Изображение выглядит как Цвет электрик

Автоматически созданное описание

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное учреждение  высшего образования | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Департамент программной инженерии и искусственного интеллекта** | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ГРИШКОВ ЛЕОНИД ПЕТРОВИЧ  ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ МЕТОДОВ АППРОКСИМАЦИИ ВЕЩЕСТВЕННЫХ ЧИСЕЛ ДЛЯ НАСТРОЙКИ ГИТАРЫ ДИФФЕРЕНЦИАЛА ЗУБОФРЕЗЕРНЫХ СТАНКОВ | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| по дисциплине «Структуры и алгоритмы компьютерной обработки данных»  по образовательной программе подготовки бакалавров по направлению  02.03.03 – Математическое обеспечение и администрирование информационных систем, профиль «Технология программирования» | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | |  |  | | Студент гр. Б9122 | | | |  | | | | Гришков Л.П. | | | | |
|  |  | |  | | | | (подпись) | | | |  | | | | |
|  | | | | | | | | | | |  |  | | Руководитель доцент департамента ПИИИ | | | | | | | | | | |  | |
|  | | | | | | | | | | |  |  | | к.т.н., |  | | | | | | С.Н. Остроухова | | | | | |
|  | | | | | |  |  | | | |  |  | |  | (подпись) | | | | | |  | |  | | | |
|  |  |  |  | | | | | |  | |  | |  |  | | «\_\_» | |  | | | | | | 2024 г. | |
|  | | | | | | | | | | | | | | (подпись) | |  |  | |  |  | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Защищен оценкой | | | | | | | |  | | | | |  | | | | | | | | | | | | | |
|  | | | |  |  | | | | | | | |  | | | | | | | | | | | | | |
| (подпись) | | | |  | Фамилия И.О. | | | | | | | |  | | | | | | | | | | | | | |
| «\_\_\_\_» | | |  | | | | | | | 2024 г. | | | | | | | | | | | | | | | | |
|  | | |  | | | | | | |  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2024 | | | | | | | | | | | | | | | | | | | | | | | | | | |

Оглавление

[Введение 3](#_Toc140664346)

[1 Алгоритмы быстрой и интроспективной сортировок 5](#_Toc140664347)

[1.1 Основные определения и понятия 5](#_Toc140664348)

[1.2 Описание быстрой сортировки 5](#_Toc140664349)

[1.3 Анализ алгоритма быстрой сортировки 7](#_Toc140664350)

[1.4 Описание интроспективной сортировки 7](#_Toc140664351)

[1.5 Анализ алгоритма интроспективной сортировки 9](#_Toc140664352)

[2 Описание компьютерной программы 10](#_Toc140664353)

[2.1 Требования к генерации 10](#_Toc140664354)

[2.2 Требования к эксперименту 12](#_Toc140664355)

[2.3 Архитектура программного средства 13](#_Toc140664356)

[2.4 Сценарий диалога 15](#_Toc140664357)

[3 Описание эксперимента 19](#_Toc140664358)

[3.1 Эксперименты над быстрой и интроспективной сортировками 19](#_Toc140664359)

[Заключение 22](#_Toc140664360)

[Список литературы и интернет-источников 23](#_Toc140664361)

# Введение

Подбор шестерен гитары дифференциала является важным этапом настройки зубофрезерных станков, от которого зависит как точность изготавливаемых деталей, так и скорость износа оборудования.

В техническом паспорте зубофрезерных станков из-за большого количества возможных комбинаций не указываются наборы шестерен гитары дифференциала для настройки необходимых параметров детали, а в таблицах для подбора шестерен по передаточному числу не учитываются как геометрические ограничения отдельного станка, так и наличие шестерен с необходимым количеством зубьев.

Отсутствие на производстве таблиц для настройки гитар дифференциала зубофрезерных станков ведет к существенному увеличению длительности этапа их настройки и, как следствие, к снижению эффективности производства и труда, что негативно сказывается на всех участниках производства.

**Цель:** провести сравнительный анализ трех методов аппроксимации передаточного числа гитары дифференциала станка 5К310: полный перебор, метод ветвей и границ, а также жадный алгоритм перебора по передаточному числу одной из пар шестерен

**Задачи:**

1. Обзор алгоритмов быстрой и интроспективной сортировок с помощью литературных источников, анализ особенности каждой сортировки, выявление критериев их производительности.
2. Разработка программного продукта, позволяющего провести экспериментальные исследования быстрой и интроспективной сортировок. Программный продукт должен включать в себя такие функции, как: генерация данных, визуализация данных и анализ временной сложности алгоритмов.
3. Проведение экспериментов с использованием программного продукта для оценки производительности быстрой и интроспективной сортировок. Измерение временной сложности алгоритмов, построение графиков зависимости между входными данными и временем их упорядочивания.
4. Анализ результатов эксперимента, определение эффективности быстрой и интроспективной сортировок для различных наборов данных. Сравнение производительности обоих алгоритмов с точки зрения временной сложности.

# Алгоритмы быстрой и интроспективной сортировок

Сочетание существующих сортировок позволяет создавать новые, более оптимальные сортировки, которые по своей быстроте превосходят старые и при этом не требуют больше памяти. Так в этой главе будет рассмотрена одна из гибридных сортировок – интроспективная, а также быстрая сортировки.

## Основные определения и понятия

***Алгоритм* –** это последовательность команд, предназначенная исполнителю, в результате выполнения которой он должен решить поставленную задачу.

***Сортировка* –** процесс упорядочивания множества элементов по какому-то признаку.

***Устойчивая сортировка* –** сортировка, в процессе которой относительное расположение элементов с равными ключами не изменяется.

***Метод* –** это совокупность рациональных действий, которые необходимо предпринять, чтобы решить определённую задачу или достичь определённой цели.

***Массив* –** пронумерованная последовательность величин одинакового типа, обозначаемая одним именем.

***Рекурсивный объект (рекурсия)*** – объект, частично состоящий или определяемый с помощью самого себя.

## Описание быстрой сортировки

[Быстрая сортировка](https://wiki2.org/ru/Быстрая_сортировка) — это алгоритм сортировки, базовая версия которого была придумана в 1960 г. Чарльзом Хоаром. Суть его предельно проста: выбирается так называемый опорный элемент, и массив делится на 3 подмассива: меньших опорного, равных опорному и больших опорного. Потом этот алгоритм применяется рекурсивно к подмассиву меньших и больших опорных.

Быстрая сортировка стала популярной прежде всего потому, что ее нетрудно реализовать, она хорошо работает на различных видах входных данных и во многих случаях требует меньше затрат ресурсов по сравнению с другими методами сортировок.

Общая идея алгоритма состоит в следующем:

1. Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
2. Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».
3. Для отрезков «меньших» и «больших» значений выполнить рекурсивно 2 пункт, если длина отрезка больше единицы.

