МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Челябинский государственный университет»**

**(ФГБОУ ВО «ЧелГУ»)**

Институт информационных технологий

Кафедра информационных технологий и экономической информатики

ОТЧЕТ

по лабораторной работе №4

Выполнил:

Студент группы ПрИ-202 Приходько Даниил Александрович

Студент группы ПрИ-202 Саламатин Алексей Юрьевич

Студент группы ПрИ-202 Скоробогатов Максим Дмитриевич

Принял:

Преподаватель ИИТ Николаев Иван Евгеньевич

Отчет защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка

Челябинск 2024 г.

Содержание

Введение..................................................................................................................3

Задание 1………......................................................................................................4

Задание 2................................................................................................................17

Задание 3................................................................................................................29

Введение

Цель работы: понять и доступно объяснить Шестикласснику, как работают различные сортировки данных. Написать приложение с визуальной частью, благодаря которой будет видно весь процесс сортировок.

Задание 1: Реализовать демонстрацию работы 4 алгоритмов внутренней сортировки. А конкретно, нами были выбраны: Select Sort, Shell Sort, Quick Sort, Merge Sort.

Задание 2: Дана таблица, состоящая из нескольких полей. Таблица находится в файле. Реализовать программу, которая одним из методов (прямым, естественным или многопутевым слиянием) сортирует записи и записывает их в результирующий файл. Метод внешней сортировки и ключевой атрибут, по которому осуществляется сортировка выбирается пользователем. Количество атрибутов в таблица может быть произвольным.

Задание 3: Пусть имеется текст, состоящий из слов. Необходимо разбить текст на отдельные слова и провести их сортировку в лексикографическом порядке. Причем одинаковые слова будут в отсортированной последовательности идти друг за другом. Надо выбрать два алгоритма сортировки, один из которых базовый или усовершенствованный, и другой алгоритм ABC- сортировка или redix сортировка. После получения отсортированного массива необходимо пройтись по нему и для каждого слова подсчитать, сколько раз оно встречается. Результаты подсчетов вывести на экран. Далее необходимо провести эксперименты с использованием обоих алгоритмов по сортировке текстов различной длины (100, 500, 1000, 2000, 5000 слов или более). Причем для каждого эксперимента произвести замеры времени сортировки. Получившиеся

данные оформить в таблицу.

Задание 1

В рамках этой части были реализованы алгоритмы внутренней сортировки Select Sort, Shell Sort, Quick Sort, Merge Sort.

Сортировка выбором – это алгоритм, суть которого заключается в постоянном сравнении элементов неотсортированой части. Наименьший элемент в неотсортированой части меняется местами с первым ее элементом и сразу становится отсортированой частью.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 1.1.1 – 1.1.3.

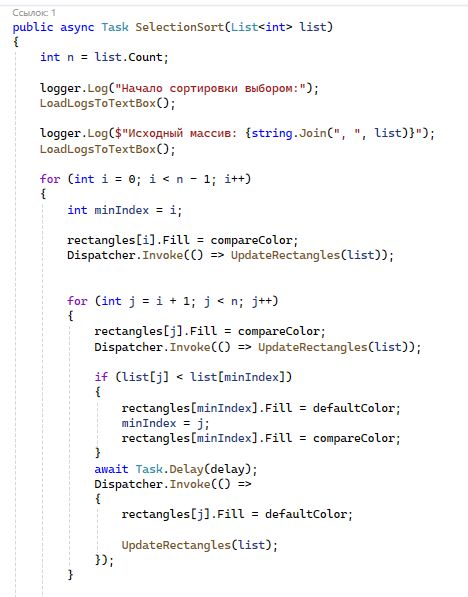


Рис.1.1.1 Код сортировки выбором

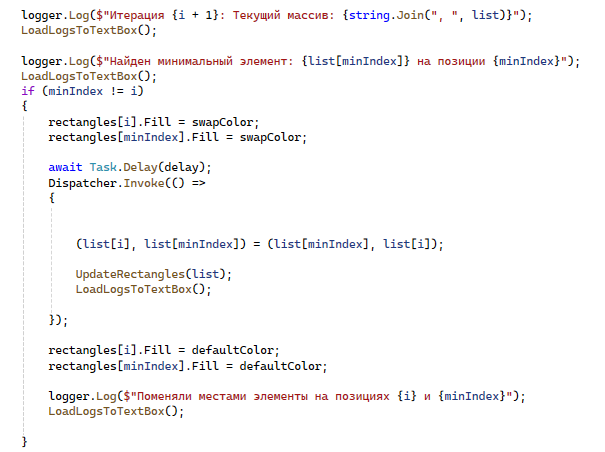


Рис.1.1.2 Код сортировки выбором

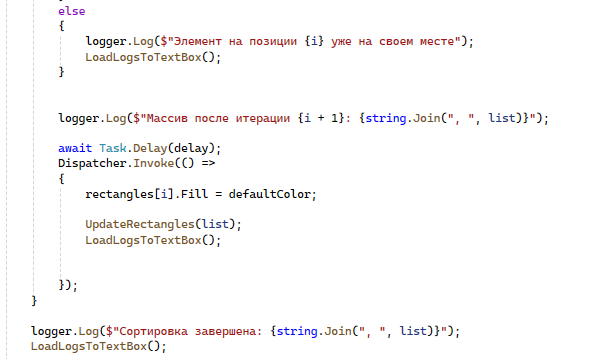


Рис.1.1.3 Код сортировки выбором

Сортировка Шелла – это модифицированая сортировка вставками. Ее суть в том, что массив данных разбивается на так называемые Gap’ы (зазоры). Каждая итерация сортировки – это сравнение элементов массива на расстоянии текущего зазора друг от друга, а затем упорядочивание этих элементов. С каждым проходом по массиву зазор уменьшается вдвое и эти действия повторяются до окончания сортировки.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 1.2.1 – 1.2.2.

