ФГБОУ ВПО

"Уфимский государственный нефтяной технический университет"

Кафедра вычислительной техники и инженерной кибернетики.

Дисциплина «Структуры и алгоритмы обработки данных»

ОТЧЕТ

По лабораторной работе № 1

# «Сравнение сортировок»

Выполнил ст. гр. БПО 15-01 Курамшин А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Проверил: преподаватель Жолобова Г.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

**Уфа 2017**

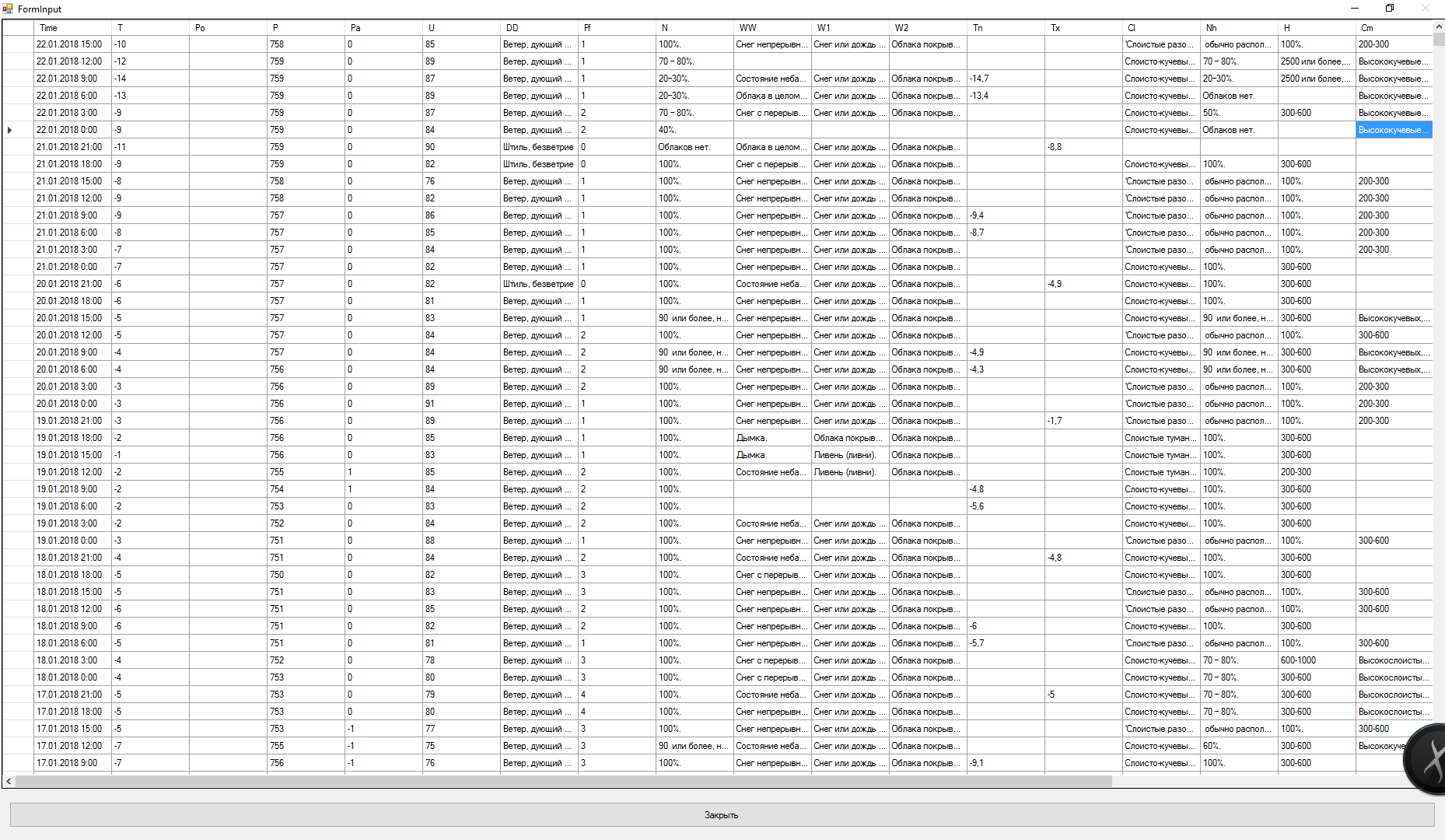
**Исходные данные:**

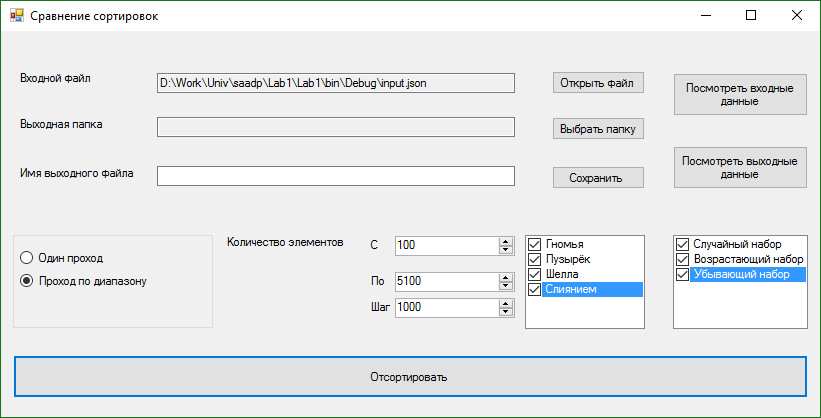
Массив погодных данных за последние несколько лет.

