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**

**«Методы сортировки»**

**Выполнил**:

студент группы 382003-1

Линев Евгений Алексеевич

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2020

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 6](#_Toc26962564)

[Описание программной реализации 8](#_Toc26962565)

[Подтверждение корректности 14](#_Toc26962566)

[Результаты экспериментов 15](#_Toc26962567)

[Заключение 17](#_Toc26962568)

[Приложение 18](#_Toc26962569)

# Постановка задачи

При выполнении данной работы требовалось написать программу на языке С, которая отсортировала бы введенный пользователем или сгенерированный случайным образом массив с элементами типа float одним из следующих способов:

* Сортировкой выбором;
* Сортировкой расческой;
* Сортировкой слиянием;
* Поразрядной сортировкой.

Также необходимо замерить количество сравнений и перестановок при использовании каждой сортировки и провести тесты, показывающие теоретическую сложность алгоритмов.

# Метод решения

1. **Сортировка выбором:**

Идея заключается в том, что находится минимальный элемент в массиве, который ставится на первое место. Затем среди оставшихся элементов находится новый минимальный, который становится вторым элементом в массиве, и так далее.

1. **Сортировка расческой:**

Сортировка является некоей модификацией сортировки пузырьком. Сравниваются не соседние элементы, а те, которые стоят на каком-то определенном расстоянии друг от друга. Так маленькие элементы в конце списка быстро перейдут в начало.

Изначальное расстояние равно целой части от деления массива на 2, а затем постепенно уменьшается, пока не дойдет до 1. Таким образом вначале элементы двигаются на большие расстояния и становятся ближе к своим «правильным» местам, а при довольно маленьких значениях массив «разглаживается» и элементы становятся на свои места. После окончания процесса требуется сортировка пузырьком (обычно требуется лишь один проход).

1. **Сортировка слиянием**

Основная идея – объединять уже отсортированные массивы и из них строить новый, который уже будет отсортирован и состоять из элементов меньших массивов.

Вводимый массив разбивается на две примерно равные части (левая и правая), после чего каждая из них тоже разбивается на части и так далее. В конце разбиеиия получаются массивы, состоящие из 1 элемента, их можно считать базой для объединения, ведь такой массив точно отсортирован.

Затем соединяются отсортированные массивы. Выделяется дополнительная память под результирующий массив, в которую будут переносится нужные элементы. На каждом шаге берем наименьший из первых элементов массивов и заносим в результирующий, а затем увеличиваем счетчики результирующего массива и того, из которого брали элемент. Если один из двух массивов заканчивается, то результирующий до конца заполняется элементами из другого.

1. **Поразрядная сортировка**

Идея этой сортировки заключается в том, что сравниваются разряды вводимых значений. Для каждой цифры справа налево используется устойчивая сортировка, которая не меняет порядок следования элементов с одинаковым значением разряда.

В данном случае в качестве одного разряда берется один байт. Необходимо помнить, что числа в памяти записываются в обратном порядке. То есть ведущие нули расположены в конце. Кроме того, надо учитывать, что отрицательные числа будут расположены после положительных, так как первый бит в старшем байте у них равен 1. Поэтому в конце следует провести еще одну перестановку в массиве.

# Руководство пользователя

Специально для пользователя разработан удобный многофункциональный интерфейс, позволяющий ему выбрать:

* Размер массива;
* Способ заполнения массива (автозаполнение или ввод данных вручную);
* Вид используемой сортировки.

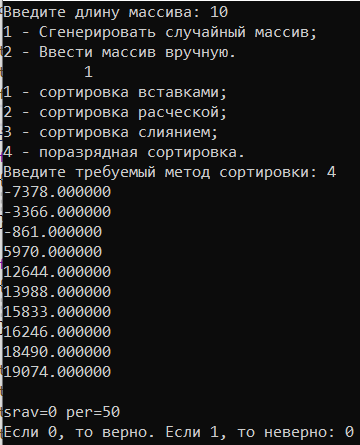
Сначала вводится длина массива, затем выбирается, каким способом он будет заполнен. В этом пункте следующие команды:

1. Сгенерировать случайный массив
2. Ввести массив вручную.