На практике массив обычно делят не на три, а на две части: например, «меньшие опорного» и «равные и большие»; такой подход в общем случае эффективнее, так как упрощает алгоритм разделения. Псевдокод алгоритма быстрой сортировки показан на рисунке 1.

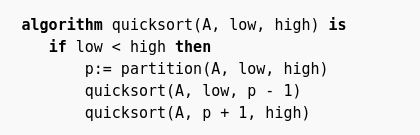
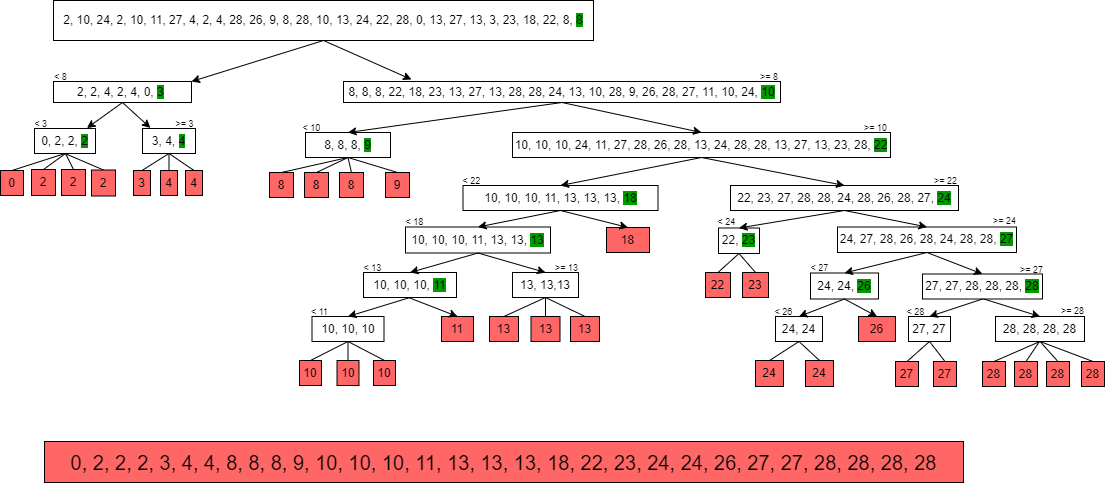


Рисунок 1. Псевдокод алгоритма быстрой сортировки

Пример работы быстрой сортировки, где опорным является последний элемент массива, представлен ниже.



## Анализ алгоритма быстрой сортировки

Алгоритм быстрой сортировки обладает весьма привлекательными особенностями: он принадлежит к категории обменных (in-place) сортировок, на выполнения сортировки **N** элементов в среднем затрачивается время, пропорциональное **N log N** и для него характерны исключительно короткие внутренние циклы.

Его недостатком является то, что он неустойчив, для его выполнения в наихудшем случае требуется **N2** операций, он хрупок в том смысле, что даже простая ошибка в реализации может пройти незамеченной и вызвать ошибки в работе алгоритма на некоторых видах файлов.

## Описание интроспективной сортировки

Несмотря на всю эффективность метода быстрой сортировки, при неудачном подборе входных данных сортировка сама себя вызывает максимальное количество раз, а рекурсия – процесс дорогостоящий для памяти. Поэтому в 1997 году Дэвидом Мюссером была придумана *Introsort*, или интроспективная сортировка, принадлежащая классу гибридных сортировок.

Предложенный алгоритм интроспективной сортировки использует быструю сортировку и переключается на пирамидальную сортировку, когда глубина рекурсии превышает некоторый заранее установленный уровень, чтобы избежать худший случай ***O(n2)***. Так как оба алгоритма используют сравнения, этот алгоритм также принадлежит классу сортировок на основе сравнений. Если количество элементов в последовательности менее 16, то принято сортировать ее простыми вставками.[4]

Псевдокод алгоритма интроспективной сортировки представлен на рисунке 2.

