Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировки»**

**Выполнила**:

студентка группы 3821Б1ПМ2

Домрачева А.В.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Сортировка данных – это упорядочивание их по какому-либо признаку. Она способна упростить работу с информацией в несколько раз.

Задачей данной лабораторной работы было реализовать четыре алгоритма сортировки разной сложности: простую сортировку вставками, быструю сортировку Хоара, сортировку слиянием и поразрядную сортировку. Программы должны производить сортировку массивов, которые содержат данные типа float. Затем необходимо подсчитать число перестановок и количество сравнений в каждом алгоритме и составить график, иллюстрирующий временную сложность алгоритма.

# Метод решения

**Сортировка вставками***.* Программа получает на вход неотсортированный массив. В нём уже есть часть, которая отсортирована (для начала можно принять, что это первый элемент массива). Из оставшейся неотсортированной части по порядку выбираются элементы и вставляются в отсортированную часть на своё место. Так происходит до тех пор, пока весь массив не будет отсортирован. Выводится отсортированный массив.

**Быстрая сортировка Хоара**. Программа получает на вход неотсортированный массив. Выбирается опорный элемент. Массив делится на две части: в левую попадают элементы, которые меньше опорного элемента, а в правую – которые больше или равны опорному элементу. Затем вызывается рекурсивная сортировка каждого из подмассивов, т.е. в каждой части исходного массива выбирается опорный элемент и элементы сортируются уже относительно него. Так происходит до тех пор, пока исходный массив не будет отсортирован. Отсортированный массив выводится программой.

**Сортировка слиянием**. Программа получает на вход неотсортированный массив. Если он состоит из одного элемента, то автоматически считается отсортированным, и программа завершает работу. Иначе массив делится на два подмассива, а каждый подмассив ещё на два подмассива до тех пор, пока длина каждого подмассива не будет равна единице. Затем два единичных подмассива сливаются в общий массив, при этом из каждого такого массива выбирается меньший элемент и ставится в свободную левую ячейку результирующего массива. После чего из двух результирующих массивов собирается третий и так далее до конца массивов. Если один массив закончится раньше, то в конец результирующего массива дописываются оставшиеся элементы другого массива. В завершении результирующий массив будет отсортированным, он переписывается в исходный и выводится программой.

**Поразрядная сортировка**. Программа получает на вход неотсортированный массив. Сортировка происходит по разрядам. Сначала массив сортируется по младшему разряду, затем по старшим. Для каждого из разрядов проводится сортировка подсчётом. В результате без знакового числа становятся в отсортированном порядке, а знаковые, из-за особенности их кодировки, записываются в конец кода в обратном порядке. Далее такой массив переписывается в новый в нужном порядке. В завершение работы программы получается отсортированный массив, он же и выводится на экран.

# Руководство пользователя

Программы получают входные данные – массив, который заполняется с помощью случайных чисел. Такой массив состоит из чисел типа float, и в него входят как знаковые, так и без знаковые числа. После работы программы пользователь получает исходный (неотсортированный) массив, отсортированный массив, а также может видеть число перестановок и количество сравнений, выполненных программой.

# Описание программной реализации

Задача проекта состоит в реализации четырех алгоритмов сортировок, каждый из которых написан отдельным файлом.

Файл InsertionSort включает в себя одну функция main, которая заполняет массив числами, реализует сортировку вставками, а также ведёт подсчёт количества перестановок и сравнений.

Файл QuickSort состоит из двух функций: main заполняет массив случайными числами, затем вызывает функцию qsort, которая реализует быструю сортировку и возвращает отсортированный массив обратно в main, откуда он и выводится. Подсчёт перестановок и сравнений ведётся с помощью глобальных переменных.

Файл MergeSort также содержит функцию main, которая заполняет массив случайными числами, а затем, вызывая функции сортировки, выводит уже отсортированный массив. Функция MergeSort делит массив на два подмассива, а затем для каждого из подмассивов вызывает функцию Merge, которая сравнивает подмассивы и собирает их в один отсортированный массив (сортировка слиянием). Число перестановок и сравнений считается с помощью глобальных переменных.