При выборе второго варианта каждый вводимый элемент печатается с новой строки. Далее пользователь может выбрать сортировку, которая будет упорядочивать элементы.

1. Сортировка вставками;
2. Сортировка расческой;
3. Сортировка слиянием;
4. Поразрядная сортировка.

После выбора сортировки выводится полученный массив – каждый элемент с новой строки. На следующей строке показывается количество сравнений и перестановок элементов: crav – сравнения, per – перестановки. Затем выводится правильность полученного результата (подробнее в разделе «подтверждение корректности»). Если «0», то сортировка прошла успешно, если «1», то массив не упорядочен по возрастанию.

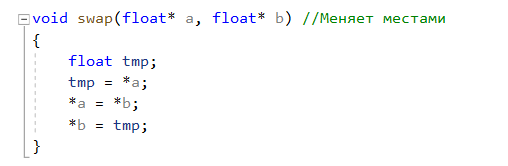


Как видно на картинке, интерфейс максимально понятен и удобен.

# Описание программной реализации

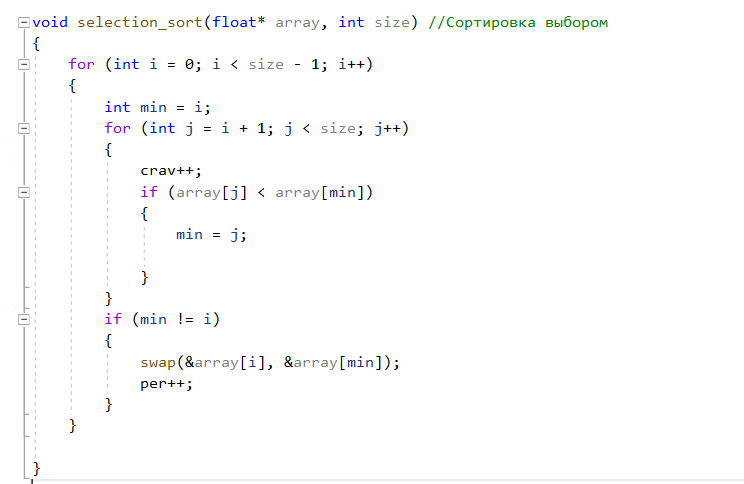
В реализации содержатся различные функции, которые используются при различных обстоятельствах.

1. Swap



Данная функция меняет два элемента местами. Указатели позволяют менять местами не локальные, а глобальные переменные. Будет использоваться в сортировке выбором и сортировке расческой.

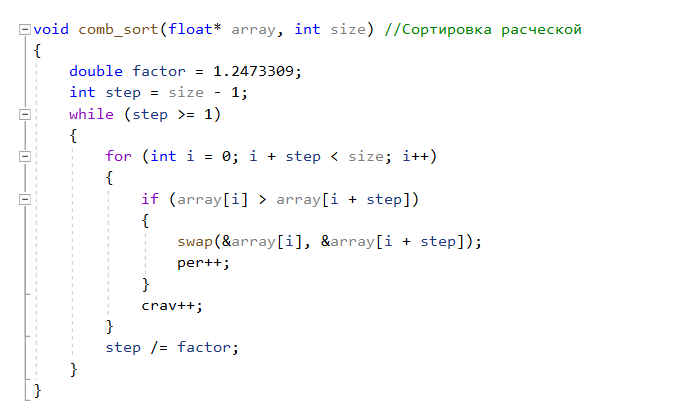
1. Реализация сортировки выбором



В функцию поступает исходный массив и его размер. Создается цикл, отвечающий за проход по всему массиву начиная с первого проверяемого элемента. Переменная min отвечает за номер минимального элемента из всех проверяемых на данном этапе и в начале равна номеру первого элемента. Затем создается еще один (внутренний) цикл, который начинается со второго элемента и идет до конца массива. Если обнаруживается элемент, который меньше всех предыдущих, то переменная min становится равна его номеру в массиве.