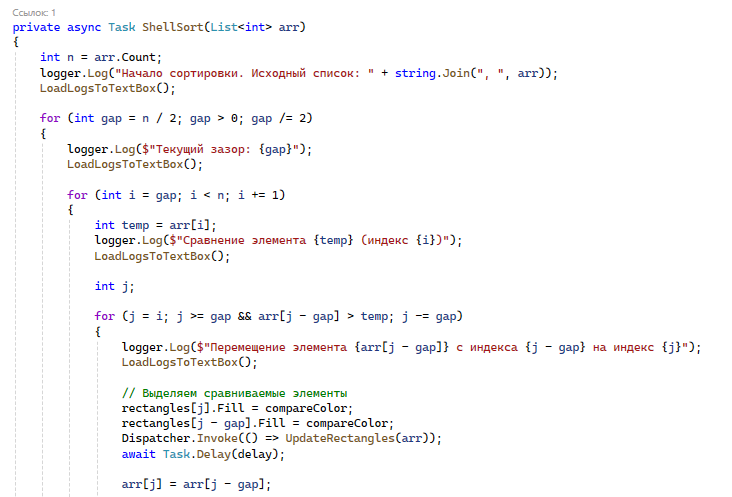


Рис.1.2.1 Код сортировки Шелла

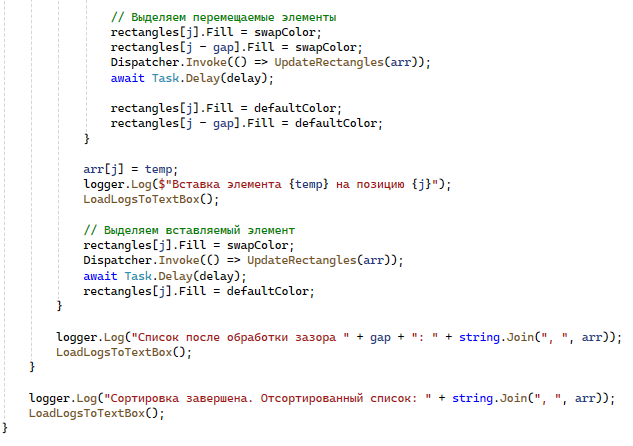


Рис.1.2.2 Код сортировки Шелла

Быстрая сортировка - это алгоритм "разделяй и властвуй", который рекурсивно сортирует массив, разделяя его на части относительно опорного элемента и объединяя отсортированные части.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 1.3.1 – 1.3.3.