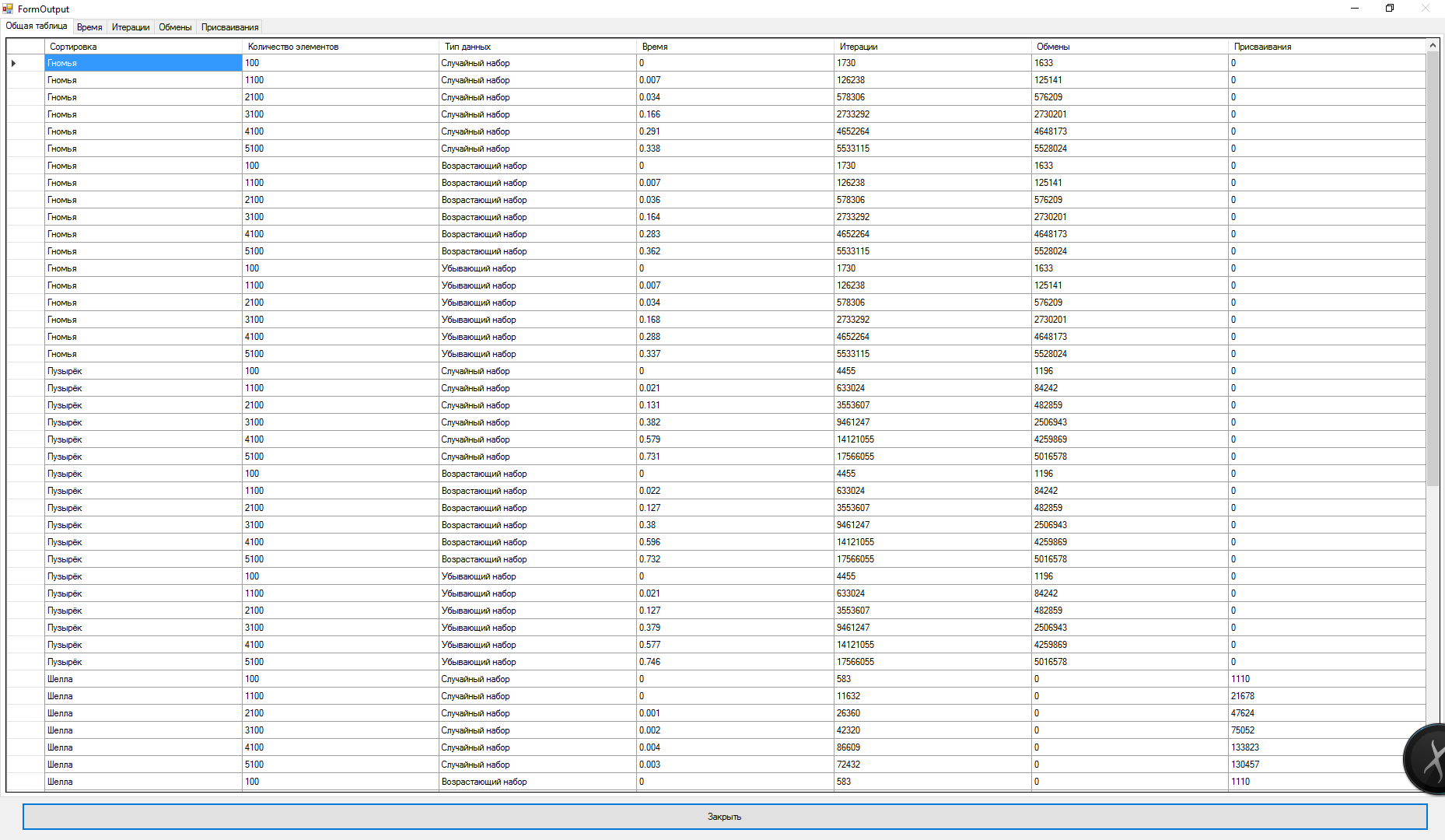
**Задача:**

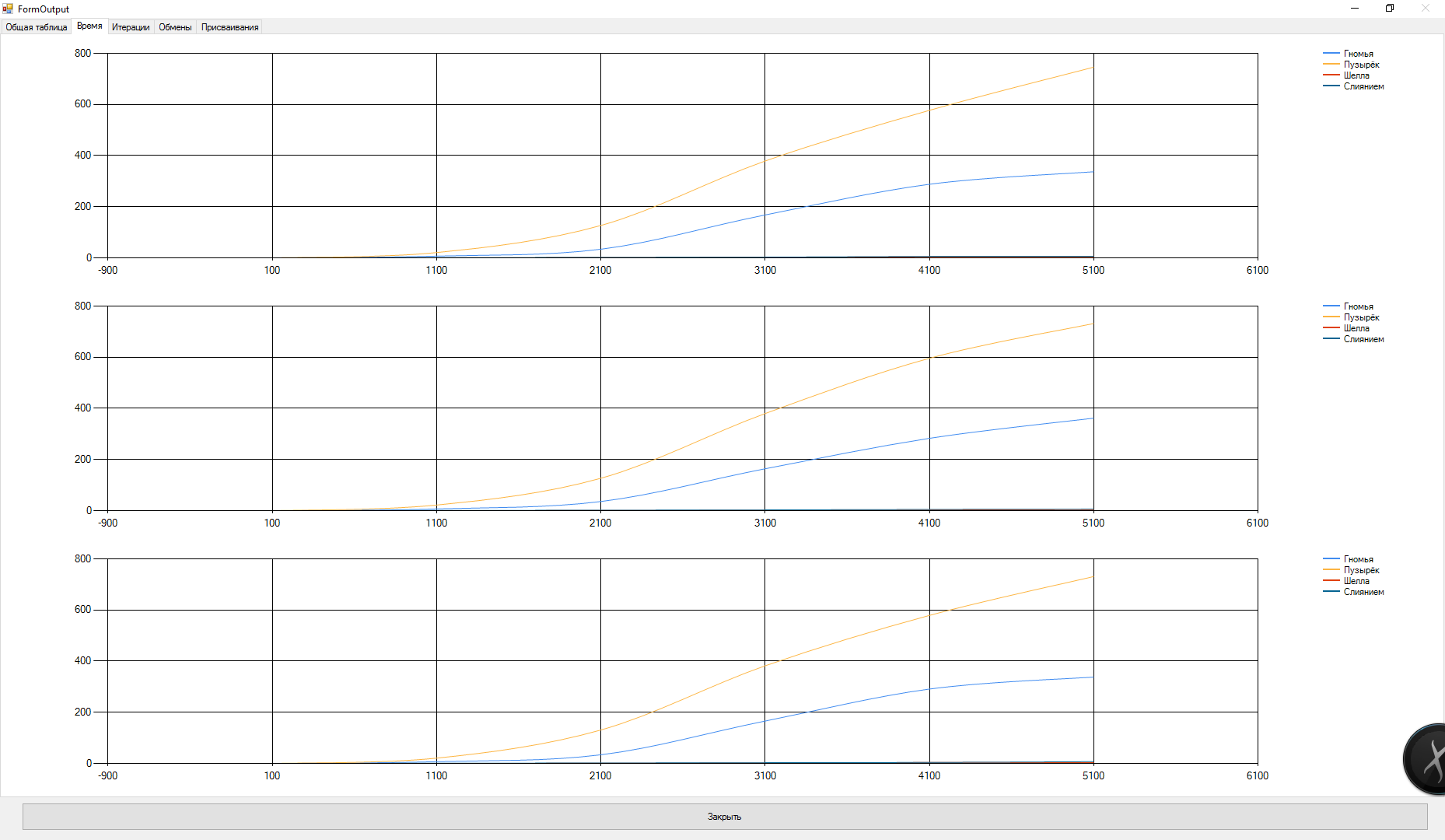
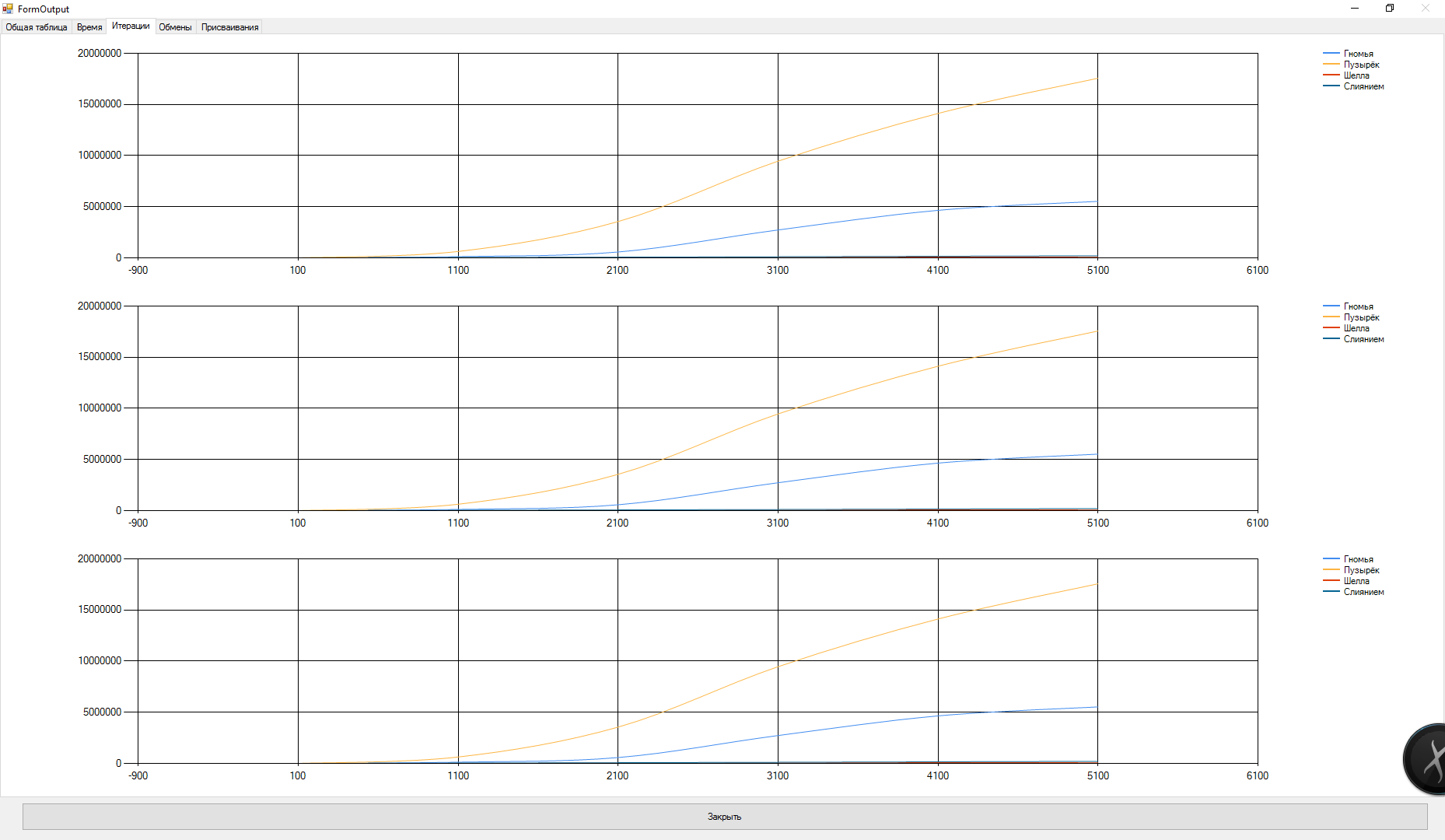
Сравнить время работы, количество сравнений и итераций при различном количестве сортируемых элементов и различных наборах элементов (случайный, отсортированный по возрастанию и по убыванию) у следующих сортировок: гномья, пузырьком, Шеллв, слянием.