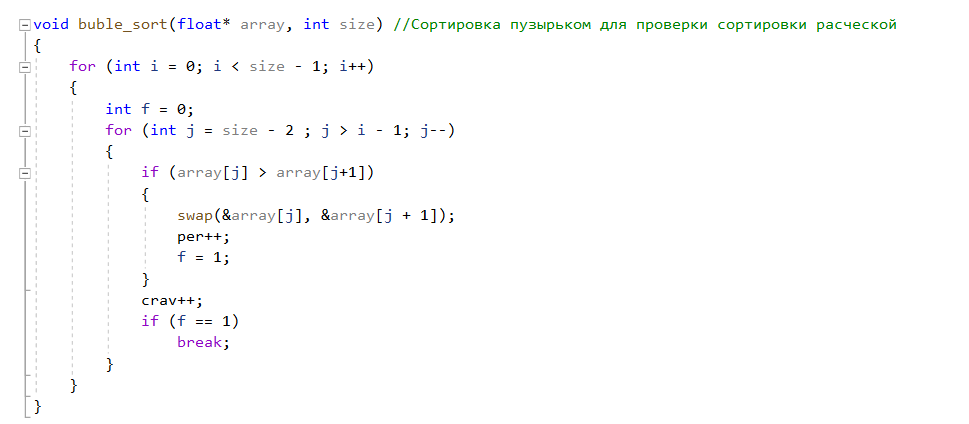
После окончания внутреннего цикла сравниваются первый проверяемый элемент массива и элемент с номером min. Если они отличаются, то элемент с номером min меньше и они меняются местами. Так на первом месте оказывается минимальный элемент массива и его уже не нужно проверять. Счетчик внешнего цикла смещается на единицу вправо, а следовательно и счетчик внутреннего цикла тоже. Проверяются оставшиеся элементы, среди которых тоже находится наименьший и так далее, пока алгоритм не обработает весь массив.

1. Реализация сортировки расческой



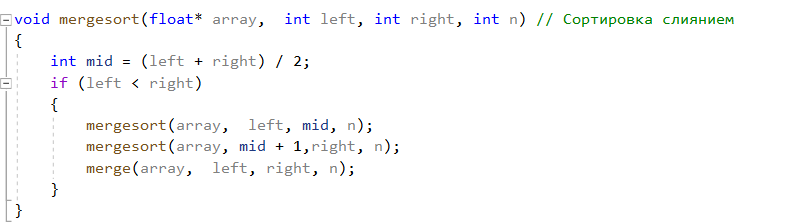
В функцию поступает исходный массив и его размер. Заводится переменная factor, отвечающая за сокращение расстояния между проверяемыми элементами, а также переменная step, которая равна размеру массива. Далее создается цикл, который остановится, когда step станет меньше единицы, то есть когда элемент будет сравниваться сам с собой.

Далее осуществляется проход по массиву и сравниваются элементы, которые стоят на расстоянии step друг от друга. Если какой-то элемент с большим номером является меньше элемента с меньшим номером, то они меняются местами. Как только массив заканчивается переменная step уменьшается – следовательно, уменьшается расстояние между сравниваемыми элементами – и осуществляется еще один проход по массиву. Как уже было сказано, это продолжается до тех пор, пока step не становится 0.