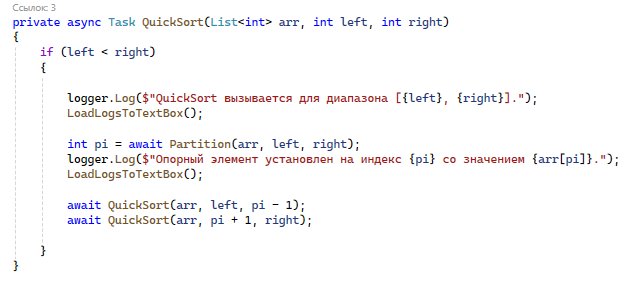


Рис.1.3.1 Код быстрой сортировки

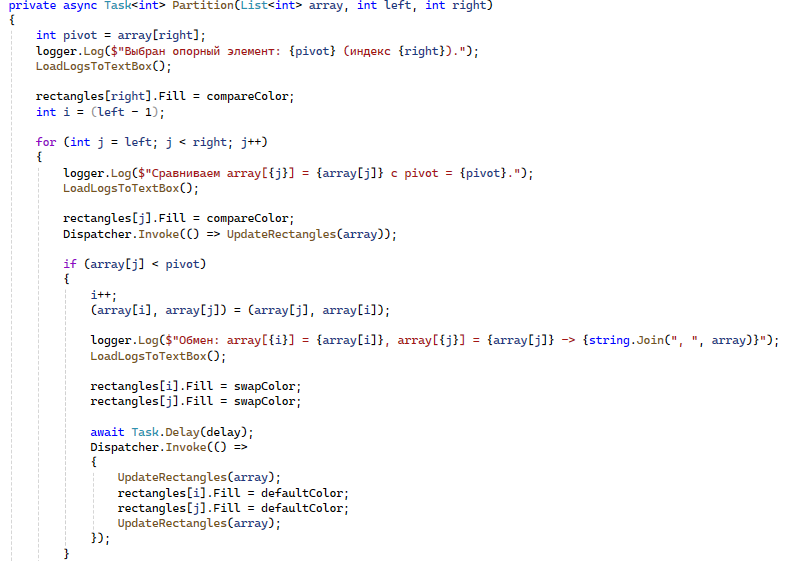


Рис.1.3.2 Код быстрой сортировки

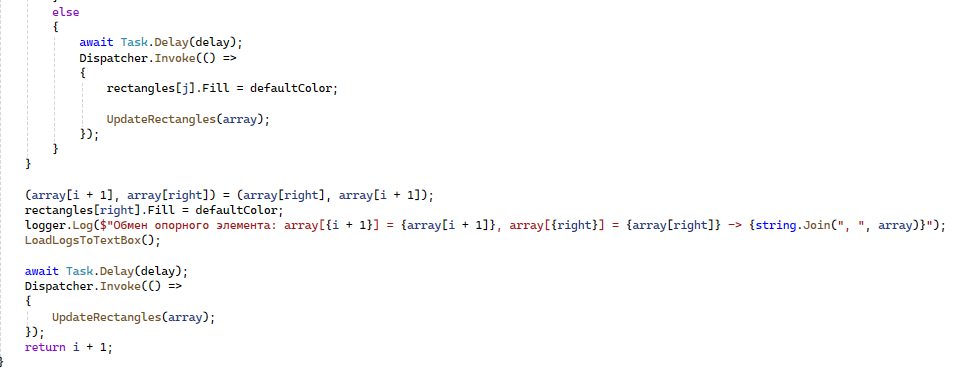


Рис.1.3.3 Код быстрой сортировки

Сортировка слиянием – это алгоритм который разделяет массив данных на две равные части, потом вызывает сам себя для каждой из этих частей и так до тех пор, пока массивы не станут единичными. Дальше эти подмассивы сравниваются со своей «половинкой» и в упорядоченом виде объединяются в новый отсортированый массив.

Данный алгоритм является внутренним, несмотря на то что есть его аналоги в виде внешних сортировок.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 1.4.1 – 1.4.4.