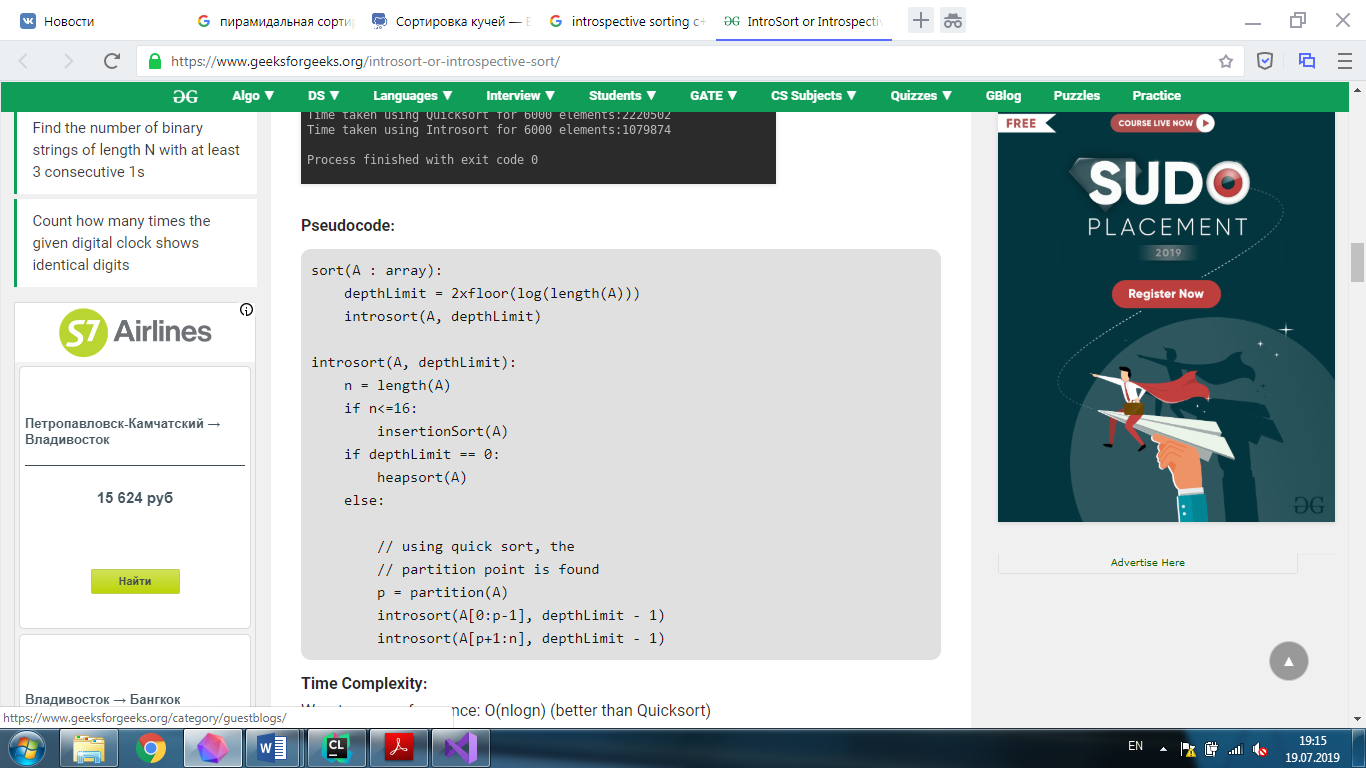


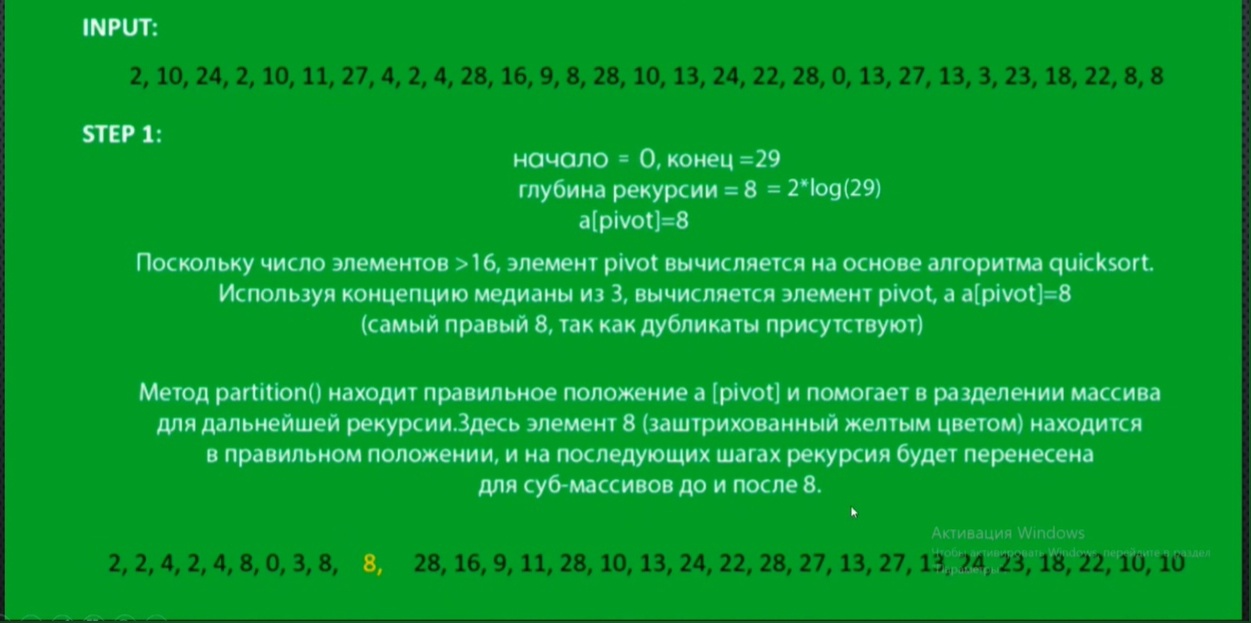
Рисунок 2. Псевдокод алгоритма интроспективной сортировки

Пример работы интроспективной сортировки, в которой если глубина рекурсии превышает некоторый заранее установленный уровень, то сортировка переключается на сортировку слиянием, представлен ниже. В данном примере опорным является последний элемент массива.

Шаг 1: начало = 0| конец = 29| глубина рекурсии = 8| a[pivot] = 8

Поскольку число элементов >16, элемент pivot вычисляется на основе алгоритма быстрой сортировки, используя концепцию медианы из 3, вычисляем a[pivot] = 8 (самый правый)

Шаг 2:

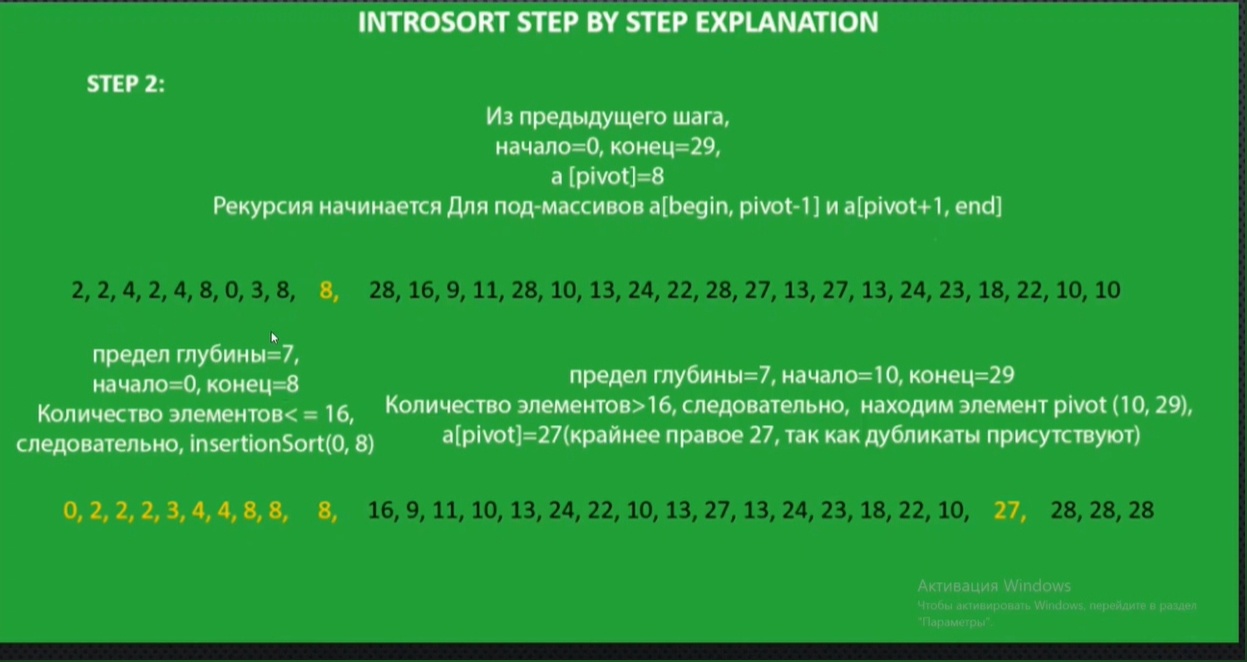


Предел глубины = 7 | Предел глубины = 7

Количество эл-тов <= 16 | Количество эл-тов <= 16

a[begin, pivot-1] | a[pivot+1, end]

Интроспективная сортировка (0,8) | a[pivot] = 27 (крайнее правое)



Шаг 3:

Рекурсия начинается для вложенных массивов a[begin, partition-1] и a[partition+1, end]

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рассортирован | интроспективная сортировка (10, 25) | интроспективная сортировка(27, 29)



Рассортирован | Рассортирован | Рассортирован

## Анализ алгоритма интроспективной сортировки

Подход интроспективной сортировки сочетает в себе достоинства обоих методов (быстрой и пирамидальной) с худшим случаем ***O****(****n\*logn****)* и быстродействием, сравнимым с быстрой сортировкой. В лучшем случае интроспективная сортировка сортирует за ***O****(****n).*** [4]

# Описание компьютерной программы

В данной главе будет рассмотрена компьютерная программа для проведения экспериментов c быстрой и интроспективной сортировками. Компьютерная программа решает такие задачи, как генерация файлов с возможностью подбора типов последовательностей и задания особенностей содержащихся в них элементов, применение вышеупомянутых алгоритмов для сортировки данных из ранее сгенерированных файлов, фиксирование времени их работы и построение гистограмм, демонстрирующих влияние различных параметров на производительность работы алгоритмов сортировки.

## Требования к генерации

1. Требования к входным данным.
   * 1. *N* – количество целых чисел в последовательности*,*

*N* ∈ [1 .. 1000000]

* + 1. *Type* – тип последовательности:
* Упорядоченная возрастающая;
* Упорядоченная убывающая;
* Частично-упорядоченная по возрастанию;
* Случайная;
* Состоящая из повторяющегося одного элемента.

1. Требования к выходным данным.

F – файл, удовлетворяющий следующим критериям:

* Файл имеет уникальное имя вида: <Имя>.<Расширение>;   
  <Имя> ::= <Sequence><N>;  
  <Расширение> ::= <Type>;

<Sequence> ∈ {up, down, similar, random, half};  
<N> – целое число из диапазона [1; 1000000];  
<type> ∈ {txt}.

* В файле должна быть записана ровно одна последовательность типа *Type*. Последовательность содержит *N* элементов, при этом все элементы принадлежат целочисленному типу. Элементы записаны в одну строку, а в качестве разделяющего элемента выступает пробел.

1. Функциональные требования

* КП должна позволять:

а) Вводить количество элементов в последовательности;

б) Выбирать один из четырех типов последовательностей;

* КП должна создавать файл и генерировать его содержимое в соответствии с заданными входными данными;
* КП должна размещать созданный файл в директории проекта;
* КП должна считать количество сравнений, проведенных каждой из сортировок в процессе работы.
* КП должна считать количество перестановок, проведенных каждой из сортировок в процессе работы.
* КП должна засекать время работы каждой из сортировок над конкретной последовательностью элементов.
* КП должна строить диаграммы на основании множества выходных данных.
* КП должна при создании нового файла проверять наличие файла с таким же названием и, если он существуют, то перезаписывать этот файл.
* КП должна проверять корректность введенных данных:
  + КП должна проверять, чтобы введенное пользователем *N* соответствовало требуемому диапазону;
  + КП должна проверять, чтобы был выбран один из типов последовательностей;
* КП должна сообщать пользователю об ошибке в случае некорректности введенных данных;

## Требования к эксперименту

1. Входные данные

FN – множество имен файлов F, удовлетворяющих критериям, указанным в пункте 2.1 – B.

1. Выходные данные
2. FN – Множество файлов удовлетворяющие следующим условиям:
   * Каждый FN файл имеет уникальное имя <I><N>. <T>;
   * N ∈ [2;1000000];
   * I ∈ {up\_sorted, down\_sorted, similar\_sorted, random\_sorted, half\_sorted};
   * T ∈ {txt};
   * Каждый n-ый файл содержит ровно две строки:
     1. прописывается N, т.е количество элементов числовой последовательности
     2. прописывается отсортированная последовательность из входного файла.
3. Гистограммы, отображающие:

* зависимость времени работы алгоритма сортировки
* зависимость количества сравнений от количества элементов
* зависимость количества обменов от количества элементов

1. Функциональные требования
   * + КП должна позволять выбирать файлы для эксперимента в директории компьютера;
     + КП должна, выполняя алгоритмы быстрой и интроспективной сортировок, упорядочить последовательности, содержащиеся в выбранных файлах;
     + КП должна засекать время работы быстрой и интроспективной сортировок и считать количество перестановок, а затем по полученным данным строить соответствующие гистограммы.

## Архитектура программного средства

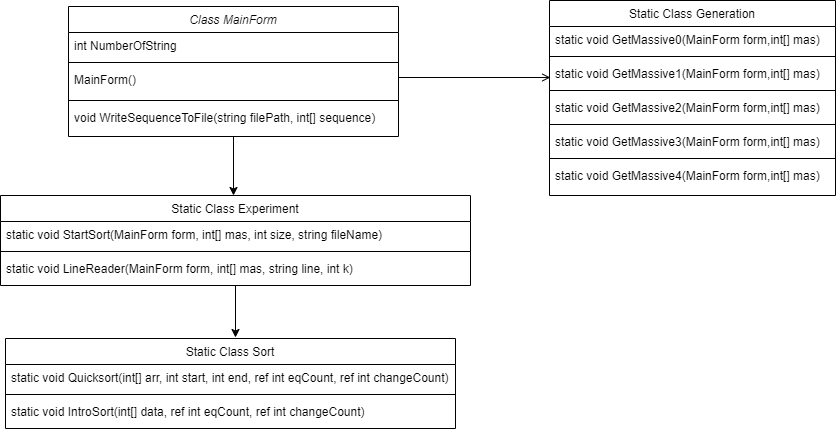


Рисунок 3. Архитектурный проект

Архитектурный проект, реализовывающий проведение эксперимента между быстрой и интроспективной сортировками, представлен на рисунке 3. Для реализации программы используются следующие классы:

1. *MainForm* – класс, наследник от класса экранной формы, реализующий главное окно программы, через которое генерируются файлы последовательностей, а так же обрабатываются события пользовательских действий (нажатия кнопок, выбор элементов и т.д.)
2. *Generation* – статический класс, реализующий методы генерации различных числовых последовательностей. Методы этого класса, вызываемые из методов класса MainForm, проверяют корректность данных и на основе переданных параметров (характеристик последовательности) производят генерацию. Имеет следующие публичные методы:
   * *GetMassive0* – создает файл, содержащий в себе упорядоченную по возрастанию последовательность целых чисел размера *N*.
   * *GetMassive1* – создает файл, содержащий в себе упорядоченную по убыванию последовательность целых чисел размера *N*.
   * *GetMassive2* – создает файл, содержащий в себе последовательность из *N* целых одинаковых чисел.
   * *GetMassive3* – создает файл, содержащий последовательность из *N* чисел, расположенных в произвольном порядке.
   * *GetMassive4* – создает файл, содержащий в себе N целых чисел, которые представляют последовательность, состоящую из частично отсортированной последовательности по возрастанию.
3. *Experiment* – статический класс, реализующий считывание последовательностей из файлов и затем, при запуске методов статического класса Sorts, подсчёт времени работы и количества пересылок.
4. *Sort­­* – статистический класс, реализующий оба метода сортировки:

* static void Quicksort(int[] arr, int start, int end, ref int eqCount, ref int changeCount) запускает на числовой последовательности *arr* быструю сортировку, разделяя последовательность на две части вокруг опорного элемента (последний элемент) и рекурсивно сортирует каждую из них. При сравнении элементов массива количество сравнений увеличивается на 1 (eqCount), а количество перестановок - на 1 (changeCount).
* static void IntroSort(int[] data, ref int eqCount, ref int changeCount) запускает на числовой последовательности *data* интроспективную сортировку, начиная с использования быстрой сортировки, и при превышении предела глубины рекурсии, переключая на сортировку кучей (HeapSort) или сортировку вставками (InsertionSort) в зависимости от размера подмассива.

## Сценарий диалога

При запуске программы появляется основное окно «Экспериментальное исследование методов быстрой и интроспективной сортировок», представленное на рисунке 4.

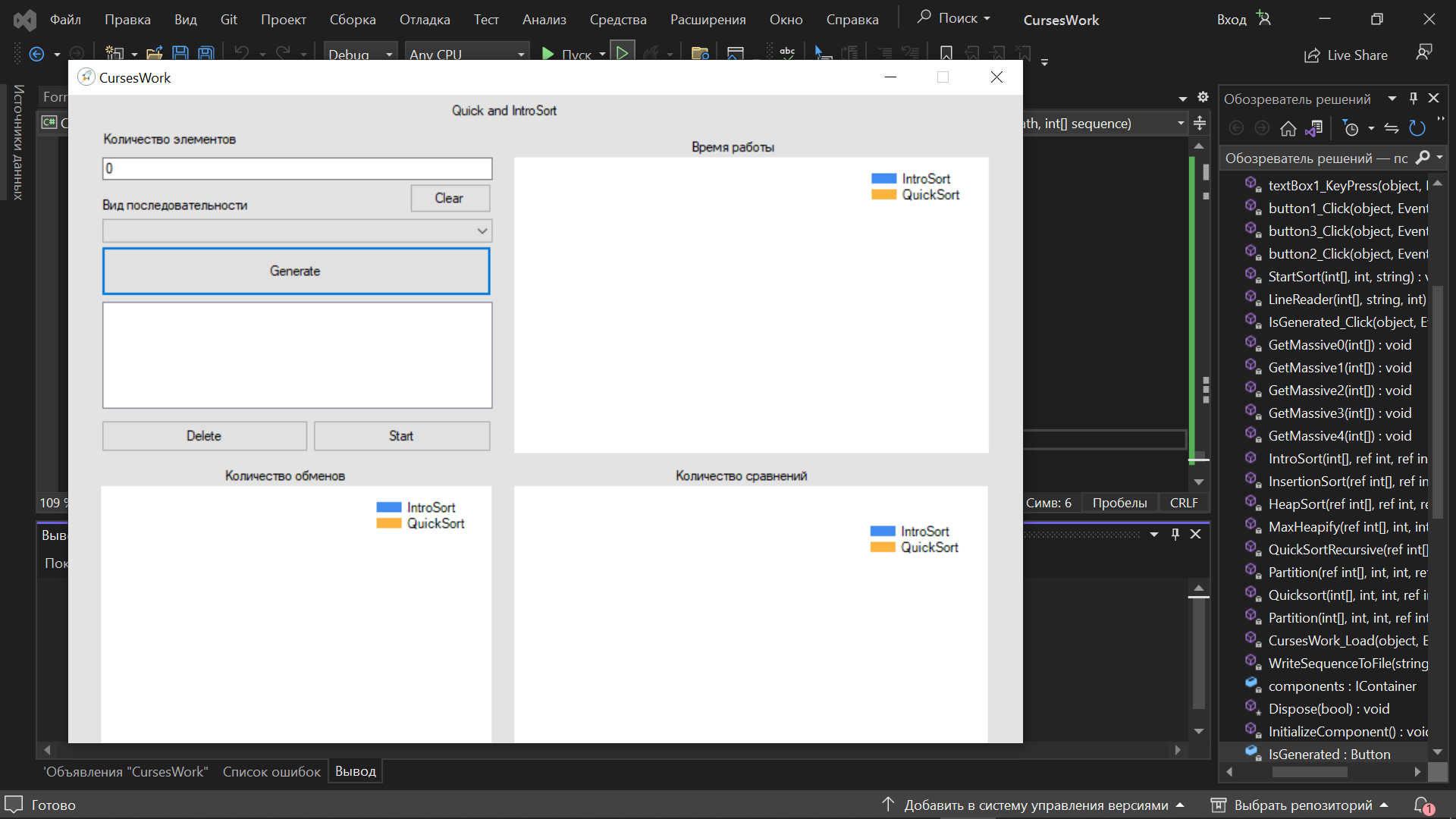


Рисунок 4. Главная экранная форма

На данном этапе пользователь может выбрать тип генерируемой последовательности, сгенерировать ее, а затем отсортировать последовательность и провести эксперимент. Для этого необходимо ввести число в поле «Количество элементов» и выбрать тип последовательности в поле «Вид последовательности» и нажать на кнопку «Generate».

1. При нажатии кнопки «Generate»:
2. Если выбран хотя бы один тип последовательности и число введено в диапазоне от 1 до 1000000, то создаются файлы, содержащие последовательности целых чисел, с соответствующим разрешением.
3. Появляется окно выбора файлов, сгенерированных ранее, представленное на рисунке 5. В данном окне можно выбрать один или несколько файлов.