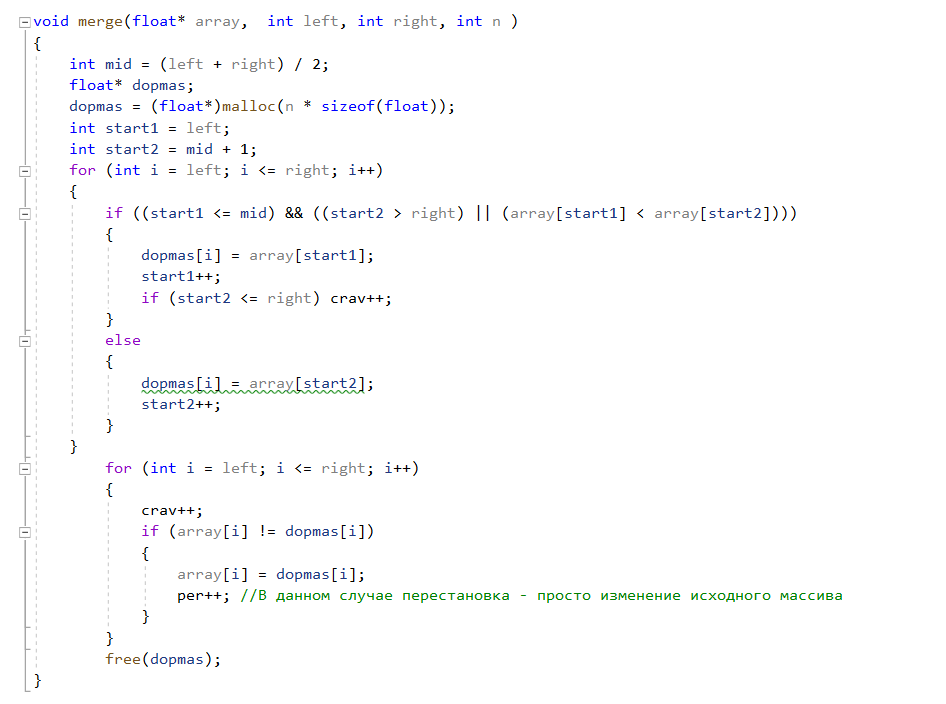
После окончания алгоритма массив еще раз сортируется пузырьком (чаще всего достаточно лишь одного прохода). Однако функция сортировки пузырьком не входит в данную функцию, а вызывается уже в основной части.

1. Реализация сортировки слиянием



В функцию передаются: массив, номер первого элемента, номер последнего элемента, размер массива. Далее ищется середина – ее значение заносится в переменную - и если номер левого элемента меньше номера правого (или, по-другому, они не совпадают), то функция вызывает саму себя. При первом рекурсивном вызове левый элемент остается прежним, а правый принимает значение переменной mid, то есть мы делим данный массив на два и рассматриваем левую часть. При втором рекурсивном вызове меняется левый элемент а правый остается прежним, то есть теперь рассматривается правая часть.

В итоге массив будет разобран на отдельные элементы, которые потом будут собираться и сортироваться на каждом этапе сборки. Объединять уже отсортированные фрагменты двух частей массива в новый отсортированный и состоящий из элементов предыдущих двух частей массив будет функция merge.

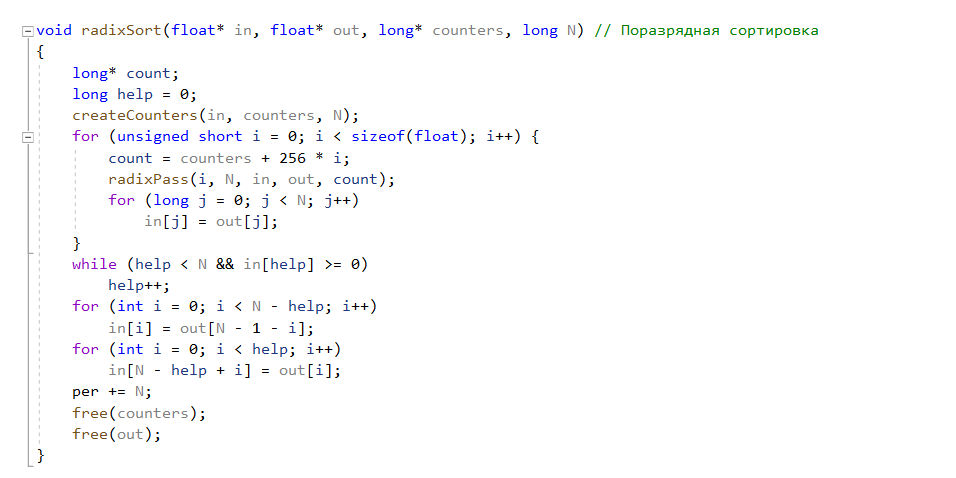


В эту функцию передаются: исходный массив, левый и правый элементы подмассива, сформированные в mergesort и длина исходного массива. Снова ищется центральный элемент, а также создается дополнительный массив, в который будут записываться сортированные значения. Далее идем по основному массиву по той области, которую мы задали в mergesort. В этой области у нас есть две уже отсортированные части, которые не отсортированы относительно друг друга.