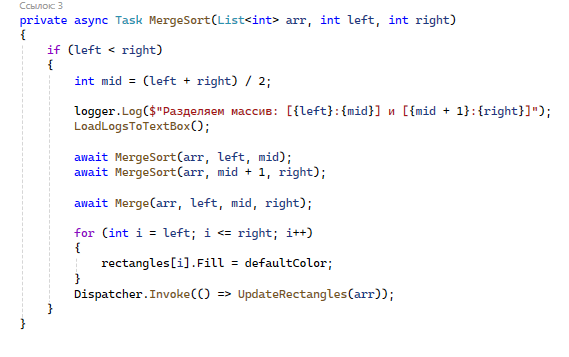


Рис.1.4.1 Код сортировки слиянием

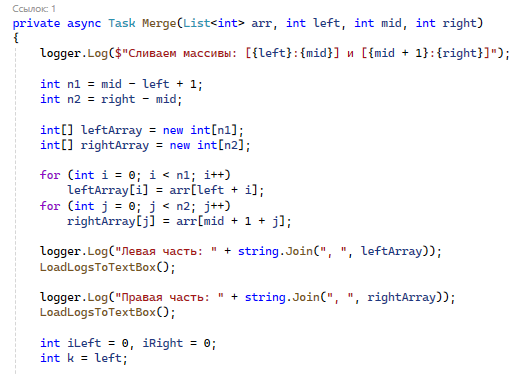


Рис.1.4.2 Код сортировки слиянием

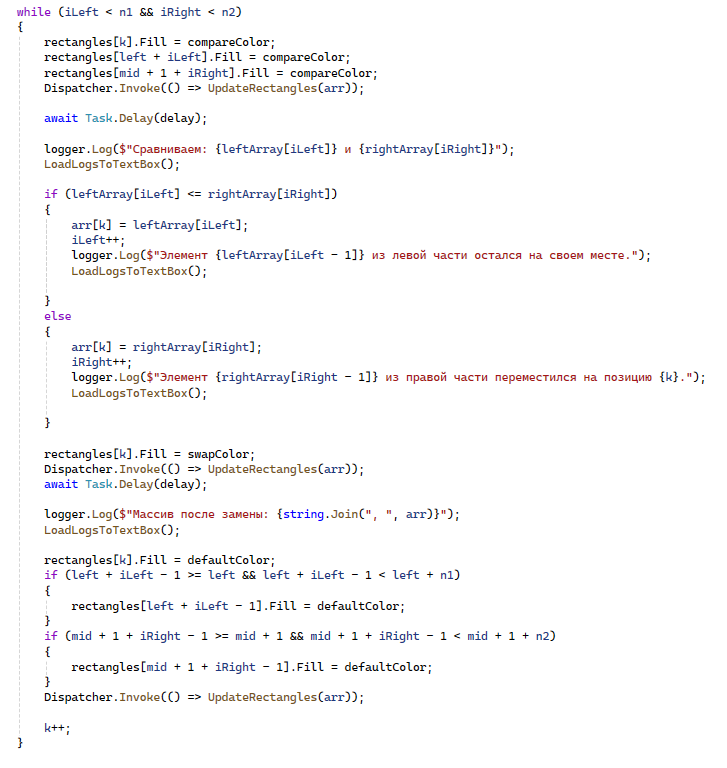


Рис.1.4.3 Код сортировки слиянием

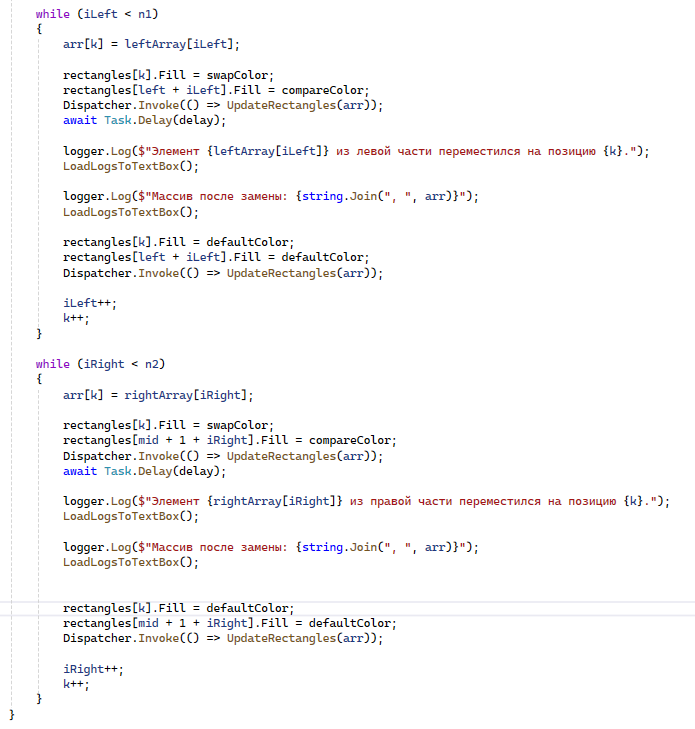


Рис.1.4.4 Код сортировки слиянием

Чтобы наглядно посмотреть на процесс сортировки был написан GUI для данной части (рис.1.5), где видно как проходит сортировка элементов, которые представлены в виде столбцов с соответствующей величиной. Так же есть окно, куда с определенной задержкой, которую выбирает пользователь, выводятся сопутствующие объяснения. Сам массив пользователь может ввести в верхнем окне ввода, через запятую или пробел. Выбрать желаемую сортировку можно в панели управления, в правой части окна.

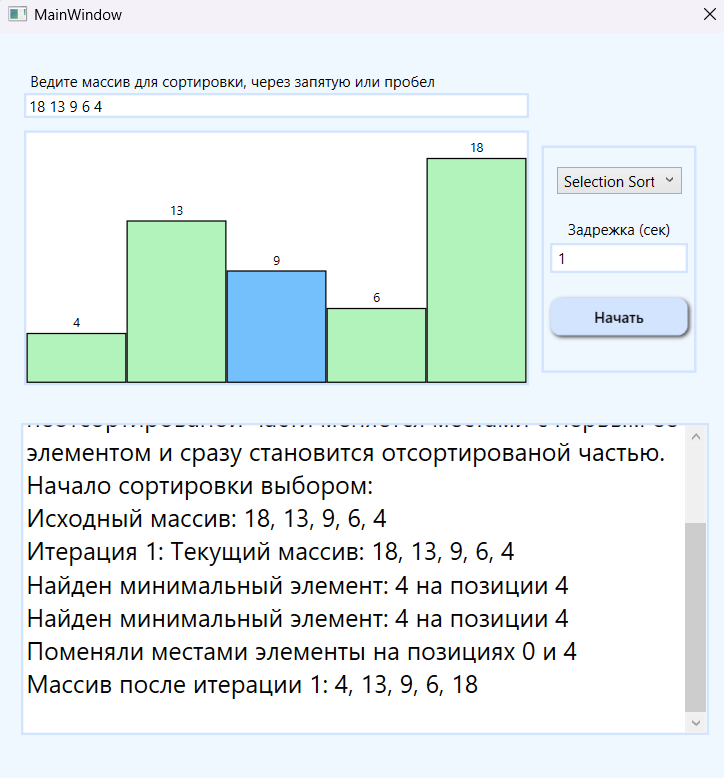


Рис.1.5 визуализация первого задания

Задание 2

Эта часть работы заключается в том, чтобы показать наглядно, и объяснить, как работают внешние сортировки, такие как прямое, естественное и многопутевое слияния.

Прямая сортировка слиянием:

Этот алгоритм сортирует массив данных путем рекурсивного разделения его на две половины, сортировки каждой половины отдельно и затем слияния отсортированных половин обратно в один массив. Процесс разделения и слияния продолжается до тех пор, пока не будет сформирован один полностью отсортированный массив.

Прямое слияние начинается с разделения исходного массива на две равные части. Если количество элементов в массиве нечетное, одна из частей будет содержать на один элемент больше. Этот процесс разделения продолжается рекурсивно для каждой половины до тех пор, пока не останутся только отдельные элементы или пустые массивы. Один элемент считается уже отсортированным, так как нет других элементов для сравнения.

После того как массив разделен на минимальные части, начинается процесс слияния. Две отсортированные половины сливаются в один отсортированный массив. Это делается путем сравнения элементов из каждой половины и добавления меньшего элемента в результирующий массив. Процесс сравнения и добавления продолжается до тех пор, пока не будут исчерпаны все элементы в обеих половинах. Если одна из половин исчерпана раньше, оставшиеся элементы другой половины просто добавляются в результирующий массив.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода, который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 2.1.1 – 2.1.2.

