработки программного обеспечения. Для программы выбрано использование «Visual Studio» с разработкой .NET, что позволяет сделать наглядный интерфейс, который будет прост в использовании.

Целью практики является формирование и развитие общих и профессиональных компетенций по модулю ПМ.03 Участие в интеграции программных модулей.

Задачами учебной практики являются:

- закрепление навыков разработки программного обеспечения;
- использование методов для получения кода с заданной функциональностью;
- разработка документации на программный продукт;
- знание моделей процесса разработки программного обеспечения;
- знание основных подходов к интегрированию программных модулей;
- знание основных методов и средств эффективной разработки программного обеспечения.

Visual Studio – имеет в себе интегрированную среду разработки программного обеспечения, а также другие инструменты. Позволяют создавать как консольные приложения, так и игры и приложения с графическим интерфейсом.

.NET Framework - программная платформа, основой которой является общеязыковая среда исполнения. Является модульной платформой для разработки программного обеспечения с открытым кодом.

Язык программирования – формализованный язык для описания алгоритма решения задачи на компьютере.

C# - объекто-ориентированный язык программирования общего назначения. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы и т.д.

1. Техническое задание на разработку программы

1. Введение

Название программы: «Сборник сортировок».

Целью создания программы является разработка продукта для сортировки массивов целых чисел в порядке возрастания.

Программа содержит в себе следующие алгоритмы сортировок:

- сортировка пузырьком;
- сортировка вставками;
- сортировка перемешиванием;
- сортировка расческой (CombSort).

Интерфейс программного продукта сделан в соответствии требованиям, выполнен благодаря Microsoft .NET Framework.

2. Требования к программному продукту

Программа должна обеспечивать возможность ввода и вывода данных непосредственно в программе, поддерживать не менее четырех различных методов сортировок, иметь понятный пользователю графический интерфейс.

Для каждого метода должен быть разработан соответствующий алгоритм.

3. Технические требования к программному продукту

Программный продукт должен быть написан на языке C# и работать на операционных системах Windows.

Интерфейс программы выполнен в WindowsForms благодаря .NET Framework. Должен иметь: поля для ввода/вывода, поля для выбора необходимой сортировки и функциональные кнопки.

4. Технико-экономические показатели

Экономическая эффективность не рассчитывается. Аналогия не проводится в виду необходимости разработки программного продукта с представленными требованиями.

5. Приемо-сдаточные испытания

При приемо-сдаточных испытания разрабатываемый программный продукт должен быть полностью готов.

Сами испытания проводятся на мощностях заказчика в оговоренные сроки.

Приемка должна проводиться согласно разработанной исполнителем и согласованной заказчиком программы.

2. Спецификация программного продукта

Сортировка – это упорядочение чисел по возрастанию или убыванию их значений.

Сборник сортировок – это программа, выполняющая сортировку по возрастанию входных данных, с возможностью выбора одного из четырех методов сортировок. Основные функции: возможность ввода количества элементов массива, сток чисел, а также возможность выбора сортировки.

Описание сортировки методом пузырька: будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Не более чем после n итераций массив будет отсортирован. Метод пузырька может быть достаточно длительным, если самые маленькие значения будут стоять в конце массива данных.

Описание сортировки вставками: создадим массив, в котором после завершения алгоритма будет лежать ответ. Будем поочередно вставлять элементы из исходного массива так, чтобы элементы в массиве-ответе всегда были отсортированы. Чтобы отсортирован был некоторый префикс исходного массива, вместо вставки будем менять текущий элемент с предыдущим, пока они стоят в неправильном порядке.

Описание сортировки перемешиванием: будем идти не только слева направо, но и справа налево. Будем поддерживать два указателя begin и end, обозначающих, какой отрезок массива еще не отсортирован. На очередной итерации при достижении end вычитаем из него единицу и движемся справа налево, аналогично, при достижении begin прибавляем единицу и двигаемся слева направо.

Описание сортировки расческой: будем переставлять элементы, стоящие на расстоянии. Зафиксируем его и будем идти слева направо, сравнивая

элементы, стоящие на этом расстоянии, переставляя их, если необходимо. Оптимально изначально взять расстояние равным длине массива, а далее делить его на некоторый коэффициент. Когда расстояние станет равно единице, выполняется сортировка пузырьком.