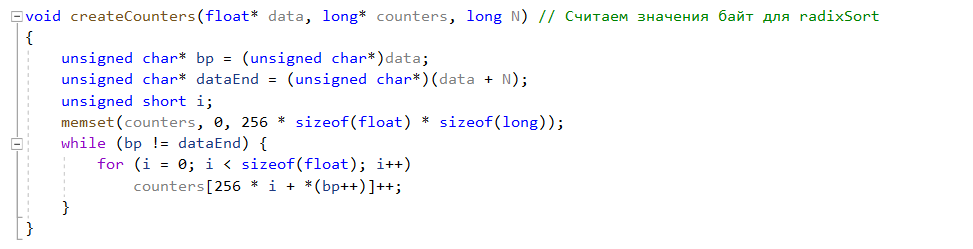
Сравниваем между собой первые элементы этих частей. Выбираем меньший из них и заносим в созданный нами дополнительный массив. В той части, из которой взят элемент, увеличиваем счетчик на 1, то есть теперь при сравнении будем брать следующий элемент. Если вдруг в одном из двух подмассивов закончатся элементы, это значит, что все оставшиеся элементы второго подмассива больше последнего элемента первого, поэтому дополнительный массив заполняется всеми оставшимися элементами второго подмассива.

Когда счетчик дойдет до предела заданной области, цикл завершится, и остается лишь перенести значения из дополнительного в основной. Создается еще один цикл, в котором это реализуется.

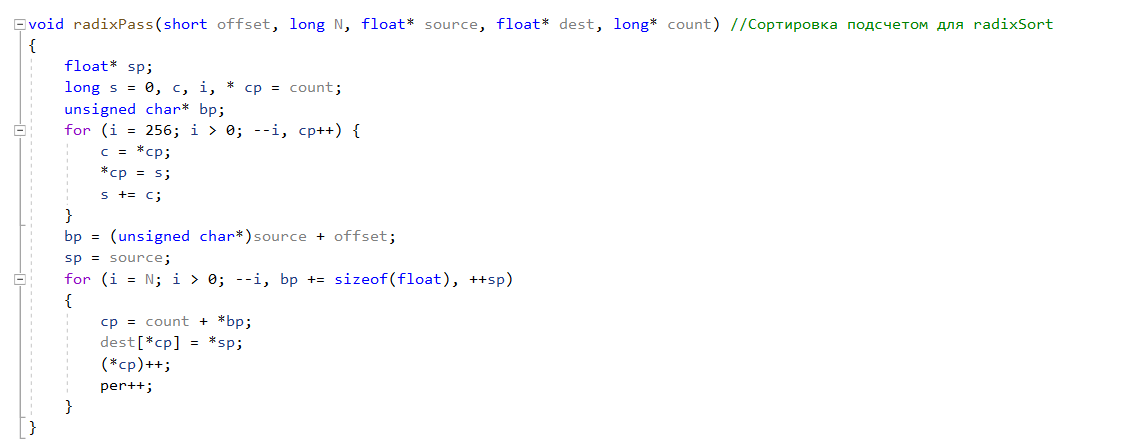
1. Реализация поразрядной сортировки



В функцию передаются: основной массив, дополнительный массив с размером основного, дополнительный массив с размером (256\*sizeof(float)\*sizeof(long)). Далее заполняется второй дополнительный массив с помощью функции createCounters.



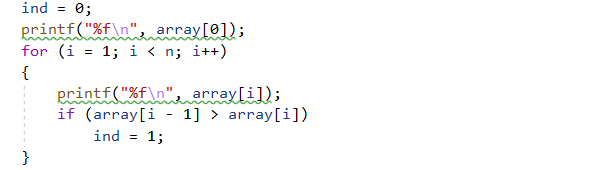
В эту функцию передаются значения исходного массива, второго дополнительного массива (пока еще пустого) и длина исходного массива. Эта функция предназначена для записи в дополнительный массив элементов из основного но в побитовом виде.

После заполнения возвращаемся в основную функцию и начинаем сортировать массив, в котором записаны побитовые значения. Делается это с помощью сортировки подсчетом. 