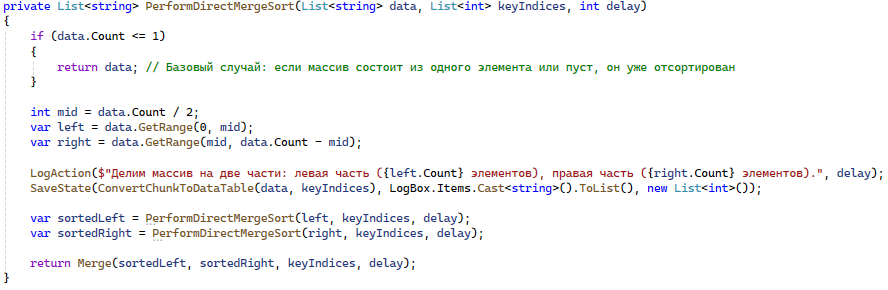


Рис.2.1.1 Код сортировки прямым слиянием

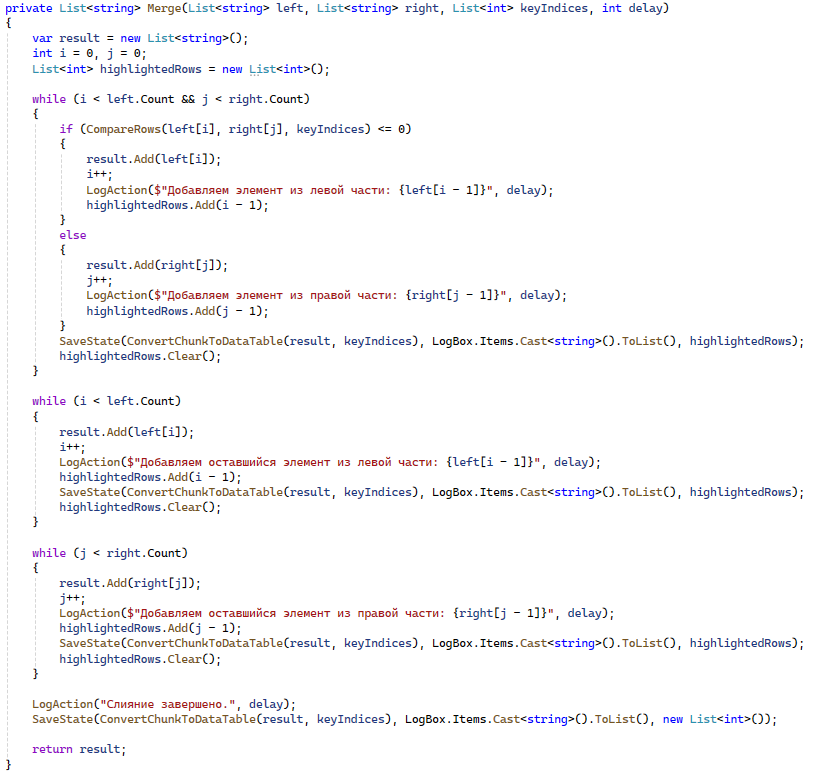


Рис.2.1.2 Код сортировки прямым слиянием

Естественное слияние:  
  
Принцип естественной сортировки слиянием, реализованный в данном коде, заключается в разбиении данных на естественные подпоследовательности (или "ранны"), которые затем сливаются в один отсортированный список. Этот метод сортировки называется естественной сортировкой слиянием, так как он использует естественные подпоследовательности, которые уже частично отсортированы, что позволяет уменьшить количество операций сравнения и перемещения.

Процесс начинается с разбиения исходных данных на естественные подпоследовательности. Это делается путем последовательного сравнения элементов данных. Если текущий элемент меньше или равен предыдущему, он добавляется в текущую подпоследовательность. Если текущий элемент больше предыдущего, текущая подпоследовательность завершается, и начинается новая подпоследовательность с текущего элемента. Этот процесс продолжается до тех пор, пока все данные не будут разбиты на подпоследовательности.

После разбиения данных на естественные подпоследовательности, начинается процесс их слияния. Слияние осуществляется с использованием очереди приоритетов, которая хранит минимальные элементы из каждой подпоследовательности. Процесс слияния начинается с добавления первых элементов каждой подпоследовательности в очередь приоритетов. Затем, пока очередь не пуста, извлекается минимальный элемент, который добавляется в итоговый список. Если в подпоследовательности, из которой был извлечен элемент, есть следующий элемент, он добавляется в очередь приоритетов. Этот процесс продолжается до тех пор, пока все элементы из всех подпоследовательностей не будут добавлены в итоговый список.

Таким образом, естественная сортировка слиянием сочетает в себе разбиение данных на естественные подпоследовательности и их последующее слияние с использованием очереди приоритетов. Этот подход позволяет эффективно обрабатывать данные, используя их естественные свойства, что уменьшает количество операций и улучшает производительность сортировки.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 2.2.1 – 2.2.4.

  
Рис.2.2.1 Код сортировки естественным слиянием

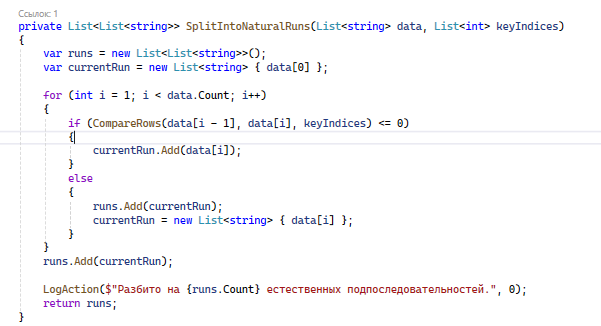


Рис.2.2.2 Код сортировки естественным слиянием



Рис.2.2.3 Код сортировки естественным слиянием



Рис.2.2.4 Код сортировки естественным слиянием

Многопутевое слияние:

Принцип многопутевой сортировки слиянием, реализованный в данном коде, заключается в параллельной сортировке и слиянии данных, разбитых на несколько частей (чанков). Этот метод позволяет эффективно обрабатывать большие объемы данных, используя параллелизм для ускорения процесса сортировки.