Алгоритм сортировки принимает исключительно целые числа, записанные через пробел, без знаков препинания.

3. Диаграммы программного продукта

Диаграмма потоков программного продукта представлена на рисунке 1.

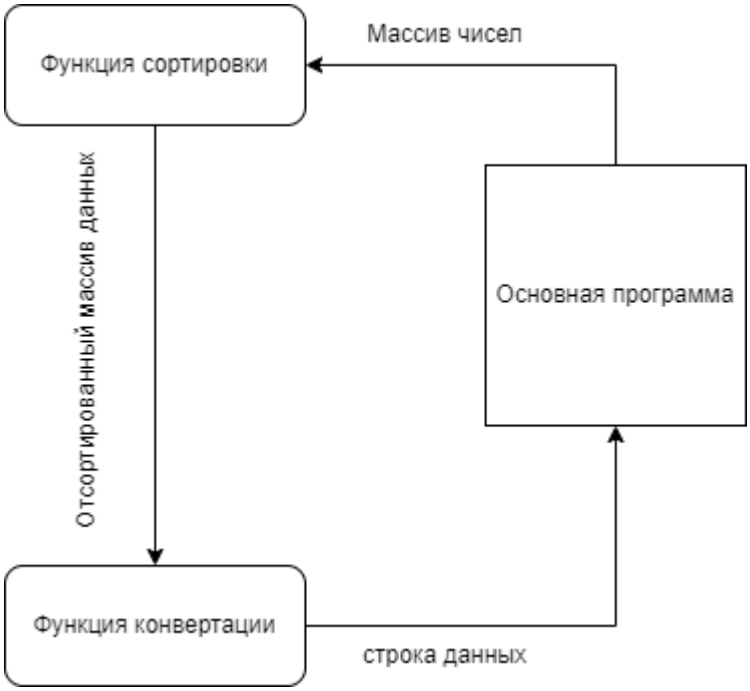

Рисунок 1 – диаграмма потоков.

Функциональная диаграмма программного продукта представлена на рисунке 2.

Рисунок 2 – функциональная диаграмма.

Функциональная схема программного продукта представлена на рисунке 3.

Рисунок 3 – функциональная диаграмма.

4. Блок-схемы программного продукта

Блок-схема — распространённый тип схем, описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности.

Блок схема сортировки методом пузырька представлена на рисунке 4.