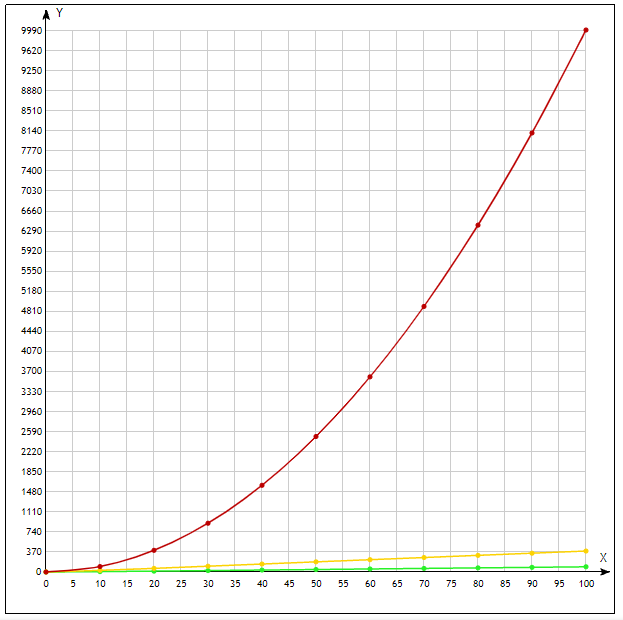
Файл RadixSort в функции main создает массив с рандомными числами, вызывает функцию поразрядной сортировки, а затем выводит уже отсортированный массив. Функция поразрядной сортировки называется radixSort, для её работы требуются ещё три функции: createCounters, radixPass и signedRadixLastPass. Первая - заполняет массив counters счётчиками. Вторая – сортирует массив так, что в итоге без знакового числа сортируются в порядке возрастания, а знаковые в порядке убывания и записываются в конец массива. Третья функция записывает числа в правильном порядке, на этом сортировка заканчиваются. Глобальная переменная global\_p считает число перестановок в алгоритме поразрядной сортировки в зависимости от количества элементов исходного массива.

# Подтверждение корректности

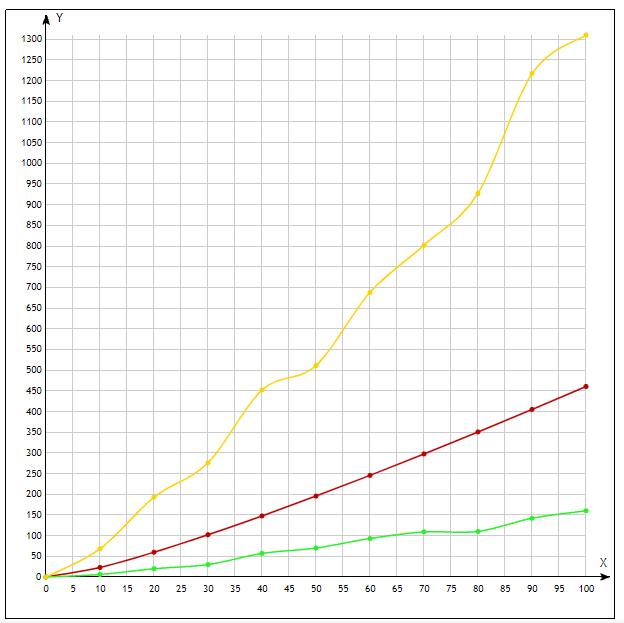
Для подтверждения корректности в программе выводятся исходный массив и уже отсортированный, для того, чтобы пользователь смог сам визуально проверить правильность сортировки. Также после написания кода несколько исходных массивов были проверены с помощью простой сортировки вставками, результат вышел тот же, что и у проверяемой программы. Из этого можно сделать вывод, что сортировки работают корректно.

Также для подтверждения корректности использовались графики, представляющие собой зависимость числа перестановок (зеленый график) и числа сравнений (жёлтый график) от количества элементов массива. Если графики получаются похожи на график теоретической сложность алгоритма (красный график), но с некоторым коэффициентом, то считается, что сортировка работает корректно. Далее представлены графики различных сортировок.