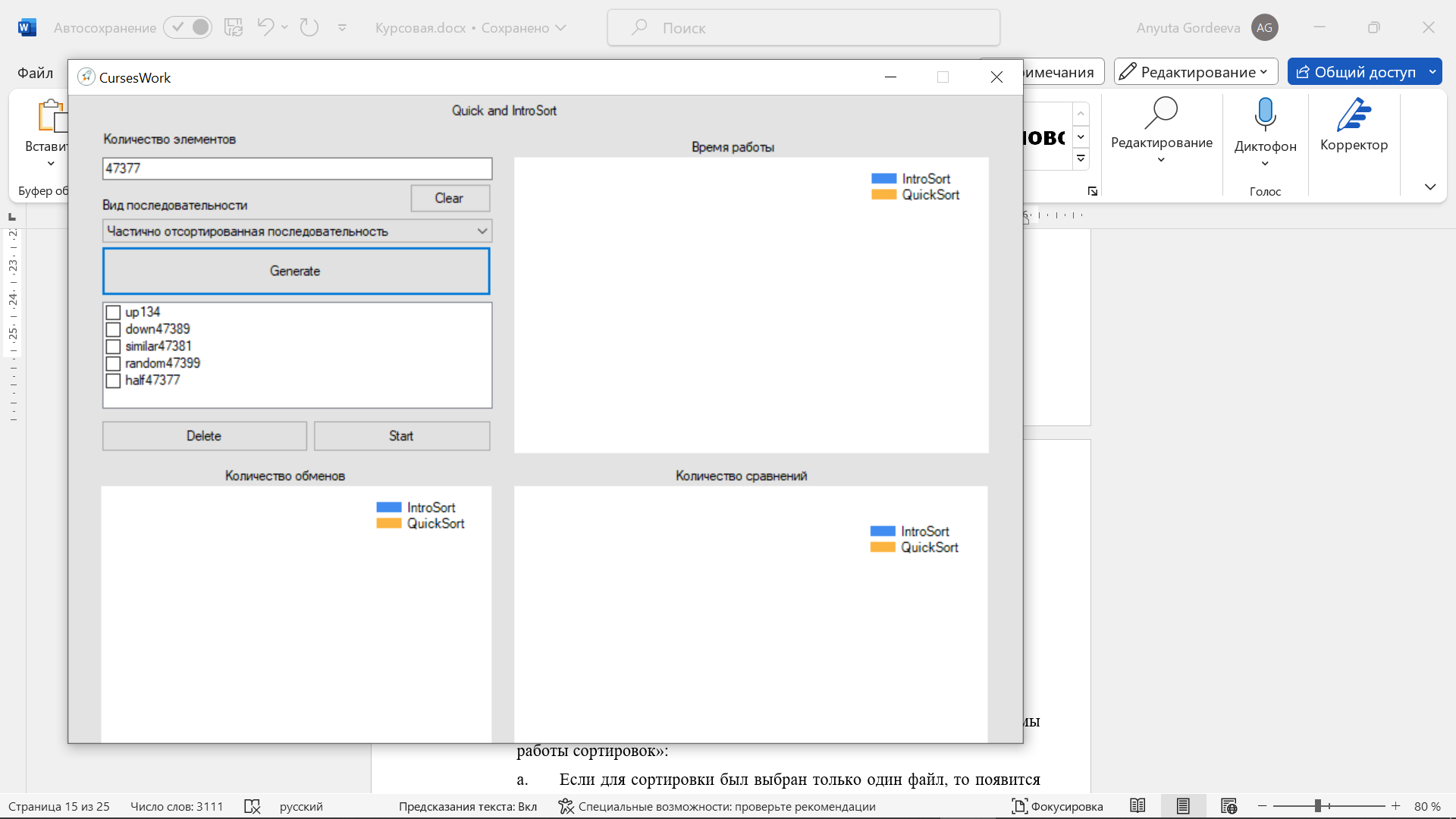


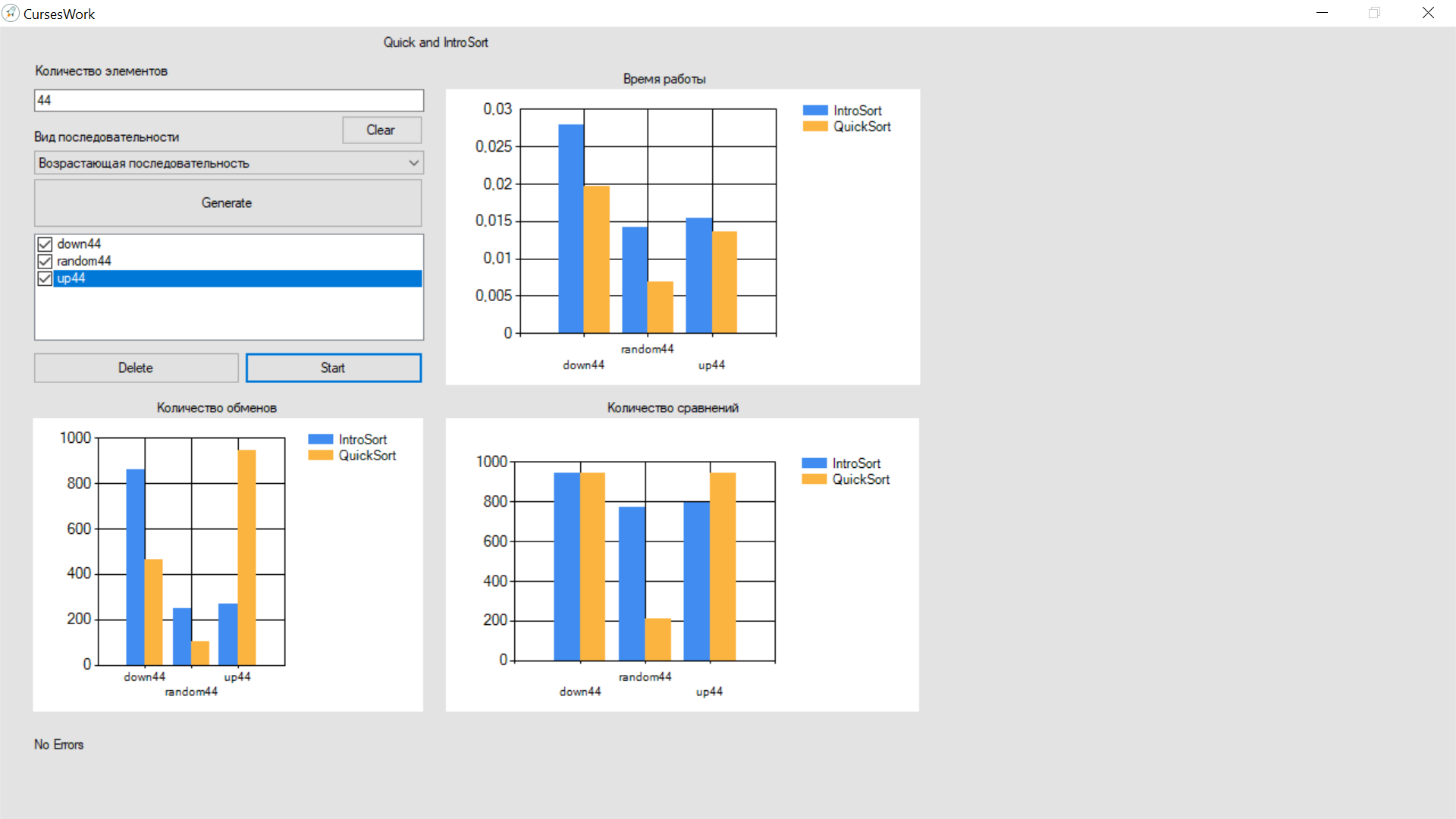
Рисунок 5. Окно выбора файлов для сортировки

При выборе файла появляются диаграммы:

* 1. Если для сортировки был выбран только один файл, то появится окно с тремя диаграммами, отображающими работу сортировок на выбранной последовательности.
  2. Если для сортировки были выбраны несколько файлов, то появится окно с тремя диаграммами, отображающими работу сортировок на всех выбранных последовательностях.