Прямое слияние начинается с разделения исходного массива на четыре равные части. Если количество элементов в массиве нечетное, одна из частей будет содержать на один элемент больше. Этот процесс разделения продолжается рекурсивно для каждой половины до тех пор, пока не останутся только отдельные элементы или пустые массивы. Один элемент считается уже отсортированным, так как нет других элементов для сравнения.

После того как массив разделен на минимальные части, начинается процесс слияния. Две отсортированные половины сливаются в один отсортированный массив. Это делается путем сравнения элементов из каждой половины и добавления меньшего элемента в результирующий массив. Процесс сравнения и добавления продолжается до тех пор, пока не будут исчерпаны все элементы в обеих половинах. Если одна из половин исчерпана раньше, оставшиеся элементы другой половины просто добавляются в результирующий массив.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода, который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 2.3.1 – 2.3.2.

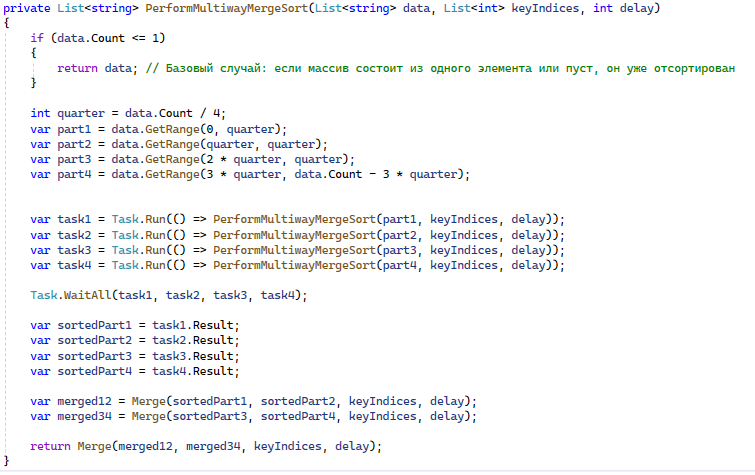


Рис.2.3.1 Код сортировки многопутевым слиянием

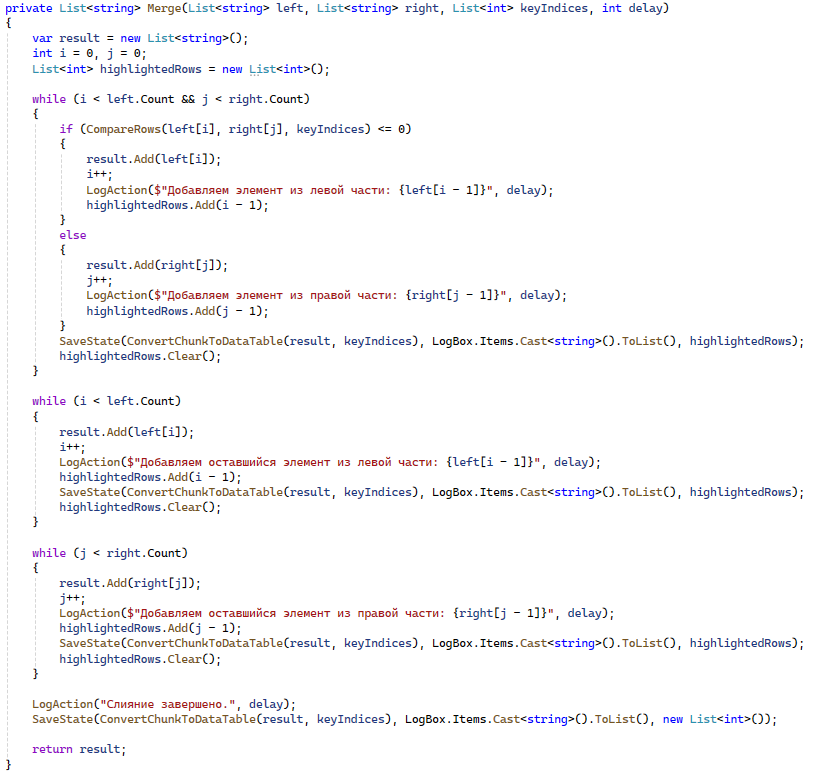


Рис.2.3.2 Код сортировки многопутевым слиянием

Чтобы наглядно посмотреть на процесс сортировки был написан GUI для данной части (рис.2.4), где видно как проходит сортировка элементов таблицы. Пользователю представлен выбор сортировки, ключа. Так же он может пошагово отслеживать изменения во время процесса, благодаря кнопкам «Вперед» и «Назад», или включить автопроигрывание.

В отдельных окнах выводится визуализация сортировки в виде состояния таблицы в текущий момент времени, текстовая документация процесса и финальный результат.