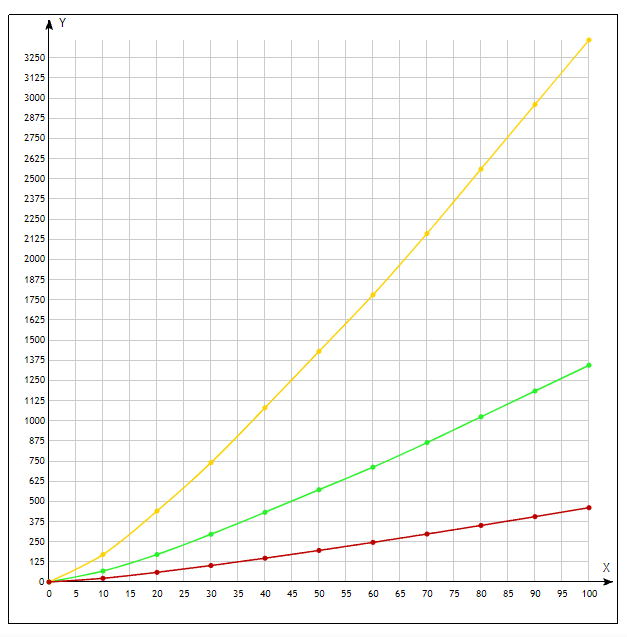
**Сортировка вставками**



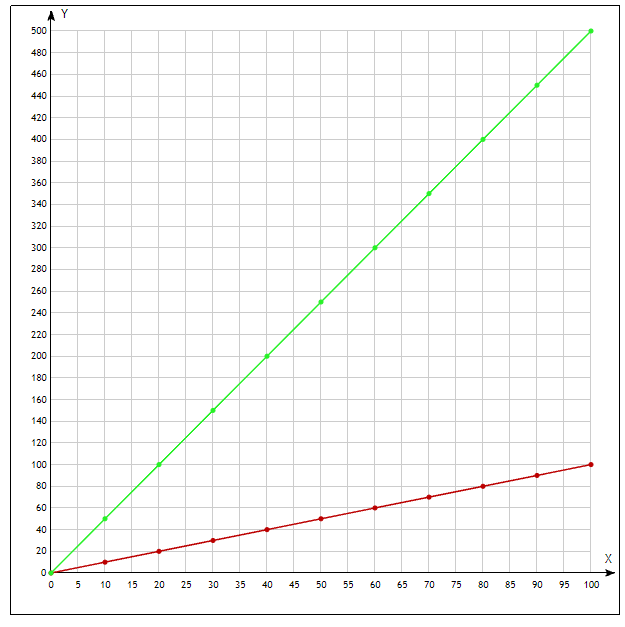
**Быстрая сортировка**

****

**Сортировка слиянием**



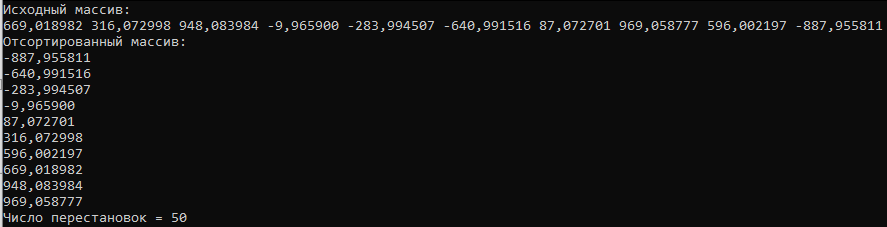
**Поразрядная сортировка**



Существенное отличие графиков сравнений и перестановок в быстрой сортировке объясняется тем, что эта сортировка имеет худший случай (n2), средний случай (n\*log(n)) и лучший случай (n\*log(n)). Остальные графики сортировок схожи с графиками своей теоретической сложности, так что можно сделать вывод, что сортировки работают корректно.

# Результаты экспериментов

По данным экспериментов видно, что сортировки работают корректно. Программы, принимая на вход массив случайных чисел, выводят уже отсортированный массив по возрастанию. Программы правильно сортируют как знаковые, так и без знаковые числа.