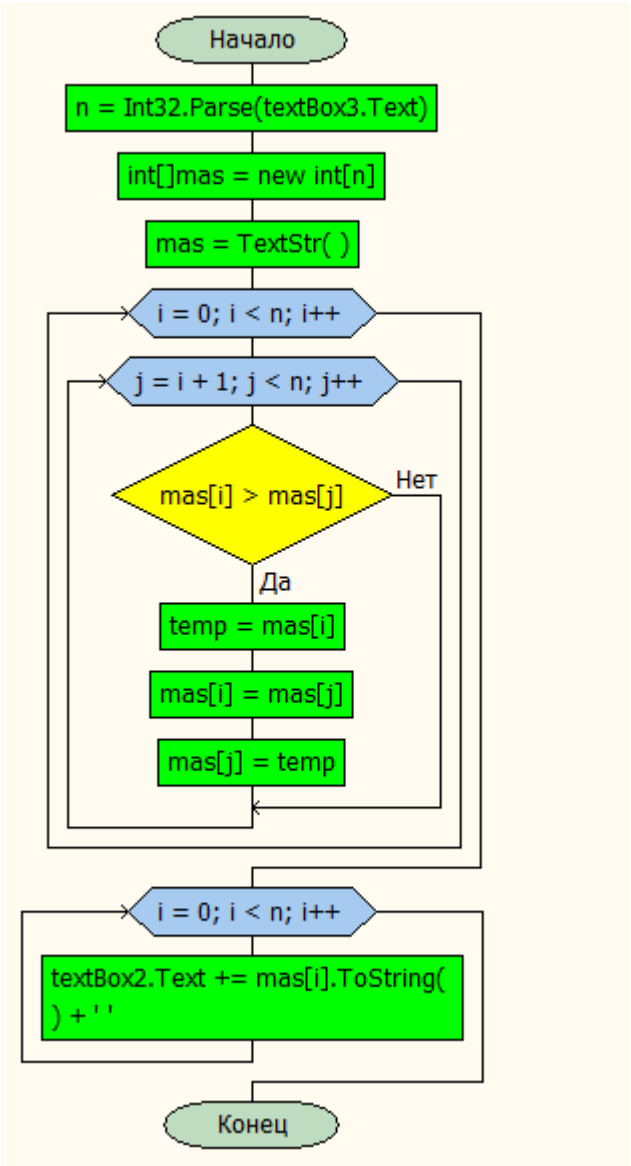

Рисунок 4- сортировка пузырьком

Схема сортировки вставками представлена на рисунке 5.

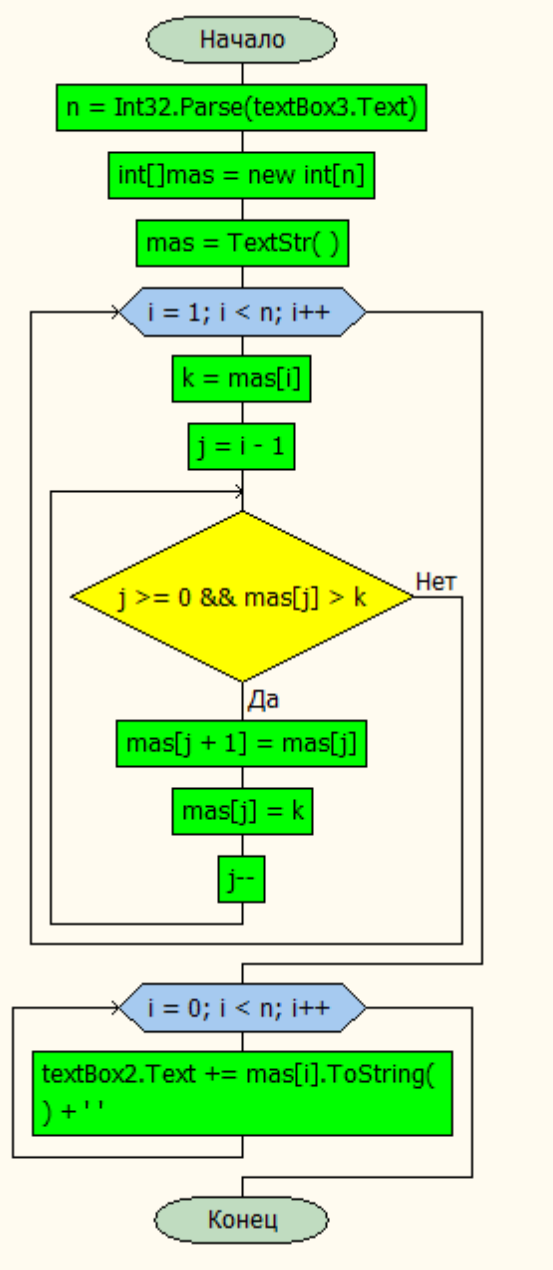

Рисунок 5 – метод сортировки вставками

Блок-схема метода сортировки перемешиванием представлена на рисунке 6.