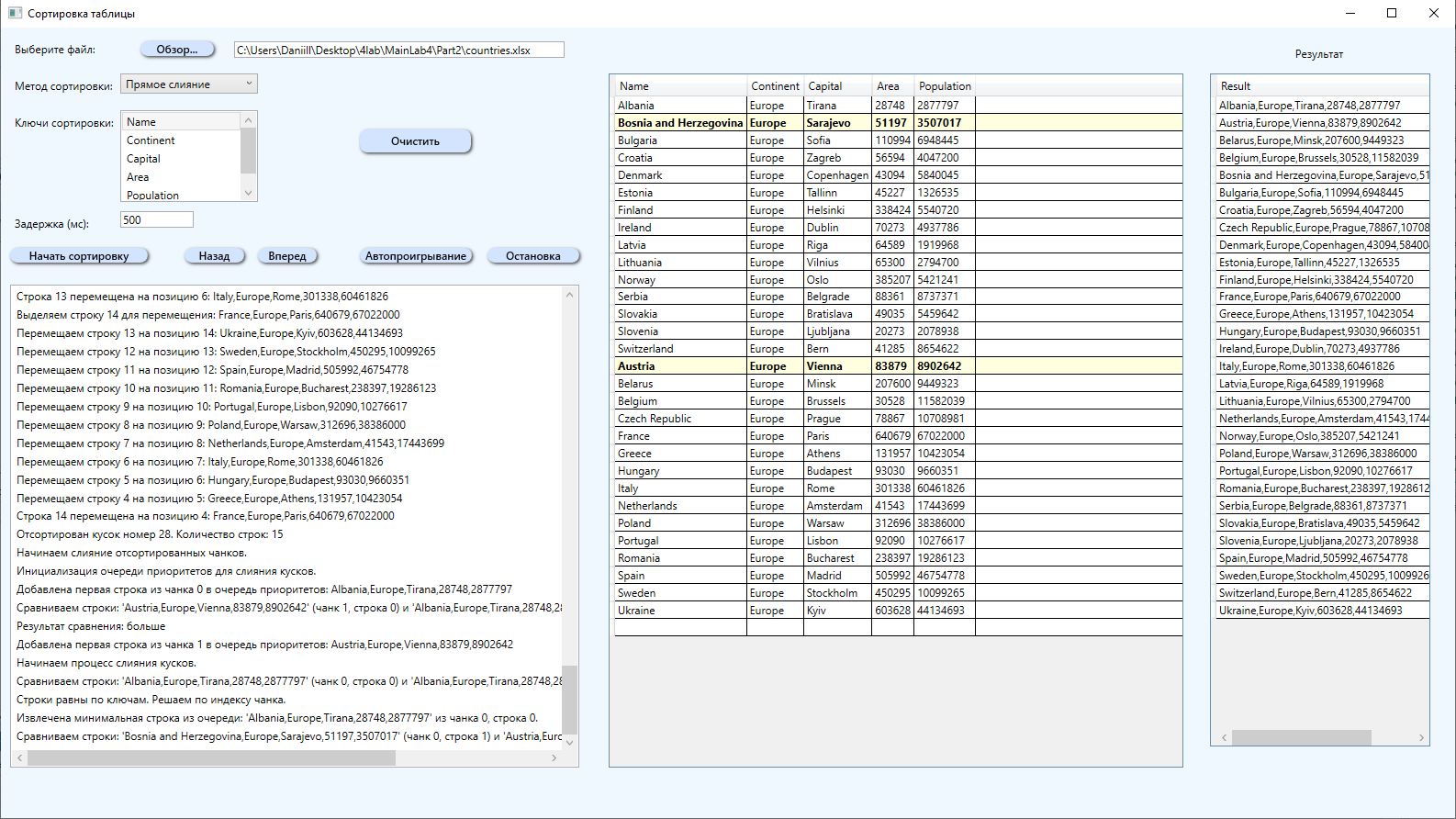


Рис.2.4 Часть 2 GUI

Задание 3

В рамках этой части были реализованы алгоритмы сортировки Modificated Buble Sort и Radix Sort.

Modificated Bubble Sort — это оптимизированная версия стандартного алгоритма сортировки пузырьком, которая позволяет завершить сортировку раньше, если массив уже отсортирован. Этот алгоритм прост в реализации и может быть эффективен для небольших массивов или частично отсортированных данных.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 3.1.1 – 3.1.2.



Рис.3.1.1 Код сортировки пузырьком.

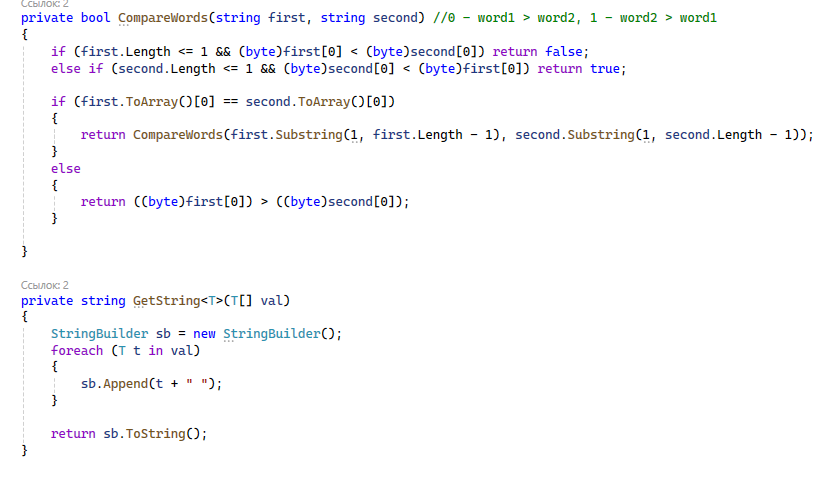


Рис.3.1.2 Код сортировки пузырьком.