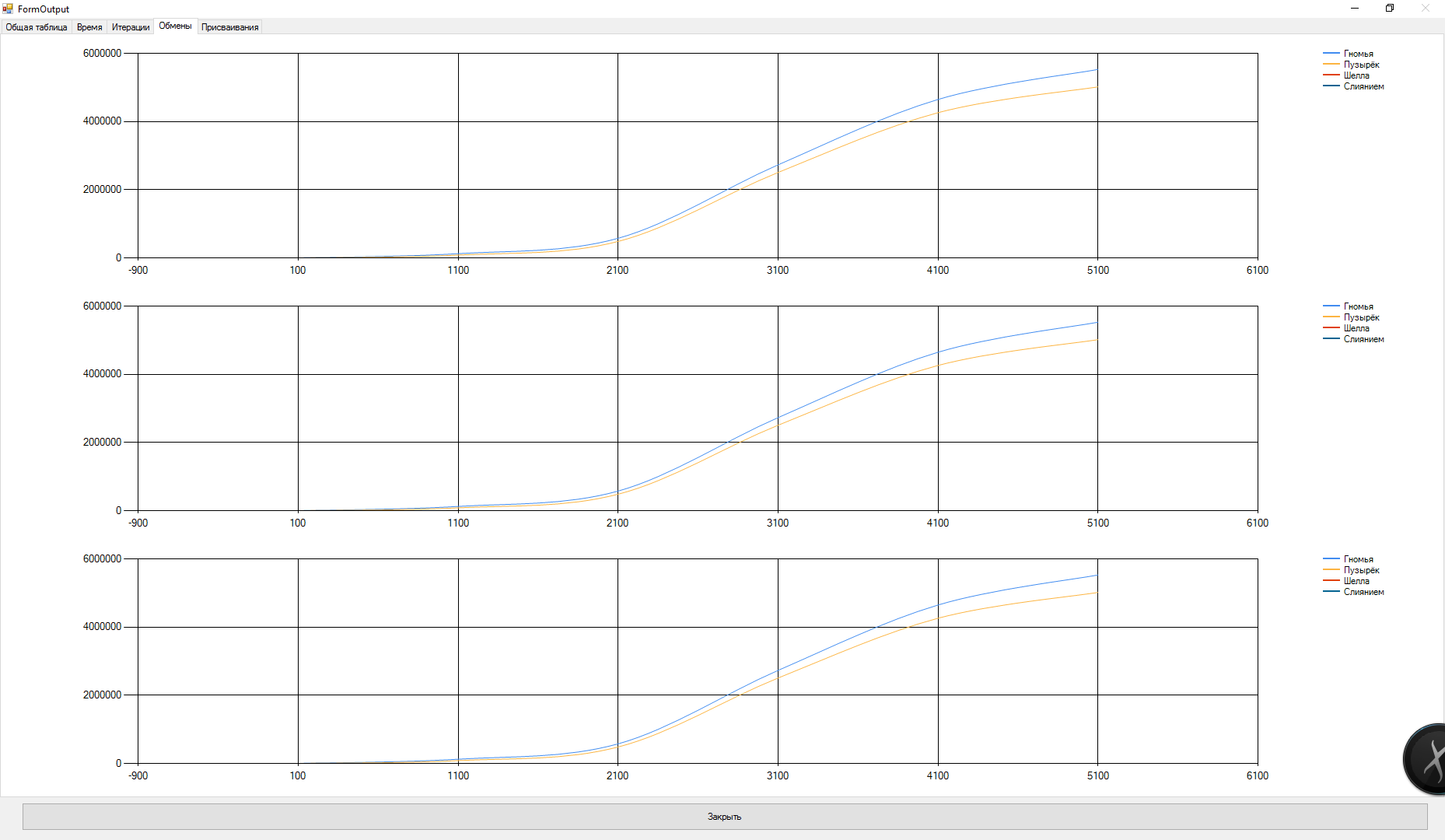
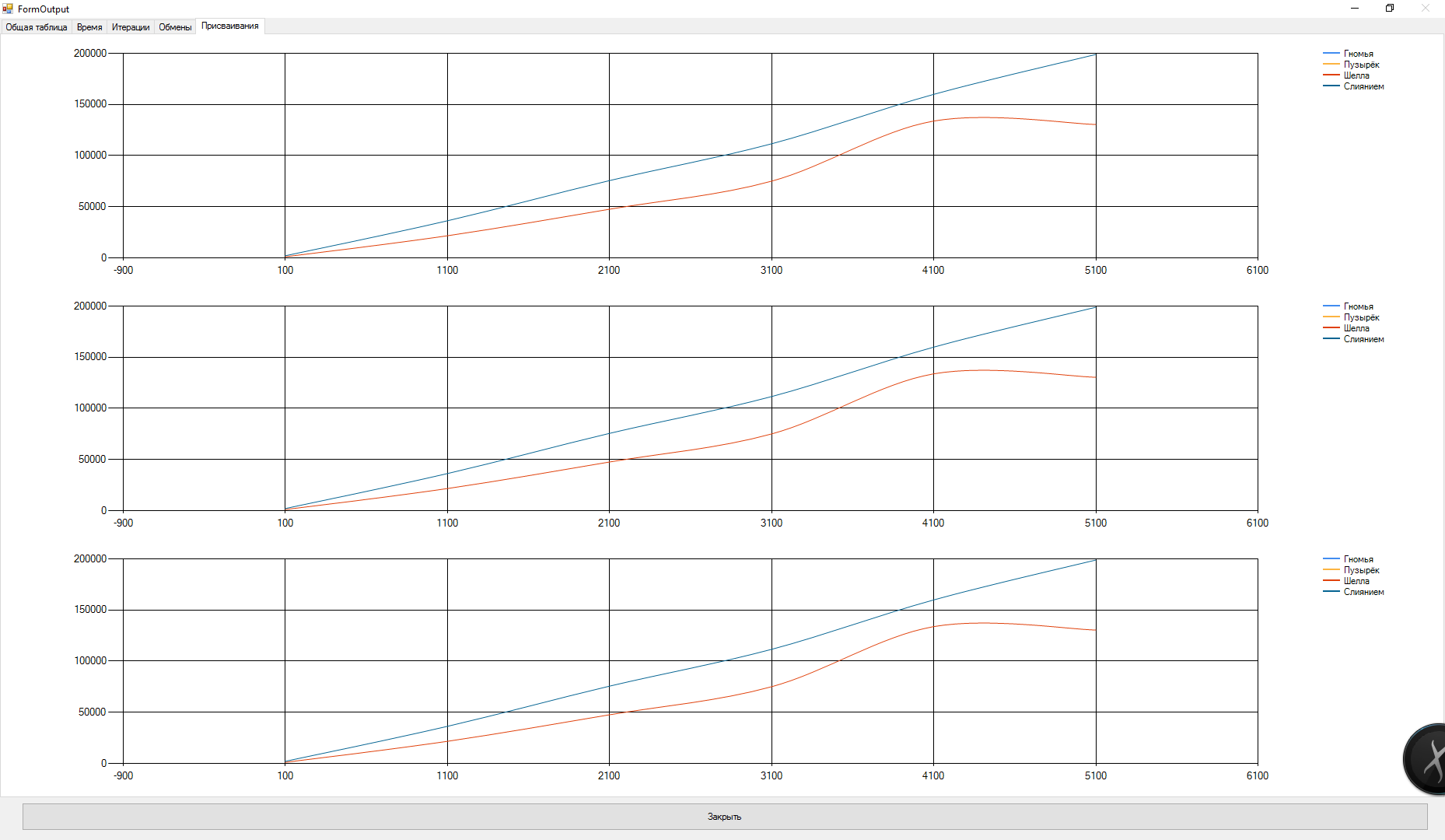
**Результат работы программы:**