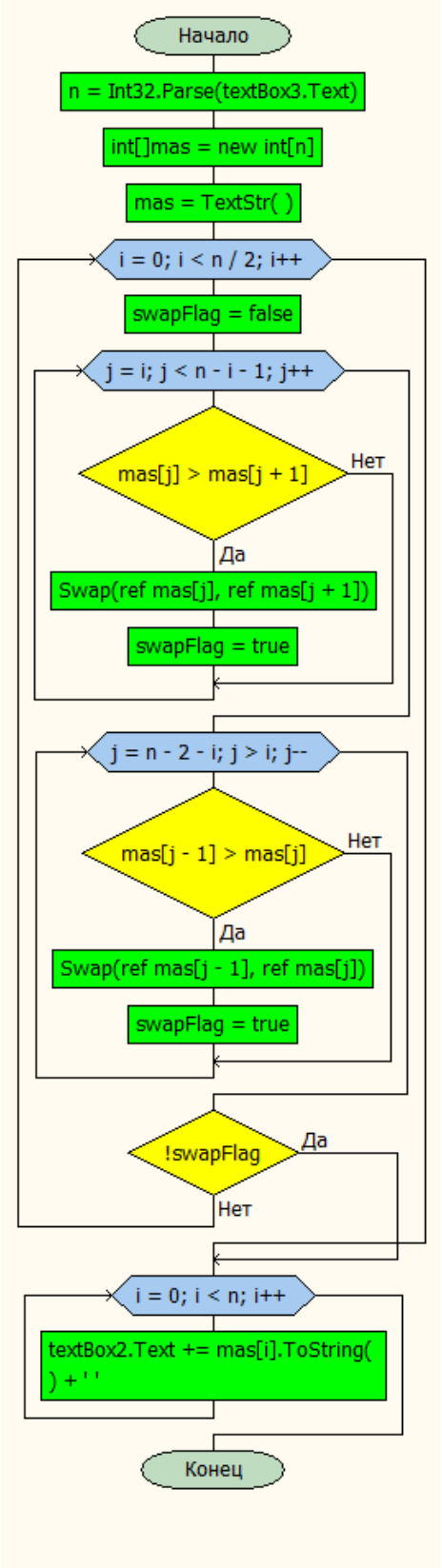

Рисунок 6 – сортировка перемешиванием

Схема сортировки расческой представлена на рисунке 7.

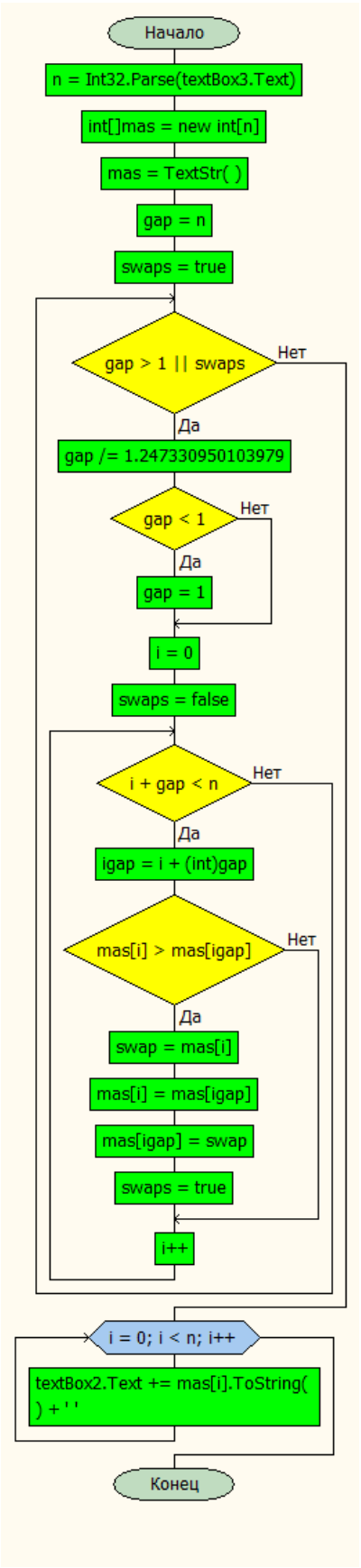

Рисунок 7 – Сортировка расческой

5. Разработка кода программных модулей

Разработка кода алгоритмов сортировки начинается с понимания основной идеи каждого из них и описания его шагов в виде блок-схем. После этого, схемы переводятся в код на выбранном языке программирования, мы используем C#.

Код для каждого алгоритма сортировки разрабатывается поэтапно, пройдя каждый шаг по схеме и преобразуя его в код. Обычно, этот процесс включает в себя следующие этапы:

- Создание функции с именем алгоритма, в нашем случае это: bubble, cocktail, combsort и insertion.
- Определение входных параметров функции.
- Написание кода для цикла, который проходит через элементы массива.
- Определение логики, выполняемой на каждой итерации цикла в соответствии со схемой алгоритма.
- Проверка кода на правильность работы.
- Если возникают ошибки или недочеты, то их устранение и повторный запуск кода.
- Тестирование функции на разных типах данных и массивах различной длины, чтобы убедиться в ее корректности и эффективности.

При разработке кода алгоритмов сортировки, очень важно понимать каждый из них и четко следовать схеме, чтобы не допустить ошибок в коде и обеспечить эффективность работы функции.