Radix Sort - выполняет поразрядную сортировку, начиная с последнего символа и двигаясь к первому. Все строки выравниваются до одинаковой длины, заполняясь символом '0' (или пробелом) для корректного сравнения. На каждом этапе используется стабильная сортировка, например, CountingSort, для сортировки символов текущей позиции.

Чтобы доступно объяснить процесс сортировки, код был подвержен постоянному логированию. Так же внутри него есть множество вставок кода который отвечает за визуализацию сортировки.

Код сортировки представлен на рисунке 3.2.1 – 3.2.2.

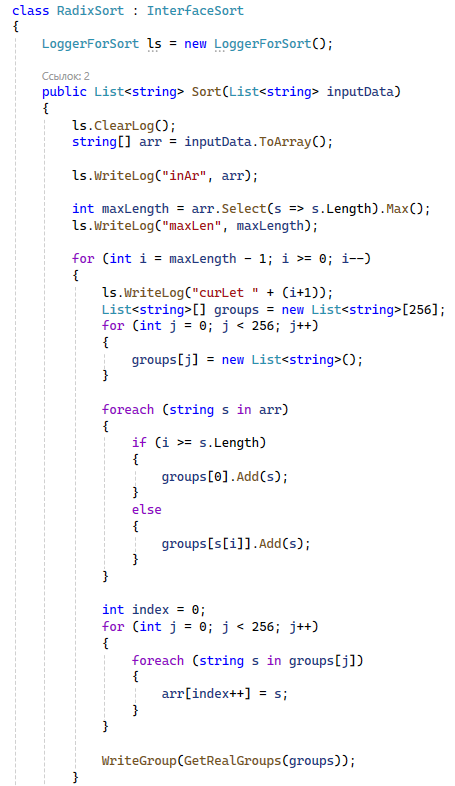


Рис.3.2.1 Код Radix Sort

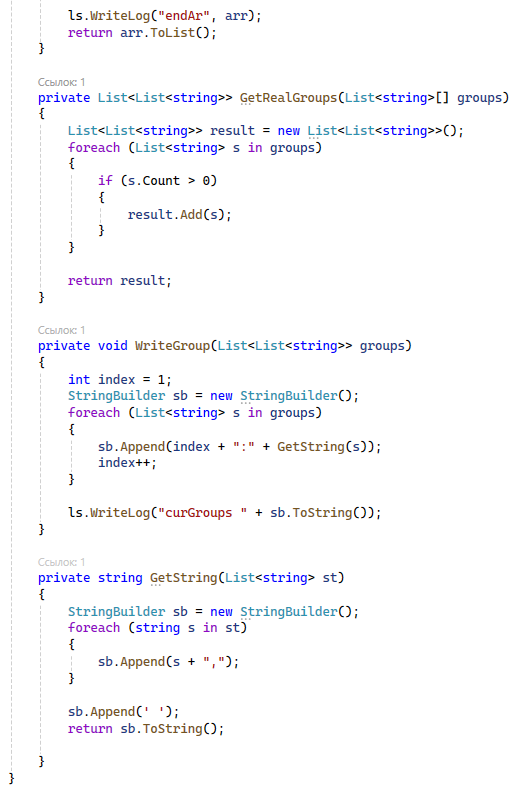


Рис.3.2.2 Код Radix Sort

Чтобы наглядно показать работу этих сортировок было создано два окна под Buble Sort (рис.3.3.1) и Radix Sort (рис.3.3.2), соответсвтенно. В каждом окне есть возможность выгрузить файл с массивом данных с компьютера. Так же пользователь может отслеживать состояния массива прямо во время сортировки, переключая шаги

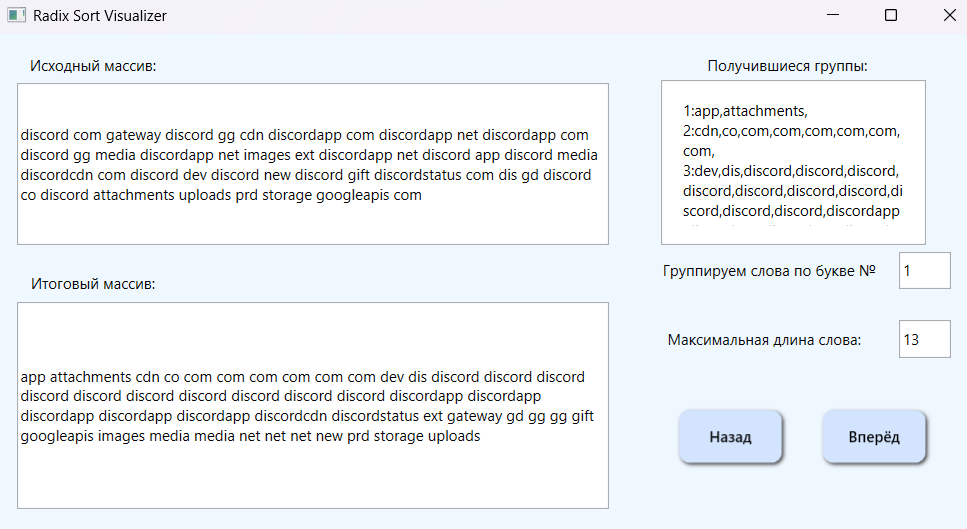


Рис.3.3.2 Визуализация Radix Sort