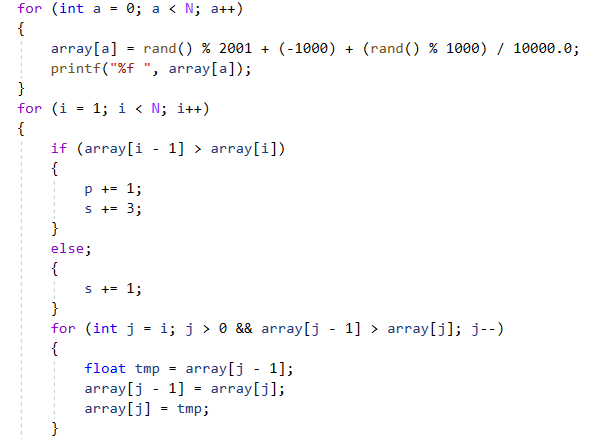
# Заключение

В результате проведённой лабораторной работы были написаны четыре программы: простая сортировка вставками, быстрая сортировка Хоара, сортировка слиянием и поразрядная сортировка. Программы работают корректно, что подтверждается экспериментами. Графики, построенные по результатам работы программ, иллюстрируют теоретическую сложность.

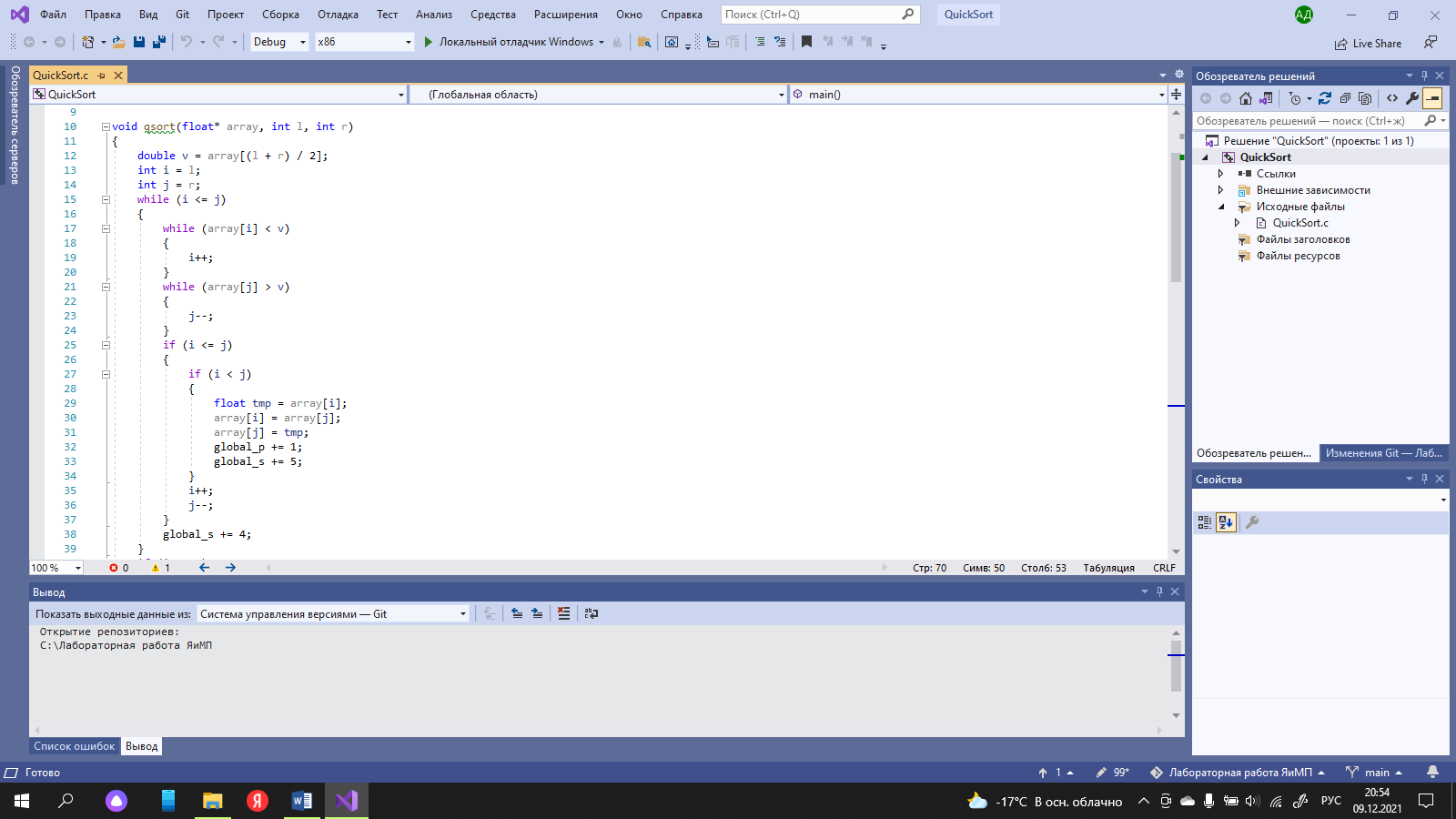
Таким образом, задача данной лабораторной работы выполнена полностью.