Проведём все возможные сортировки на малых объёмах данных:



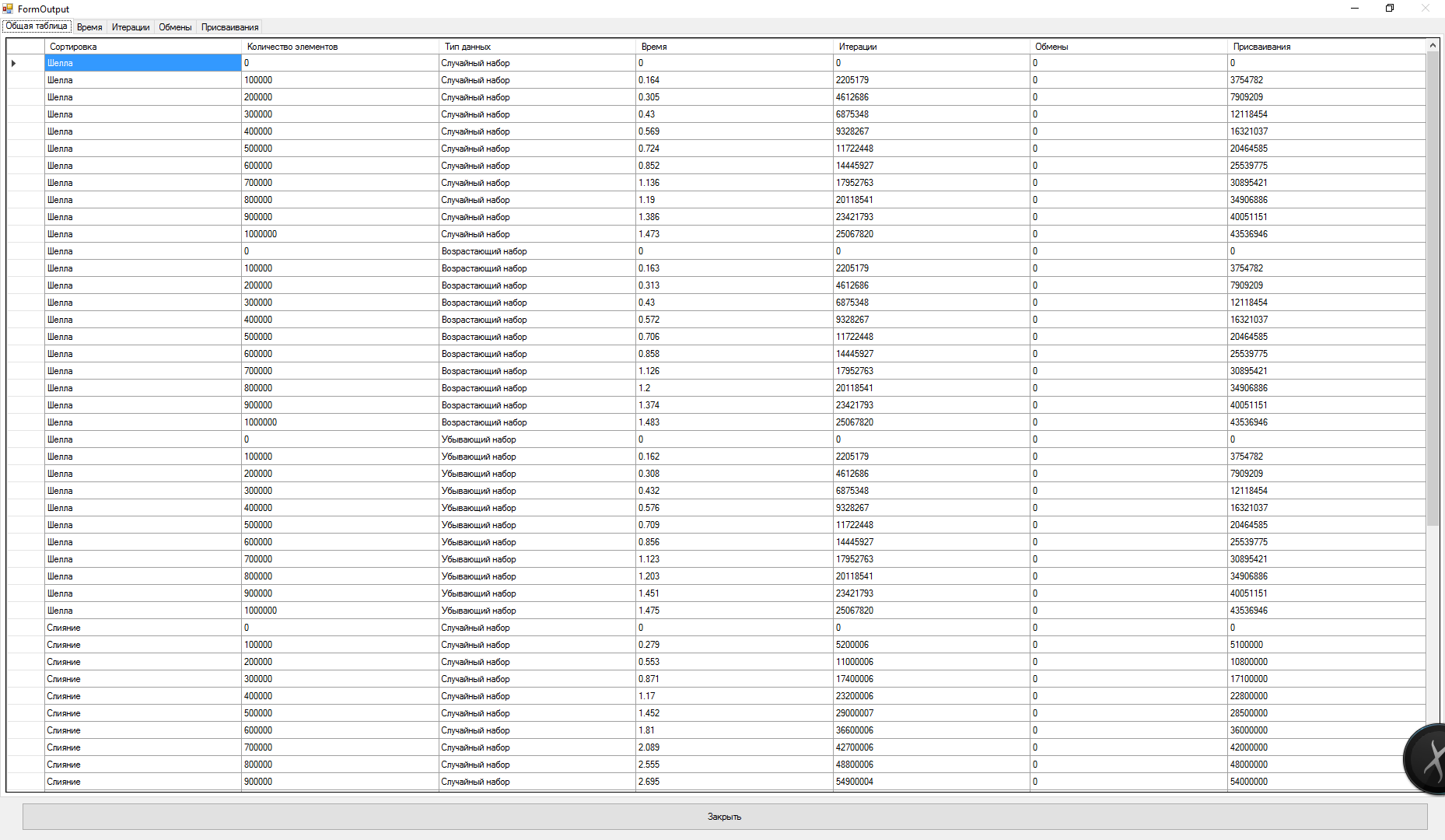


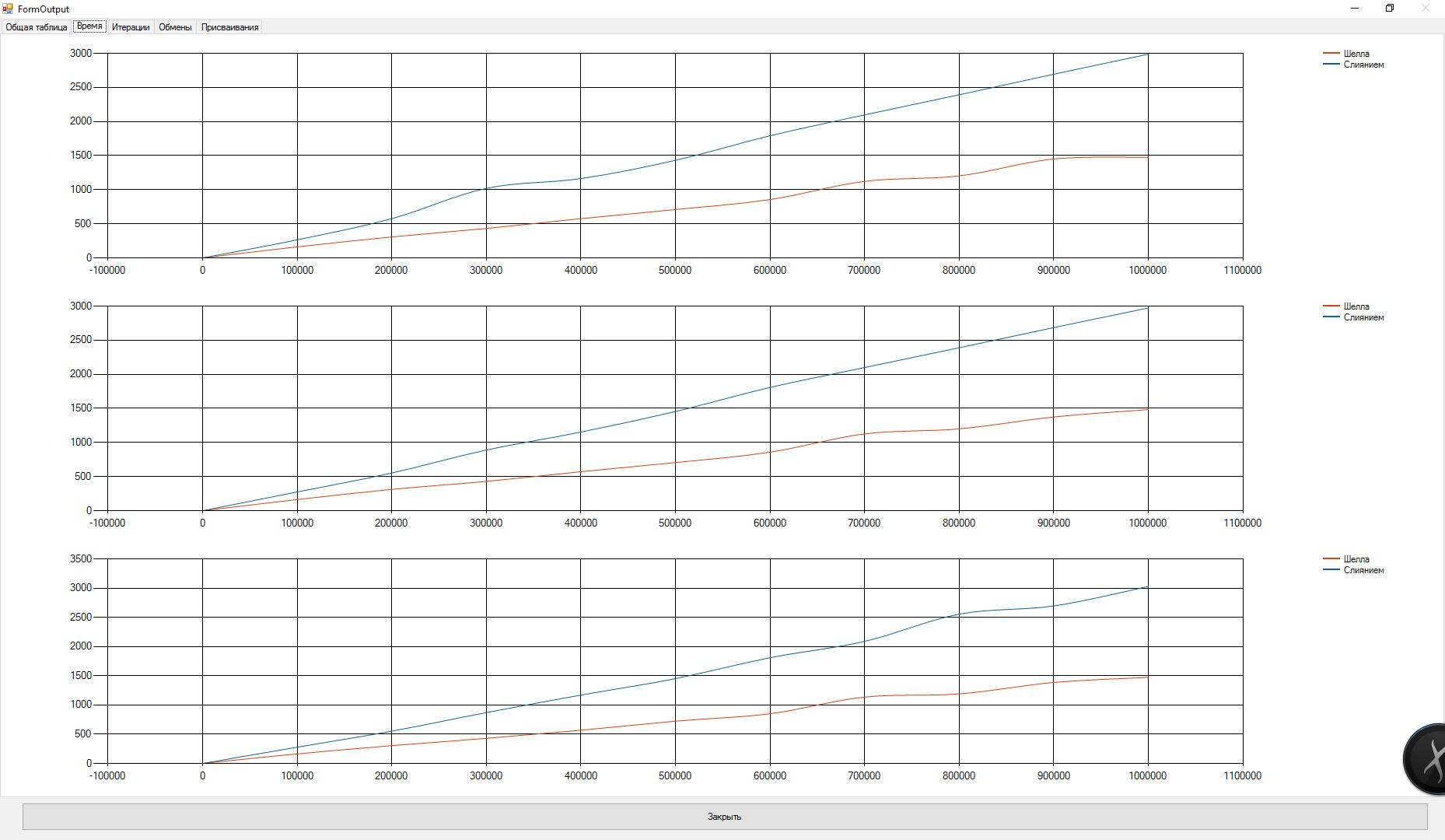
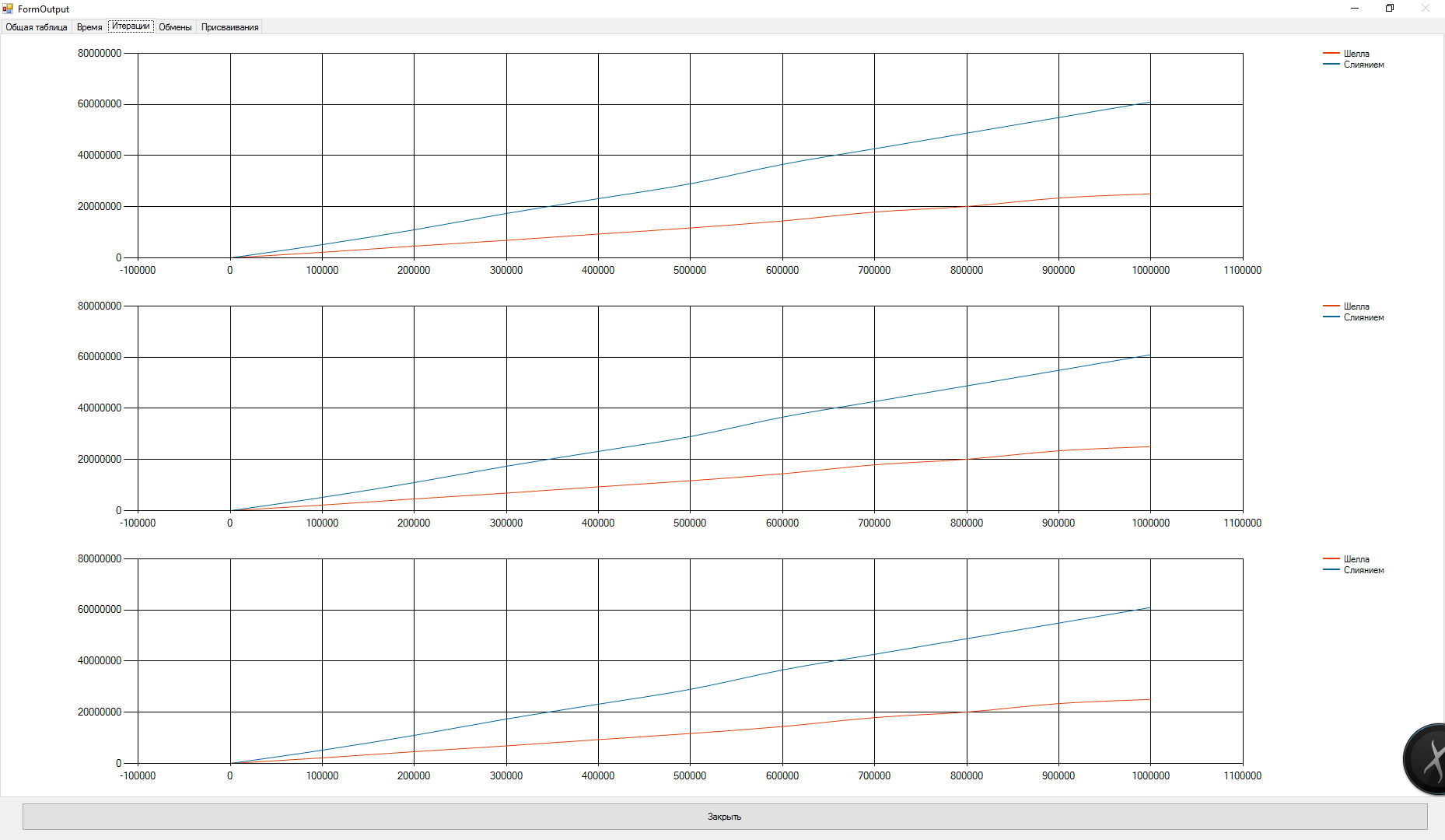
 