Здесь происходит сортировка массива по одному байту и итог записывается в дополнительный массив, который по размеру равен основному. После этого значения из дополнительного массива переносятся в основной и сортировка проводится уже по следующему байту. В итоге получается отсортированный массив, вот только отрицательные элементы стоят после положительных. Поэтому сначала ищется первый отрицательный элемент в массиве, а потом с помощью знания номера этого элемента производится перестановка отрицательных чисел в начало массива.

# Подтверждение корректности

Для подтверждения корректности в программе используется сравнение соседних элементов. Требуется, чтобы элементы в массиве находились в возрастающем порядке.



Сравнение производится вместе с выводом элементов. Переменная «ind» отвечает за корректность выводимых значений. Если нашлось хотя бы одно несоответствие требованиям, то переменная принимает значение «1». Если все правильно, то переменная так и останется нулевой. Пользователь может увидеть результат проверки на последней выводимой строке.

# Результаты экспериментов

Были проведены эксперименты, которые показали эффективность алгоритмов. Первое число – количество сравнений, второе – перестановок.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вид сортировки** | **Экс.1** | **Экс.2** | **Экс.3** | **Экс.4** | **Экс.5** | **Экс.6** | **Экс.7** | **Экс.8** | **Экс.9** | **Экс.10** |
| **Выбором** | 45 0 | 45 5 | 45 6 | 105 12 | 120 14 | 105 11 | 105 12 | 105 14 | 105 12 | 105 11 |
| **Расческой** | 84 0 | 84 5 | 84 8 | 176 14 | 206 24 | 169 15 | 176 16 | 176 18 | 176 17 | 176 19 |
| **Слиянием** | 53 0 | 34 34 | 46 26 | 84 36 | 89 43 | 89 33 | 74 43 | 80 43 | 74 43 | 82 43 |
| **Поразрядная** | 0 50 | 0 50 | 0 50 | 0 75 | 0 80 | 0 75 | 0 75 | 0 75 | 0 75 | 0 75 |

1. **Сортировка выбором**

Лучший случай: O(n^2)

Худший случай: O(n^2)

Cуть алгоритма заключается в проходе по массиву от начала до конца в поиске минимального элемента массива и перемещении его в начало, как и в сортировке пузырьком. Это трудоемкая операция и поэтому сложность довольно высокая.

1. **Сортировка расческой**

Лучший случай: O(n\*(log(n)))

Худший случай: O(n^2)

Здесь уже сравниваются не два соседних элемента, а элементы на промежутке, к примеру, в пять элементов. Это обеспечивает от избавления мелких значений в конце, что способствует ускорению сортировки. Практически в любом случае сложность алгоритма равняется O(n×log2n), но в худшем случае все же O(n^2).

1. **Сортировка слиянием**

Лучший случай: O(n\*(log(n)))

Худший случай: O(n\*(log(n)))

Эта сортировка не проходит по всему массиву, поэтому имеет меньшую сложность. Исходный массив будет разбиваться на маленькие, вплоть до единичных. Уже дальше будут сравниваться элементы и переставляются, причем сравниваются уже не все элементы а лишь не обходимые. В любом случае сложность будет O(n\*(log(n))).

Минусом сортировки является выделение дополнительной памяти.

1. **Поразрядная сортировка**

Лучший случай: O(w\*n), w – количество бит

Худший случай: O(w\*n), w – количество бит

Сортировка тоже не идет по массиву, а заносит значения в новый массив и передает исходному. Поэтому операций не так много, как в сортировке расческой и сортировке выбором. Однако идет обращение непосредственно к битовой системе, что замедляет процесс. Эта сортировка имеет сложность O(w\*n) и эффективна при больших массивах.

# Заключение

Написанная программа позволяет выбрать размер массива, способ его заполнения (ввод/ автозаполнение случайными числами) и сортировку, с помощью которой элементы массива будут выстроены в возрастающем порядке. Описаны все алгоритмы и их реализация на языке C и создан понятный интерфейс для пользователя, который решит воспользоваться программой.

Реализованы все четыре сортировки и поставлены эксперименты, по результатам которых построена таблица, в которой видно, в каком из алгоритмов больше сравнений и перестановок. Наиболее эффективными являются сортировка слиянием и поразрядная сортировка. Они не проходят по всему массиву, а сравнивают отдельные элементы с уже заданными адресами.

# Приложение