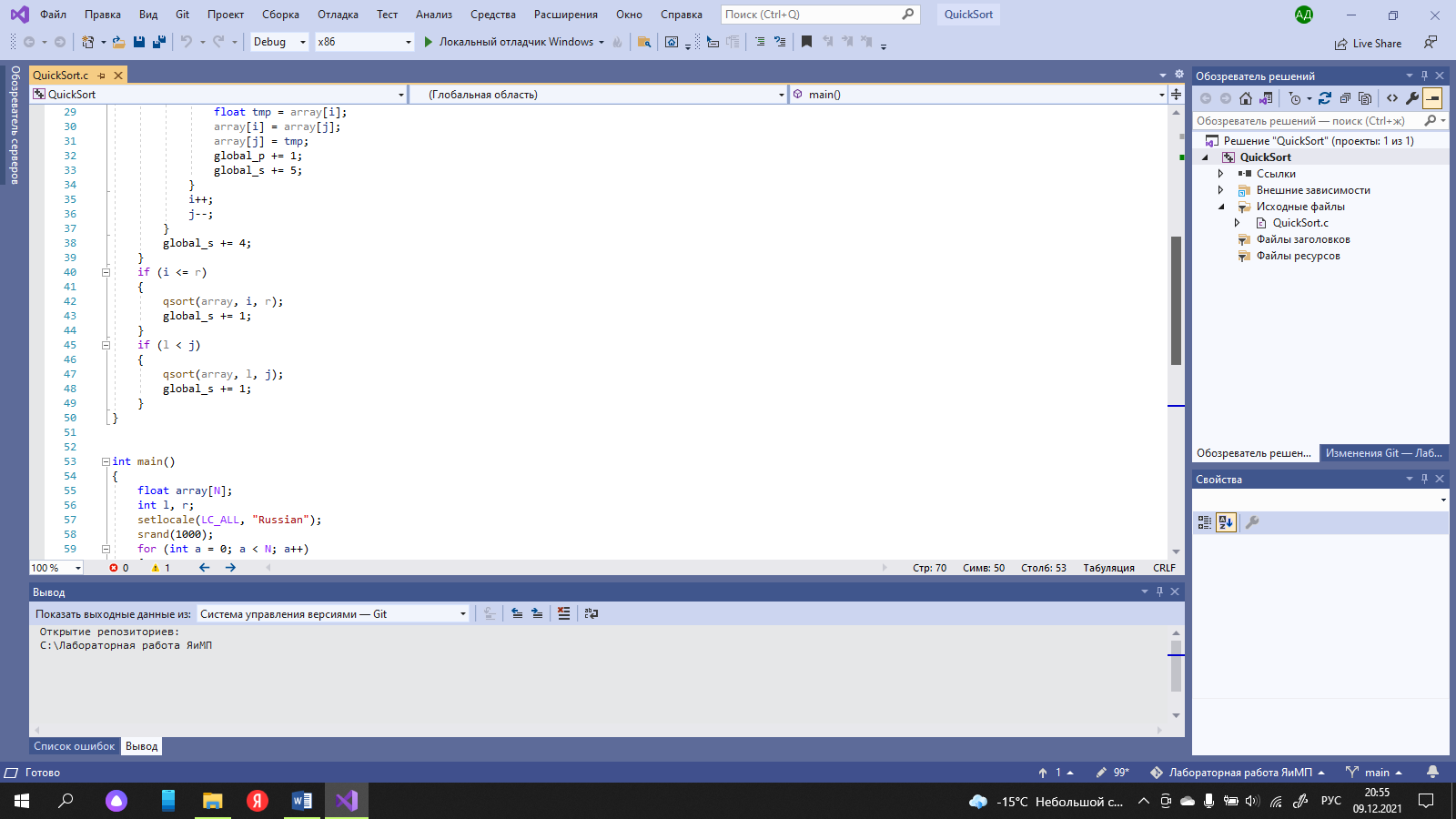
# Приложение

Сортировка вставками