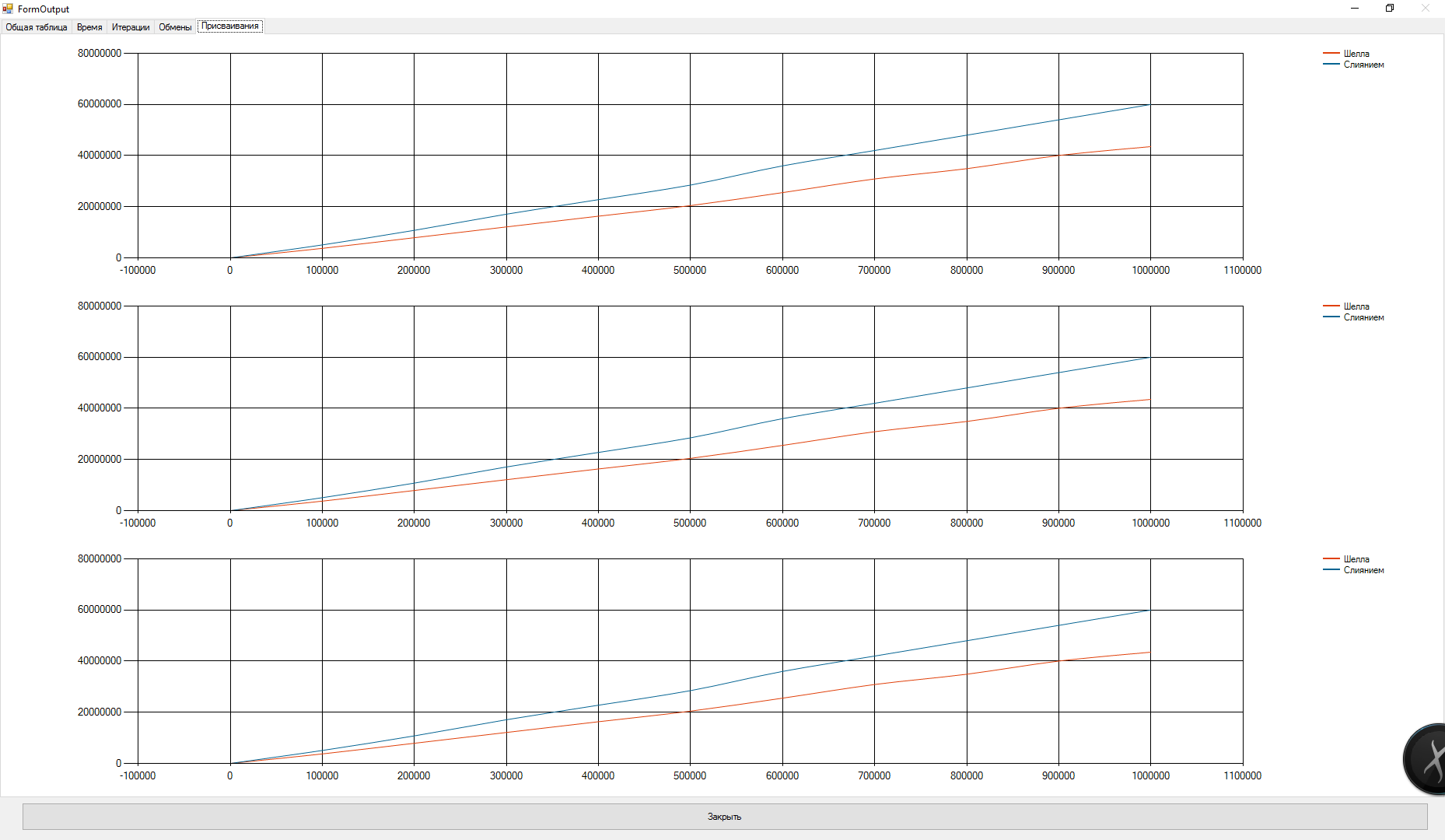
 

Мы видим, что и пузырёк и гномья сортировка тратят больше времени и совершают больше итераций при любых наборах данных. Также они совершают больше операций по перемещению сортируемых элементов – в одном обмене происходит три присваивания, а обменов что у гномьей сортировки, что у пузырька, в разы больше, чем у сортировок Шелла и слиянием. Сортировки гномья и пузырьком неэффективны при работе с данными. При увеличении размеров данных разницы в затратах по времени и ресурсам у этих сортировок по сравнению с сортировками Шелла и слиянием будут лишь увеличиваться.

Проведём сравнение сортировок Шелла и слиянием при больших объёмах данных:





Мы видим, что сортировка Шелла эффективнее сортировки слиянием по времени и количеству выполненных итераций приблизительно в два раза, а также примерно в полтора раза по количеству присваиваний. Но даже при объёмах данных в 1 000 000 элементов время сортировки выходит довольно малым – около трёх секунд для слияния и полутора для Шелла, в то время как такое же время сортировки пузырьком и гномья сортируют 5-10 тысяч элементов.

**Вывод:**

Сортировки пузырьком и гномья неэффективны при любых размерах данных. Сортировки слиянием и Шелла быстро работают даже при больших массивах данных размером в миллион элементов, но сортировка Шелла, всё же, в полтора раза эффективнее сортировки слиянием.

ФГБОУ ВПО

"Уфимский государственный нефтяной технический университет"

Кафедра вычислительной техники и инженерной кибернетики.

Дисциплина «Структуры и алгоритмы обработки данных»

ОТЧЕТ

По лабораторной работе № 2

# «Внешние сортировки»

Выполнил ст. гр. БПО 15-01 Курамшин А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Проверил: преподаватель Жолобова Г.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

**Уфа 2017**

**Исходные данные:**

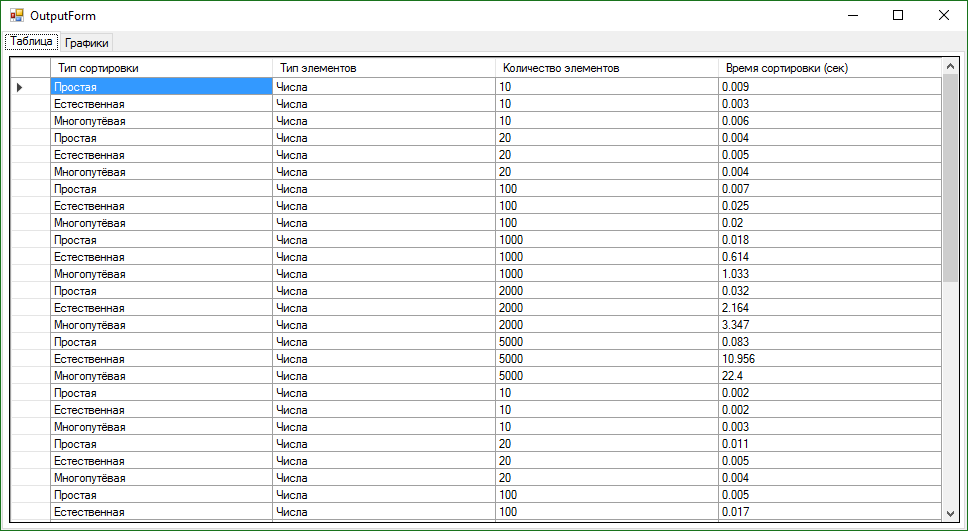
Множество файлов со случайными наборами данных разного размера, делящиеся на два типа: содержащие числа и содержащие массивы структур данных.