6. Пользовательский интерфейс

Интерфейс программного продукта реализован благодаря .NET Framework. Выглядит он следующим образом:

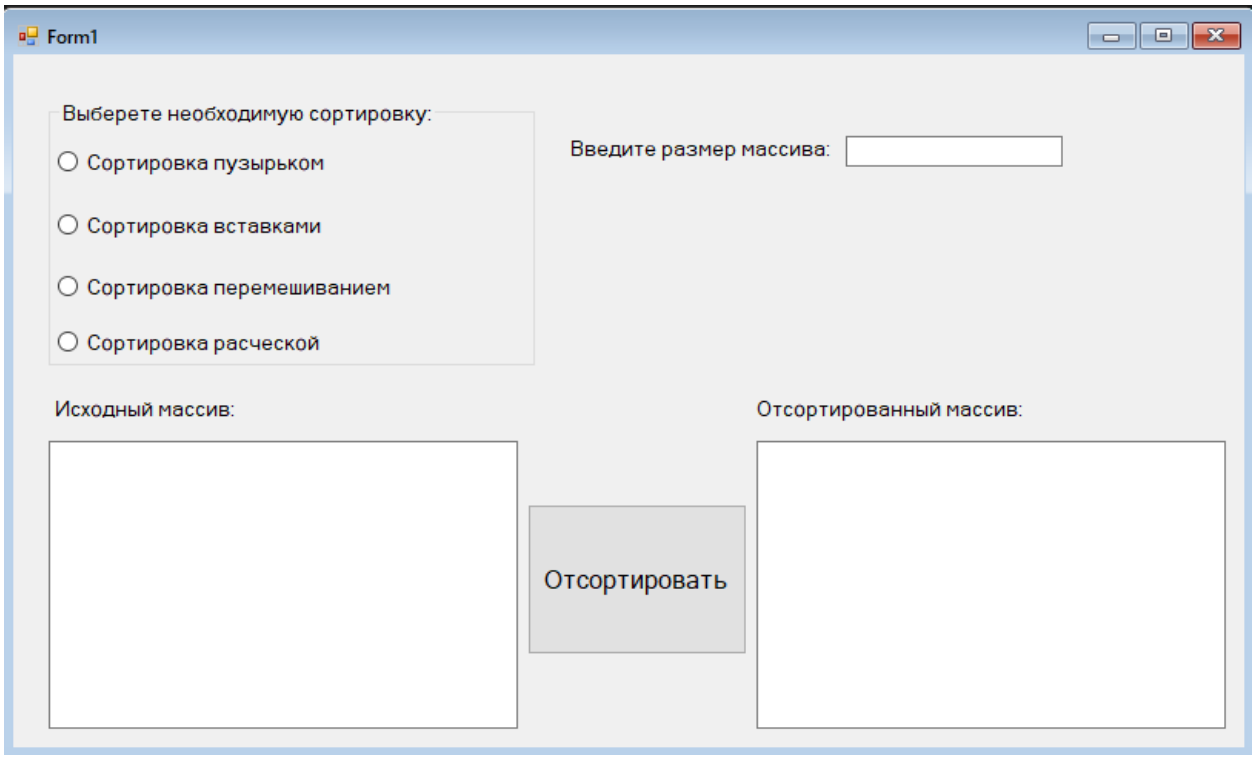

Рисунок 8 – интерфейс программы.

Исходным массив, отсортированный массив и размер массива – элементы графического интерфейса, позволяющие пользователю ориентироваться в программе.

Отсортировать – это функциональная кнопка, запускающая процесс сортировки.

Примеры элементов интерфейса, которые можно использовать для алгоритмов сортировки, включают в себя кнопки для запуска алгоритмов и поля ввода для задания массива элементов.

При разработке пользовательского интерфейса для алгоритмов сортировки важно следить за тем, чтобы он был интуитивно понятен для

пользователей и позволял им удобно взаимодействовать с функциями сортировки.

Минималистичность пользовательского интерфейса программы обусловлены возможностями среды разработки Visual Studio.

7. Интеграция модулей в программный продукт

Интеграция программных модулей в программный продукт может выполняться различными способами, в зависимости от используемой архитектуры приложения и специфики разрабатываемого программного продукта:

- создание отдельных файлов с кодом для алгоритмов сортировки;
- включение этих файлов в проект;
- использование функций или методов из файла в других частях программы;
- использование модификаторов доступа к этим функциям или методам, чтобы ограничить их использование только внутри этого модуля.