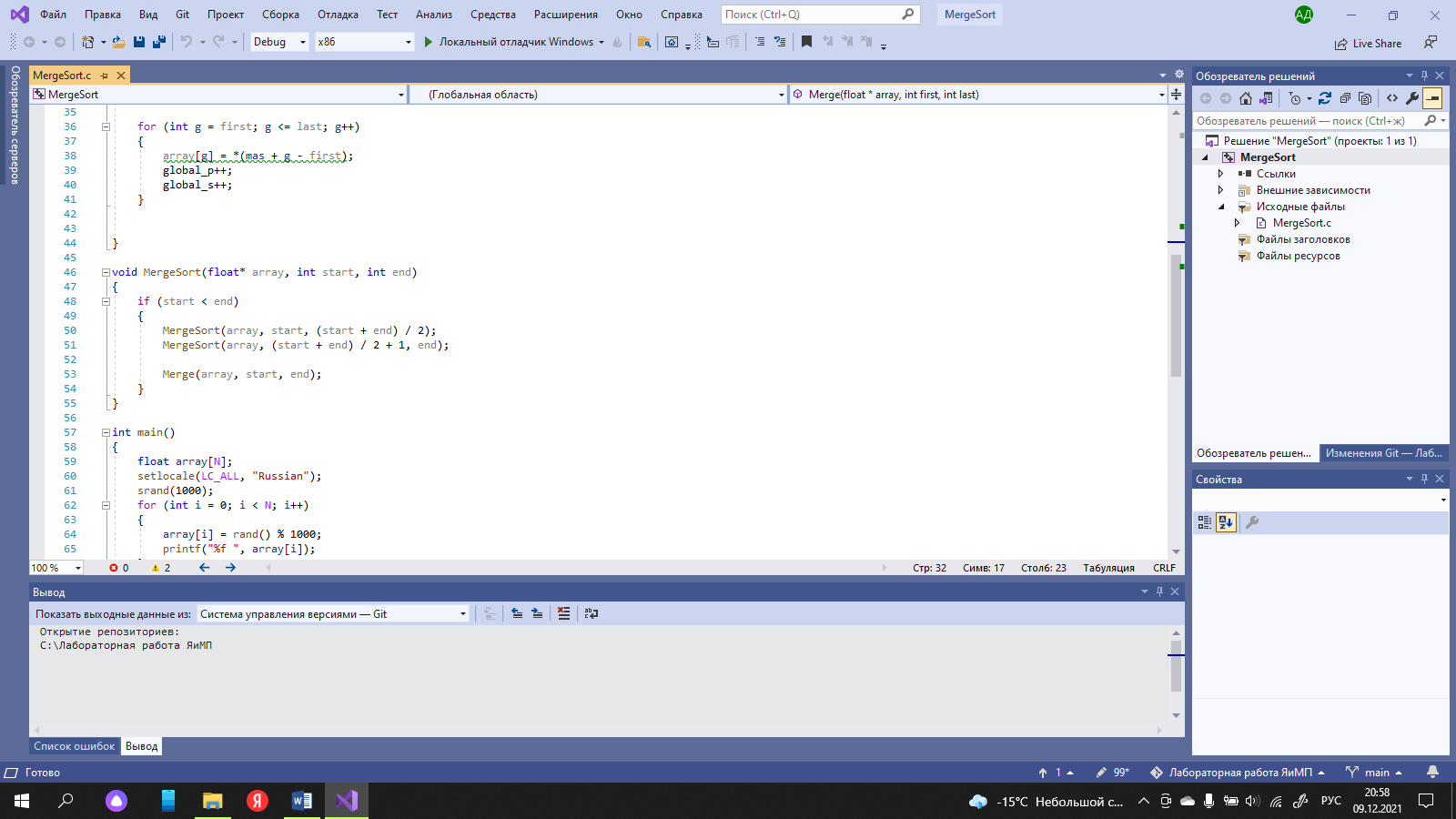


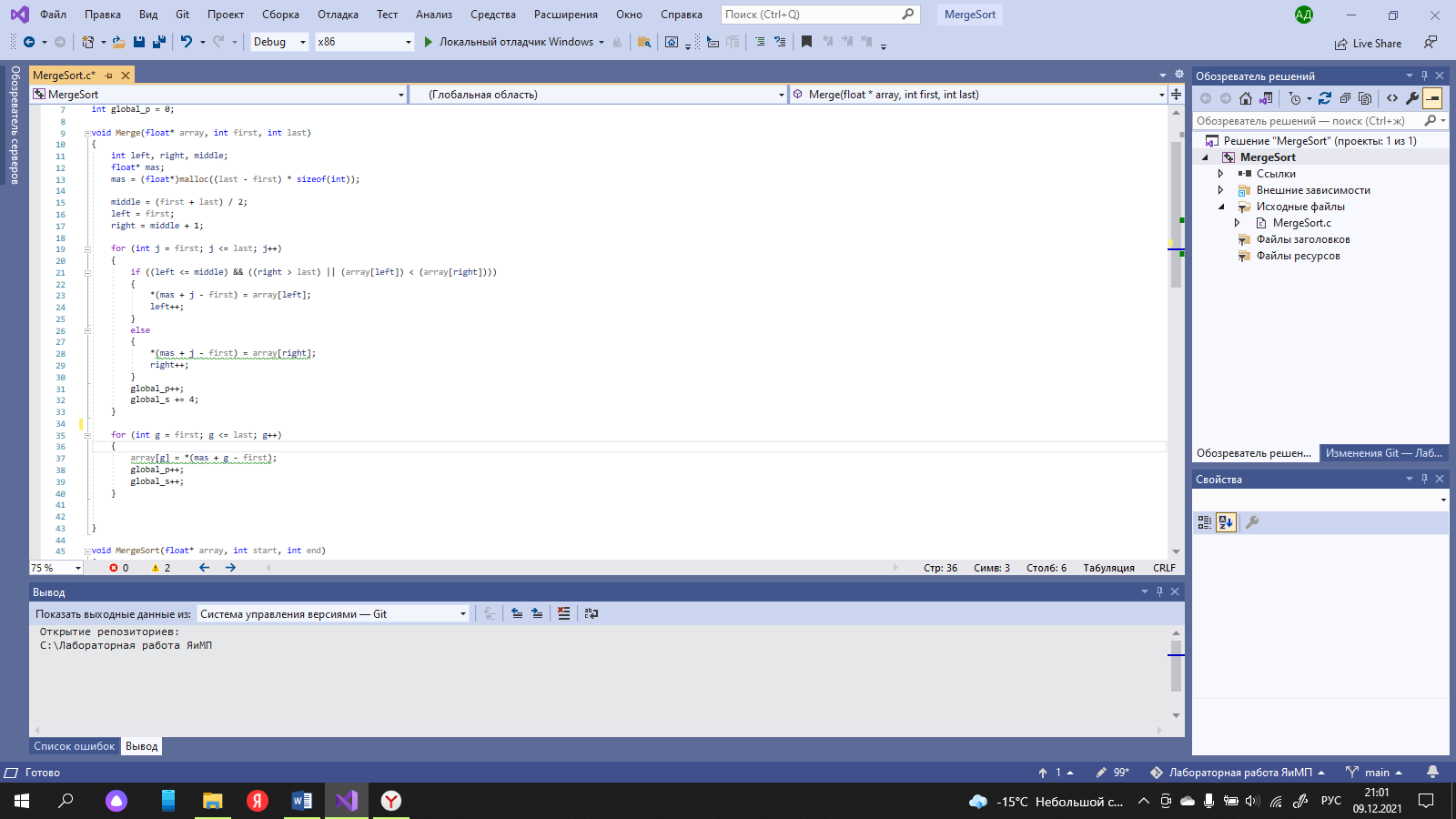
Быстрая сортировка





Сортировка слиянием





Поразрядная сортировка