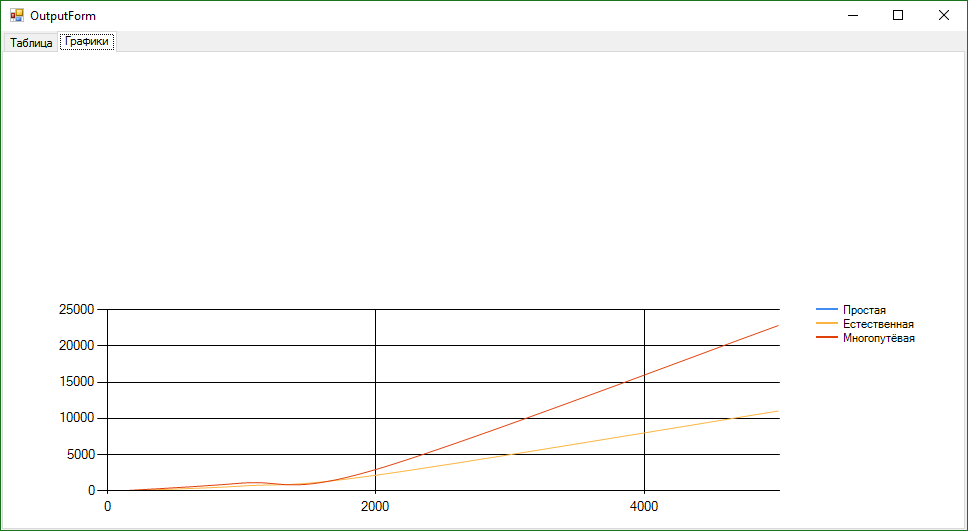
**Задача:**

Сравнить время работы трёх видов внешней сортировки при разных наборах данных.

**Результат работы программы:**

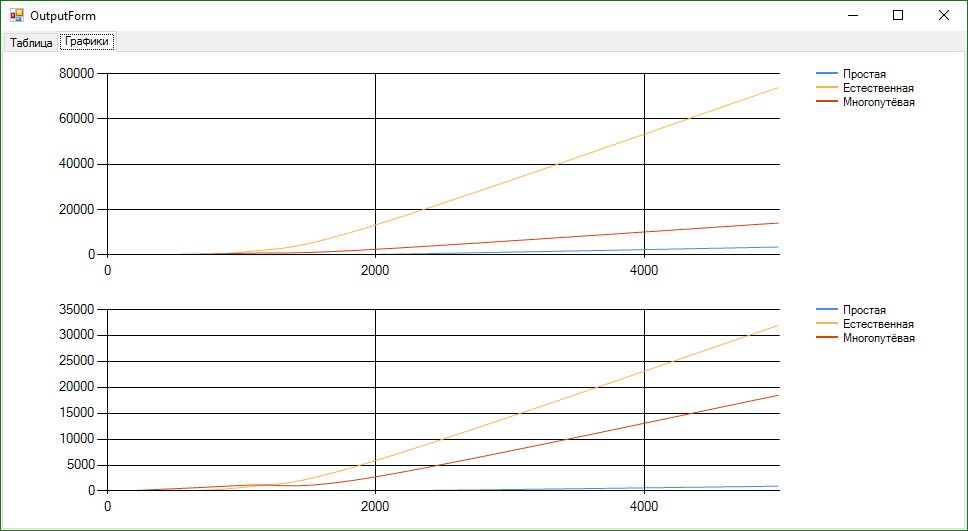
Сравним работу сортировок на массивах чисел, размерами в 10, 20, 100, 1000, 2000 и 5000 элементов. Для достоверности перепроверим результат два раза:





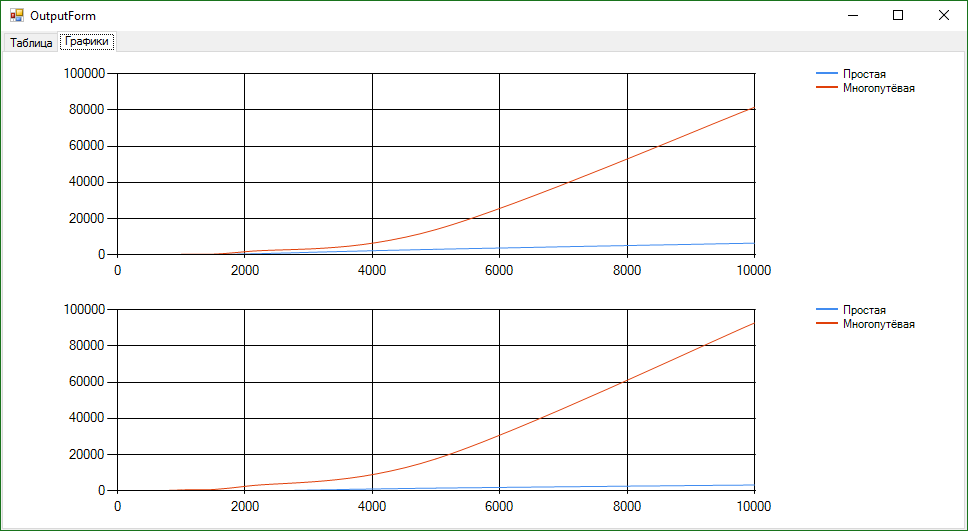
На графике простая сортировка даже не заметна, т.к. её линия почти сливается с осью координат. В данном случае она намного быстрее естественной и многопутёвой сортировок. Многопутёвая же в данном случае наименее эффективная.

Проведём сортировку массивов структур данных размерами в 10, 20, 100, 1000, 2000 и 5000 элементов:



Естественная сортировка получила наименьшую производительность. Это не удивительно, т.к. она наиболее эффективная при упорядоченных наборах данных, либо же при данных с большими сериями.

Добавим в список источников данные размером десять тысяч и отсортируем их простой и многопутёвой сортировками:



Простая сортировка остаётся наиболее оптимальной, но только в данном случае: все данные во всех файлах являются случайным набором. Если набор будет отсортирован по убыванию, то эффективность простой сортировки не изменится, в то время как естественная сортировка отсортирует данные за один проход. Многопутевая сортировка также отсортирует все данные за один проход, но при этом будет создано количество файлов, равное количеству сортируемых элементов, из-за чего её производительность может стать хуже, чем у простой сортировки. При упорядоченном по возрастанию наборе данных многопутевая и естественная сортировки покажут приблизительно одинаковые результаты, но многопутевая будет ближе, т.к. она при первом же проходе запишет результат в один файл, в то время как естественная сортировка будет возвращать результат в исходный файл.