Диаграммы отображают следующие сведения: количество сравнений элементов в последовательности, количество сделанных перестановок элементов в последовательности, время работы процедуры сортировки в секундах. Пример диаграммы представлен на рисунке 6.

Данные о работе быстрой сортировки обозначены желтым цветом, данные о работе интроспективной сортировки– синим цветом.

Изображение выглядит как снимок экрана, текст, диаграмма, программное обеспечение

Автоматически созданное описаниеИзображение выглядит как снимок экрана, текст, диаграмма, программное обеспечение

Автоматически созданное описание

Рисунок 6. Диаграммы работы сортировок

Подписи под каждой диаграммой обозначают тип сортируемой последовательности и количество сортируемых элементов.

Данные по оси ОУ в диаграмме «Количество перестановок» обозначают количество совершенных сортировкой перестановок. Данные по оси ОУ в диаграмме «Количество сравнений» обозначают количество проведенных сортировкой сравнений. Данные по оси ОУ в диаграмме «Время» обозначают время, затраченное на сортировку. Если шкала пуста, значит сортировка заняла относительно малое время.

Если при заполнении полей были встречены ошибки, то программа не станет генерировать последовательность и выведет одно из следующих сообщений об ошибке (рисунок 7–11).

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание

Рисунок 7. Сообщение об ошибке: Вид последовательности не был выбран

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание

Рисунок 8. Сообщение об ошибке: Недопустимое количество элементов

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание

Рисунок 9. Сообщение об ошибке: Недопустимое количество элементов

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

Рисунок 10. Сообщение об ошибке: Такой файл уже существует

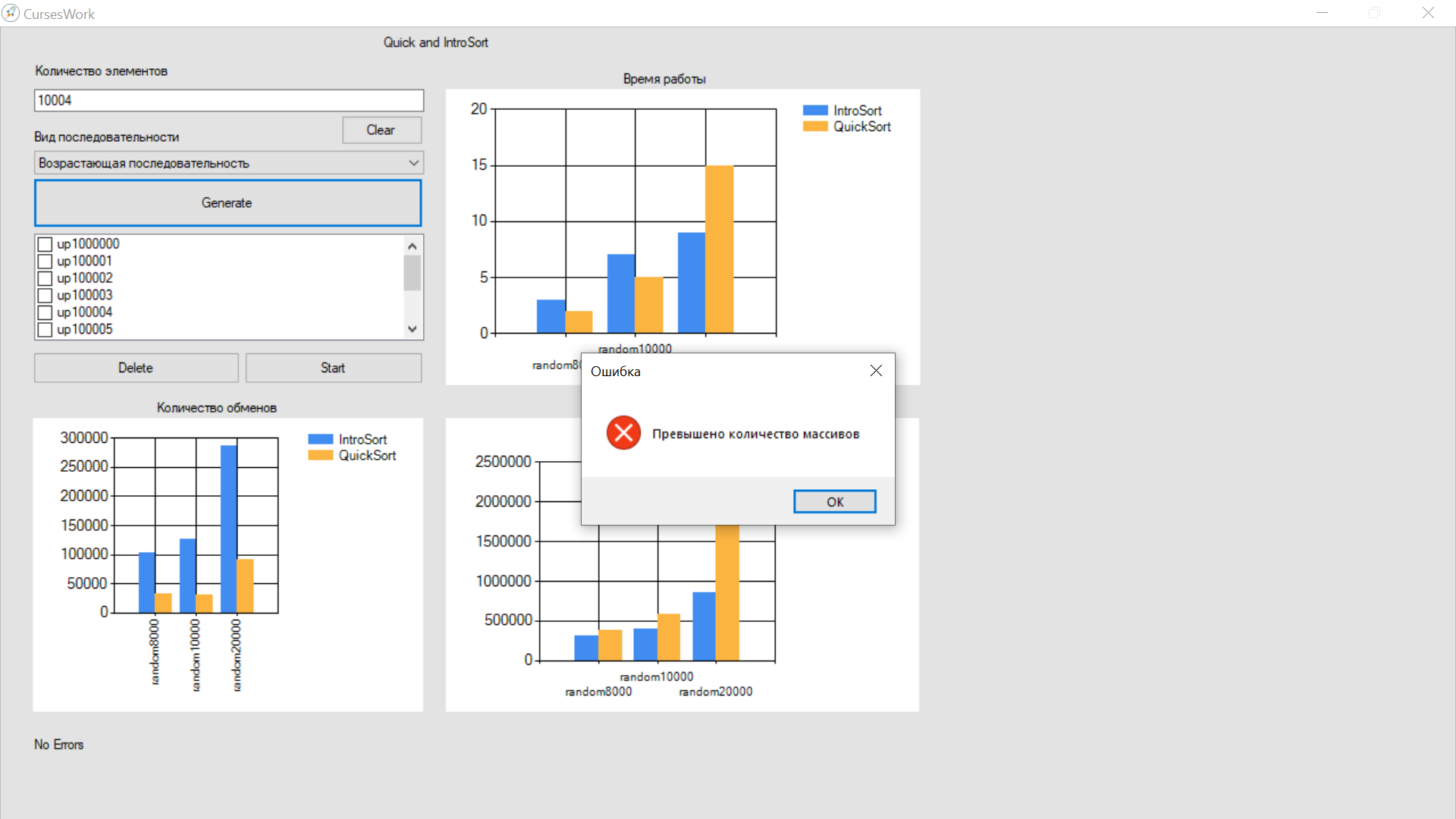


Рисунок 11. Сообщение об ошибке: Превышено количество массивов

# Описание эксперимента

В данной главе будут описаны проведенные над быстрой сортировкой и интроспективной сортировкой эксперименты, а также обоснованы результаты их работы.