Интеграция программных модулей должна быть выполнена таким образом, чтобы она соответствовала принципам хорошего программирования, таким как модульность, читаемость и повторное использование кода.

8. Тестирование программного продукта

Для проведения процедур тестирования в полной мере необходимо выполнить следующие шаги:

- unit-тесты для каждого метода сортировки, которые проверяют корректность работы алгоритмов;
- проверка производительности алгоритмов поможет вычислить эффективность каждой сортировки;
- интеграционное тестирование. Необходимо протестировать работоспособность интерфейса программы, все ли выполняется согласно указанным требованиям.

Протокол тестирования программного продукта по сортировкам:
Дата тестирования: 21.03.2023
Тестируемый продукт: SortApp
Тестировщик: Горбач Анастасия Алексеевна
Цель тестирования: проверка корректности работы алгоритмов сортировки в программном продукте.

- Юнит-тестирование алгоритмов сортировки

1. Тест сортировки пузырьком:
   Входные данные: [3, 2, 1, 5, 4, 8, 7, 6]
   Ожидаемый результат: [1, 2, 3, 4, 5, 6, 7, 8]
   Результат теста: пройден.
2. Тест сортировки вставками:
   Входные данные: [4, 7, 2, 5, 1, 6, 3]
   Ожидаемый результат: [1, 2, 3, 4, 5, 6, 7]
   Результат теста: пройден.
3. Тест сортировки перемешиванием:

Входные данные: [5, 6, 3, 2, 4, 1]

Ожидаемый результат: [1, 2, 3, 4, 5, 6]

Результат: пройден.

4. Тест сортировки расческой:

Входные данные: [7, 2, 3, 5, 1, 4, 6]

Ожидаемый результат: [1, 2, 3, 4, 5, 6, 7]

Результат: пройден.

– Тестирование производительности алгоритмов

1. Тест производительности сортировки пузырьком на массиве из 100000 элементов:

Время выполнения: 23.29 секунд

Результат: неудовлетворительный.

2. Тест производительности сортировки вставкой на массиве из 100000 элементов:

Время выполнения: 7.82 секунды

Результат: удовлетворительный.

3. Тест производительности сортировки перемешиванием на массиве из 100000 элементов:

Время выполнения: 10.97 секунды

Результат: удовлетворительный.

4. Тест производительности сортировки расческой на массиве из 100000 элементов:

Время выполнения: 4.56 секунды

Результат: отличный.

– Интеграционное тестирование для каждой из сортировок пройдена на отлично.

Итоговый результат: все тесты пройдены успешно. Программный продукт по сортировкам на языке C# интегрирован и готов к использованию.

9. Руководство пользователя

Программный продукт содержит четыре различных метода сортировки на языке программирования C#: сортировка пузырьком, сортировка вставками, сортировка перемешиванием и сортировка расческой. Пользователю предоставлен выбор между этими сортировками, а также доступен ввод набора данных для сортировки. После сортировки программа отображает полученный массив данных.

При запуске программы пользователю будет доступен весь необходимый функционал. Интерфейс содержит выбор нужной сортировки, место для ввода массива данных, который требуется отсортировать, а также место для вывода отсортированных данных.

Пользователь выбирает метод сортировки, после чего указывает размер массива, который будет необходимо отсортировать. Затем вводятся значения без запятых, через пробел. А дальше нажав на функциональную кнопку отсортировать, пользователь получает отсортированный массив данных.

Инструкция по использованию:

- запустите программу;
- выберите необходимый метод сортировки;
- назначьте размерность массива данных;
- введите набор значений через пробел, без запятых и иных знаков препинания;
- нажмите на кнопку «отсортировать»;
- программа отобразит отсортированный массив данных;
- если пользователь не ввел значения или не выбрал метод сортировки, программа отобразит сообщение об ошибке.