Также и многопутевая сортировка будет наиболее эффективной при наличии множества серий, и чем длиннее серии, тем она будет эффективнее, т.к. каждая серия будет записана в отдельный файл.

Также на производительность многопутевой сортировки сильно влияет скорость записи данных винчестера – в отличие от других двух сортировок в многопутевой приходится не только записывать данные в файлы, но и создавать множество файлов. И многопутевая сортировка тратит наибольшее количество ресурсов, т.к. в каждый момент времени в оперативной памяти хранится массив, содержащий последние считанные из каждого входного файла элементы, в то время как у естественной и простой сортировок хранятся лишь два элемента.

В то же время многопутевая сортировка наиболее производительна при гигантских объёмах данных – размером одного элемента в сотни мегабайт, а то и гигабайт, т.к. на выходе изначально будет получаться множество относительно небольших файлов, в то время как две других сортировки будут работать с тремя гигантскими файлами всё время (например сотня входных файлов по 5 гб, которые превратились в двадцать выходных по 25гб в многопутевой сортировке и 1 входной на 500 гб и два выходных по, например, 200 и 300гб в двух оставшихся сортировках). Такой подход помогает размещать данные на разных носителях, если на одном не хватает свободного места.

**Вывод:**

В зависимости от формата входных данных, их количества и свободных ресурсов компьютера применяются различные виды сортировок. Так, при прочих равных, при случайных наборах данных наиболее эффективна простая сортировка, на упорядоченных – естественная, а на имеющих большое количество серий – многопутевая. При гигантских объёмах сортируемых данных (в сотни ГБ, или даже ТБ) наиболее эффективна многопутевая сортировка, т.к. позволяет распределить данные во время сортировки по нескольким носителям.

ФГБОУ ВПО

"Уфимский государственный нефтяной технический университет"

Кафедра вычислительной техники и инженерной кибернетики.

Дисциплина «Структуры и алгоритмы обработки данных»

ОТЧЕТ

По лабораторной работе № 3

# «Структуры»

Выполнил ст. гр. БПО 15-01 Курамшин А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Проверил: преподаватель Жолобова Г.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

**Уфа 2017**

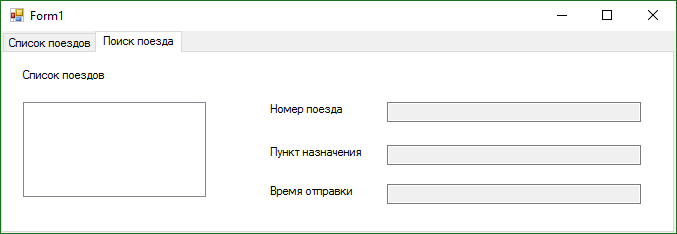
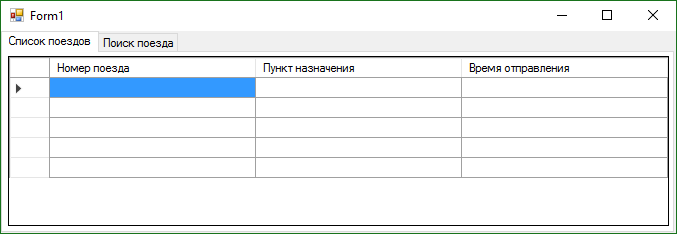
**Задача:**

Описать структуру с именем TRAIN, содержащую следующие поля: название пункта назначения, номер поезда, время отправления. Написать программу, выполняющую следующие действия:

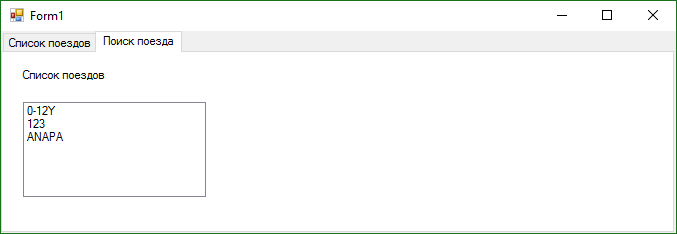
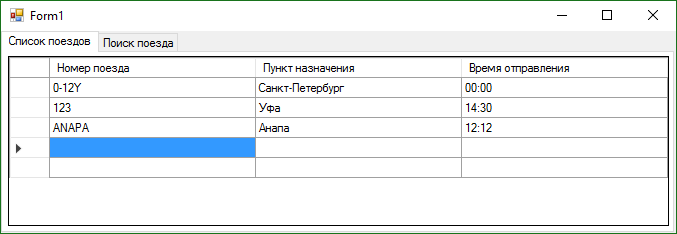
* Ввод с клавиатуры данных в массив, состоящий из пяти элементов типа TRAIN. Записи должны быть упорядочены по номерам поездов.
* Вывод на экран информации о поезде, номер которого введён с клавиатуры. Если такого поезда нет, вывести на экран соответствующее сообщение.

**Результат работы программы:**

Начальное состояние:

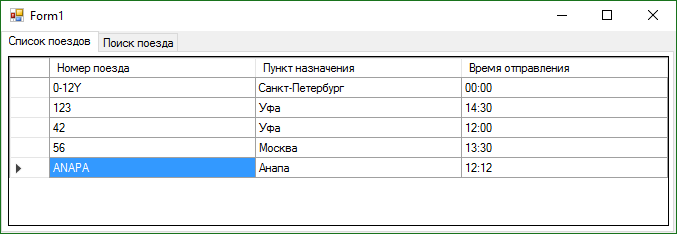


Ввод нескольких поездов в случайном порядке:

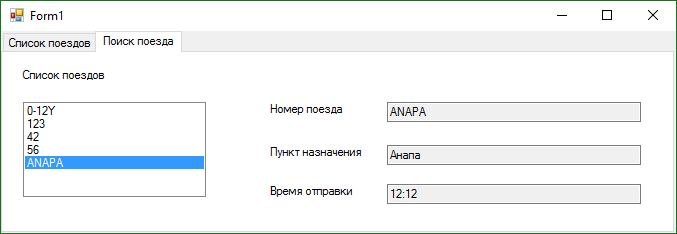
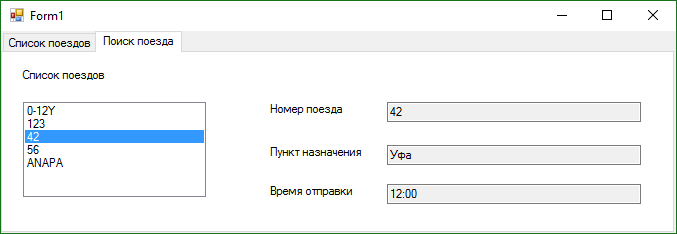


Данные были отсортированы автоматически в процессе ввода.

Ввод всех поездов:



Получение информации по поездам (т.к. приложение не консольное, а GUI, то ввод названия пользователя был заменён на выбор названия из автоматически генерирующегося списка):



**Вывод:**

Было создано приложение Windows Forms, реализующее функционал составления списка поездов и получения информации по интересующему поезду. Для хранения списка поездов был использован тип данных List(список), как наиболее оптимальный в данной ситуации – каждому поезду был присвоен индекс, и при выборе названия поезда в списке именно по индексу получается информация из списка.