## Эксперименты над быстрой и интроспективной сортировками

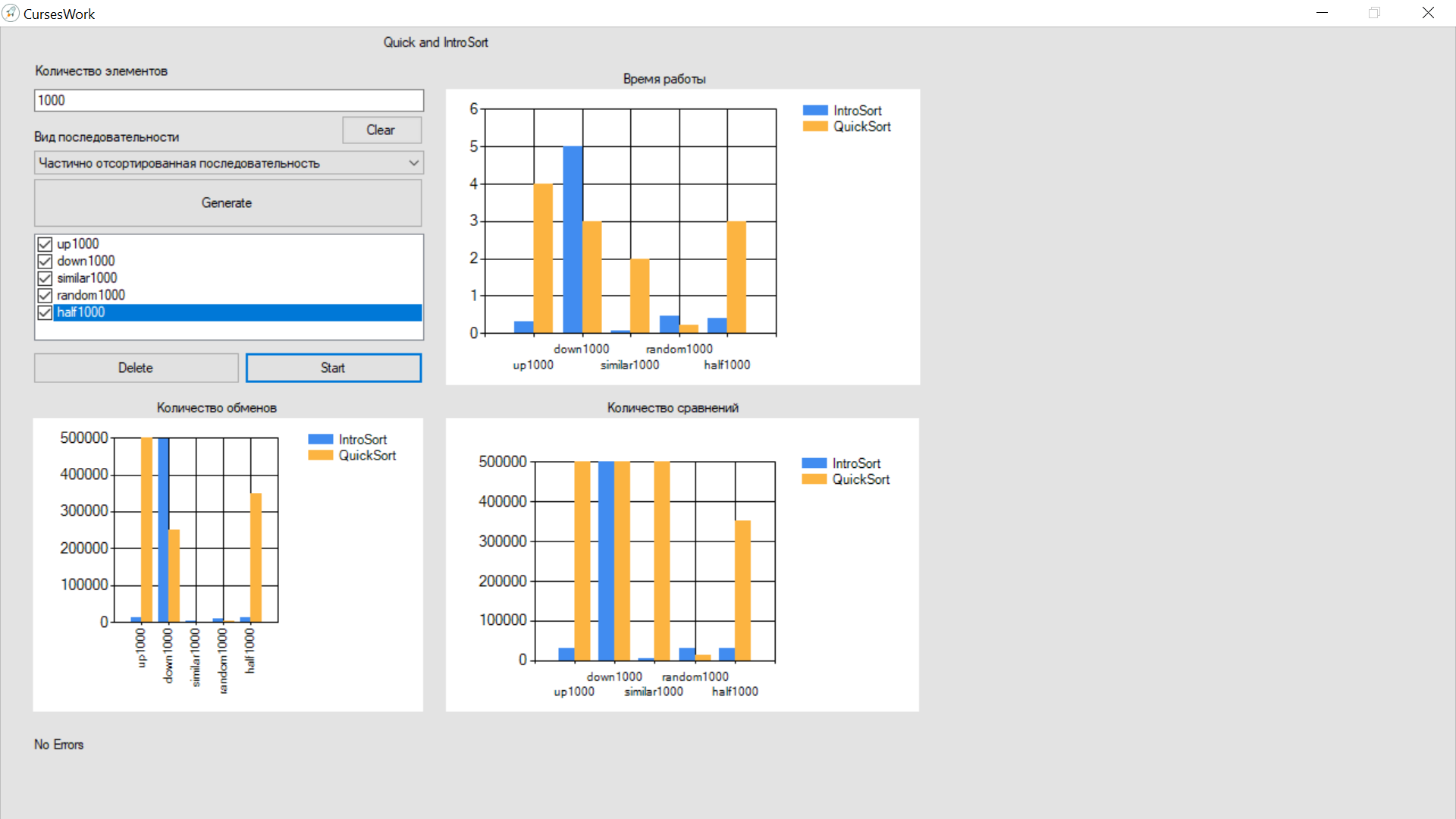
1. ***Первый эксперимент.***

Цель эксперимента: показать, что эффективность быстрой сортировки и интроспективной сортировки зависит от типа последовательности.

Входные данные: пять различных последовательностей по 1000 элементов: упорядоченная по возрастанию, упорядоченная по убыванию, случайная, частично упорядоченная, с одинаковыми элементами.

Выходные данные: результат работы сортировок представлен на рисунке 12.

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание  Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

Рисунок 12. Результат работы сортировок

Обоснование результата: при упорядоченной последовательности быстрая сортировка показывает худший результат. В то время как интроспективная сортировка не требует каких-либо действий, что является лучшим случаем для нее. Последовательность из элементов, расположенных в произвольном порядке, интроспективная сортировка сортирует эффективнее, так как в основе лежит быстрая сортировка, а при пирамидальной сортировке каждый раз происходит построение пирамиды. При частично упорядоченной последовательности быстрая сортировка, достигая заданного уровня рекурсии, переходит на пирамидальную, для которой частично упорядоченная последовательность близка к худшему случаю (упорядоченной последовательности).

1. ***Второй эксперимент.***

Цель эксперимента: показать, что эффективность интроспективной сортировки при последовательностях с произвольным порядком размещения элементов с увеличением количества элементов сильно деградирует.

Входные данные: три случайные последовательностей по 8000, 10000 и 20000 элементов.

Выходные данные: результат работы сортировок представлен на рисунке 13.

Изображение выглядит как снимок экрана, текст, диаграмма, Параллельный

Автоматически созданное описание Изображение выглядит как снимок экрана, текст, диаграмма, Параллельный

Автоматически созданное описание Изображение выглядит как снимок экрана, текст, диаграмма, Параллельный

Автоматически созданное описание

Рисунок 13. Результат работы сортировок

Обоснование результата: низкая эффективность по времени объясняется тем, что по сравнению с перестановками интроспективная сортировка производит также много сравнений в процессе выполнения быстрой сортировки.

# Заключение

В рамках данного курсового проекта была разработана компьютерная программа, выполняющая генерацию последовательностей целых чисел и ставящая эксперименты над различными сгенерированными последовательностями.

На основе проведенных экспериментов можно сделать вывод о том, что эффективность метода сортировки в общем случае зависит от типа сортируемой последовательности и количества элементов. Однако эффективность интроспективной сортировки сильно колеблется при изменении количества элементов, что объясняется входом в глубокую рекурсию. Если сортируются последовательности небольших или средних размеров, то интроспективная сортировка проявляет себя лучше, чем быстрая.

Таким образом, при выборе алгоритма сортировки, большое значение имеют размеры сортируемых последовательностей и их тип.

# Список литературы и интернет-источников

1. Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – М.: Мир, 1989. – 360 с.
2. Седжвик Р. Фундаментальные алгоритмы на С++. Части 1- 4. Анализ. Структуры данных. Сортировка. Поиск / Р. Седжвик. – К.: Издательство «ДиаСофт», 2001. – 688 с.
3. Гуев Т. Сортировки в .NET [Электронный ресурс] // habr – Коллективный блог – 2013. – Режим доступа: <https://habr.com/ru/post/188012/>.
4. AlgoLab [Электронный ресурс] – Блог. – Режим доступа: <http://algolab.valemak.com/introspective>.
5. GeeksforGeeks [Электронный ресурс] – Информационный портал. – Режим доступа: https://clck.ru/HBNbY.