ЗАКЛЮЧЕНИЕ

В ходе учебной практики были выполнены следующие задачи:

- разработано техническое задание для программного продукта. Это помогло лучше определиться с назначением программы в целом;
- разработана спецификация на программный продукт;
- составлены диаграммы программного продукта;
- разработаны блок-схемы программного продукта, позволяющие лучше понять организацию процесса алгоритмов;
- написан код программного модуля;
- создан понятный пользователю графический интерфейс;
- выполнена интеграция модулей в программном продукте;
- проведена процедура тестирования программного продукта;
- создано руководство пользователя;
- проект был загружен на GitHub (https://github.com/Anisimko/Praktic.git).

Проделанная работа укрепляет знания в сфере программирования и разработки программного обеспечения. Помогает лучше понимать работу программных продуктов. Позволяет укрепить навыки понимания моделей процесса разработки программного обеспечения, подходов к интегрированию программных модулей, основных принципов процесса разработки программных модулей.

В итоге все поставленные задачи были выполнены в полной мере, цель учебной практики успешно достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОНИКОВ

1. Все об алгоритмах // Хабр URL: https://habr.com/ru/hub/algorithms/ (дата обращения: 20.03.2023).
2. Алгоритмы // СкиллФактори URL: https://blog.skillfactory.ru/glossary/algoritm/ (дата обращения: 20.03.2023).
3. Алгоритмы и структуры данных // Octus URL: https://otus.ru/lessons/algorithm/ (дата обращения: 21.03.2023).
4. GitHub - документация URL: https://github.com/Anisimko/Praktic.git (дата обращения: 20.03.2023).

ПРИЛОЖЕНИЕ А

1. Сортировка пузырьком:

```
private void bubble()
      {
         int n = Int32.Parse(textBox3.Text);
         int[] mas = new int[n];
         mas = TextStr();
         int temp;
         for (int i = 0; i < n; i++)
         {
            for (int j = i + 1; j < n; j++)
            {
               if (mas[i] > mas[j])
               {
                  temp = mas[i];
                  mas[i] = mas[j];
                  mas[j] = temp;
               }
            }
                     }
                     for (int i = 0; i < n; i++)
                     {
                     textBox2.Text += mas[i].ToString()+' ';
                     }
           }
```

2. Сортировка вставками:

```
private void insertion()
      {
         int n = Int32.Parse(textBox3.Text);
         int[] mas = new int[n];
         mas = TextStr();
         for (int i = 1; i < n; i++)
         {
            int k = mas[i];
            int j = i - 1;
```

```
            while (j >= 0 && mas[j] > k)
            {
                mas[j + 1] = mas[j];
                mas[j] = k;
                j--;
            }
        }
        for (int i = 0; i < n; i++)
        {
            textBox2.Text += mas[i].ToString() + ' ';
        }
    }
```

3. Сортировка перемешиванием:

```
private void cocktail()
    {
        int n = Int32.Parse(textBox3.Text);
        int[] mas = new int[n];
        mas = TextStr();

        for (var i = 0; i < n / 2; i++)
        {
            var swapFlag = false;

            for (var j = i; j < n - i - 1; j++)
            {
                if (mas[j] > mas[j + 1])
                {
                    Swap(ref mas[j], ref mas[j + 1]);
                    swapFlag = true;
                }
            }
            for (var j = n - 2 - i; j > i; j--)
            {
                if (mas[j - 1] > mas[j])
                {
                    Swap(ref mas[j - 1], ref mas[j]);
                    swapFlag = true;
                }
```

```
            }
            if (!swapFlag)
            {
                break;
            }
        }
        for (int i = 0; i < n; i++)
        {
            textBox2.Text += mas[i].ToString() + ' ';
        }
    }
```

4. Сортировка расческой:

```
private void combsort()
    {
        int n = Int32.Parse(textBox3.Text);
        int[] mas = new int[n];
        mas = TextStr();
        double gap = n;
        bool swaps = true;
        while (gap > 1 || swaps)
        {gap /= 1.247330950103979;
            if (gap < 1) { gap = 1; }
            int i = 0;
            swaps = false;
            while (i + gap < n)
            {
                int igap = i + (int)gap;
                if (mas[i] > mas[igap])
                {
                    int swap = mas[i];
                    mas[i] = mas[igap];
                    mas[igap] = swap;
                    swaps = true;
                }
                i++;
            }
        }
        for (int i = 0; i < n; i++)
        {
            textBox2.Text += mas[i].ToString() + ' ';
        }
    }
```