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»**

**Выполнил**:

студент группы 3822Б1ПМ1

Коротин Е. В.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**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

Описание задачи, что от вас требовалось сделать

# Метод решения

1. **Сортировка вставками.**

Идея сортировки состоит в разделении массива на две части: отсортированную и неотсортированную, и поочерёдной вставке элементов из неотсортированной части на своё место в отсортированной. Изначально в отсортированной части находится лишь первый элемент, так как массив из одного элемента считается отсортированным, а затем последовательно считываются другие элементы и размещаются на подходящее место в отсортированной части.

1. **Сортировка слиянием.**

Идея сортировки состоит в разделении массива на несколько отсортированных частей и объединении их, одновременно с размещением элементов нужном нам порядке. Массив из одного элемента изначально отсортирован, поэтому мы делим единый массив на множество частей, состоящих из одного элемента, затем попарно объединяя элементы и расставляя их в правильном порядке, образуются несколько отсортированных массивов из двух элементов. Проделав те же самые действия, образуются отсортированные массивы из 4х элементов, и т.д., пока мы не получим отсортированный массив изначального размера.

1. **Поразрядная сортировка.**

Идея сортировки состоит в рассмотрении не числа целиком, а отдельных байтов чисел, и последовательном выставлении чисел в массиве, в порядке, соответствующем этим байтам. Утверждается, что проделав такую операцию со всеми байтами чисел, мы получим массив в отсортированном виде. В данной сортировке, вместо непосредственного сравнения байт, то есть чисел от 0 до 255 между собой, мы подсчитываем количество разных чисел и выставляем их в нужном порядке. Сначала рассматривается только первый байт чисел, все байты расставляются нужном порядке, затем все числа расставляются на места, соответствующие своим байтам, сохраняя изначальный порядок, если у нескольких чисел первый байт одинаковый. Такие же действия проводятся со вторым и так далее байтами, на выходе получается отсортированный массив.

# Руководство пользователя

При запуске программы требуется ввести размер (количество элементов) массива, который нужно будет отсортировать, это должно быть целое положительное число

После ввода размера массива, появится возможность выбрать вид сортировки, с помощью которой нужно будет отсортировать массив случайных чисел данного размера:

При вводе «1» к массиву будет применена сортировка вставками, при вводе «2» – сортировка слиянием, «3» –поразрядная сортировка. При вводе числа, не совпадающего, со значениями выше, будет выведено сообщение о некорректности введённой информации и будет предложено заново выбрать сортировку.

В случае корректного ввода программа проведёт выбранную сортировку и выведет: время работы сортировки и подтверждение корректности проведённой сортировки через библиотечную функцию qsort.

# Описание программной реализации

Программа состоит из одного файла LabSorts.cpp, в котором реализованы все функции: main, функции сортировок и вспомогательные функции:

**void swap(float\* a, float\* b)** – вспомогательная функция, для смены элементов между собой без использования дополнительных переменных, принимает на вход указатели на два элемента, которые мы хотим поменять местами.

**void Insert(float\* arr, int n) –** функция сортировки вставками, принимает на вход сортируемый массив и его размер. Функция состоит из цикла от 1 до n-1, т.е. проходит по всем элементам массива, кроме нулевого. В него вложен другой цикл, который поочерёдно сравнивает i-ый элемент массива с предыдущими и меняется с ними местами до тех пор, пока не будет обнаружен элемент, меньший i-го или до того момента, когда i-ый элемент станет первым в массиве.

**void merge(float\* inp, float\* out, int l, int m, int r) –** вспомогательная функция для сортировки слияния, объединяющая две разные части массива. Принимает на вход сортируемый массив, дополнительный массив того же размера, номер первого элемента первой части массива, номер первого элемента второй части массива и номер последнего элемента второй части массива + 1. Функция состоит из цикла while, который сравнивает первые элементы двух частей, которые мы хотим объединить, и записывает в дополнительный массив наименьший из них, после чего указатель на наименьший из элементов смещается на 1, цикл продолжается до тех пор, пока один из указателей не дойдёт до конца соответствующей ему части. Циклы for переносят в дополнительный массив все неперенесённые элементы из той части, в которой указатель не дошёл до финального элемента. На выходе во вспомогательном массиве на промежутке от l до r образуется отсортированный массив.

**void Mergesort(float\* inp, float\* out, int l, int r)** –функция сортировки слиянием, рекурсивно-заданная функция, принимает на вход сортируемый массив, другой массив такого же размера, а так же левую и правую границы массива (при первом вызове они равны 0 и n-1 соответственно, где n – введённый пользователем размер массива). Условие остановки рекурсии - равенство правой и левой границ массива, то есть тогда, когда массив состоит из одного элемента. Если массив не состоит из одного элемента, функция разделяет массив на 2 части, и запускает **Mergesort** по этим частям, после того как обе функции будут выполнены, запускается функция **Merge,** после чего переносит значения с l до r из вспомогательного массива в основной.

**void countByte(uint\* mas, int size, int count[256], int byte)** – вспомогательная функция для функции **LSD**, считающая количество различных байтов. Принимает на вход сортируемый массив, его размер, массив для подсчёта байтов размером 256, и номер байта чисел в сортируемом массиве. Функция состоит из нескольких циклов for, первый из которых от 0 до size-1, считает, сколько байтов хранят значение 0, 1, …, 255 и записывает это число в массив count с соответствующим индексом. Второй цикл for от i=1 до 255 превращает массив count в массив префиксных сумм, где count[i] равняется префиксной сумме от count[i-1], count[0]=0, таким образом, мы узнаём, на каком месте должны быть расположены определённые байты.

**void LSD(uint\* mas, int size, uint\* mas\_tmp) –** вспомогательная функция для функции **radix**, которая сортирует массив, как массив чисел типа unsigned int(uint). Функция принимает на вход сортируемый массив, его размер и вспомогательный массив того же размера. Функция состоит из цикла от i=0 до syzetype-1 (syzetype – размер типа данных в байтах, в случае с float равен 4), который запускает функцию **countByte**. В этот цикл включён ещё один, который располагает элементы массива во вспомогательном массиве, в соответствии с местами, на которых должны быть расположены принадлежащие им байты, при этом сохраняется изначальный порядок, если у двух разных числе одинаковые байты. После чего вспомогательный массив копируется в основной массив.

**void radix(float\* mas, int n)** – функция поразрядной сортировки, принимает на вход сортируемый массив и его размер. Функция запускает функцию **LSD**, при работе с типом данных float, после её завершения мы получим массив, в котором сначала расположены по возрастанию все положительные числа, а затем по убыванию расположены все отрицательные. Через цикл while мы находим индекс первого положительного элемента, после чего, с помощью цикла for переносим все положительные числа из начала сортируемого массива в конец вспомогательного, а затем, с помощью ещё одного цикла for, переносим все отрицательные числа в перевёрнутом виде в начало вспомогательного массива. После чего копируем вспомогательный массив в основной. На выходе получаем отсортированный массив.

# Подтверждение корректности

Для подтверждения корректности в программе используется функция **void check(float\* mas, float\* copy, int n),** которая принимает на вход отсортированный массив, его неотсортированную копию и их размер. Функция сортирует копию массива библиотечной сортировкой qsort, а затем, через цикл for, поэлементно сравниваются оба массива, для этого запускается счётчик, который изначально равен 0, после первой пары несовпавших элементов, цикл прервётся и появится сообщение о неуспешном проведении сортировки.

# Результаты экспериментов

По данным экспериментов видно, что …

# Заключение

# Приложение

Часть кода, который вы считаете основным