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**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 382003-1

Котихин Александр Андреевич

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2020

**Содержание**

Постановка задачи……………………………………………………………………………………………3

Метод решения……………………………………………………………………………….………………4

Руководство пользователя…………………………………………………………………..……………….6

Описание программной реализации…………………………………………………………..…………….8

Подтверждение корректности……………………………………………………………………………….9

Результаты экспериментов……………………………………………………………..……….…………..10

Заключение…………………………………………………………………………………..………………12

Приложение…………………………………………………………………………………….……………13

# Постановка задачи

Описание задачи, что от вас требовалось сделать

Необходимо:

1) Реализовать следующие сортировки для типа данных float.:

1. Сортировка вставками
2. Сортировка Хоара (Быстрая сортировка)
3. Сортировка слиянием
4. Поразрядная сортировка

2) Замерить количество сравнений и перестановок при реализации сортировки.

3) Тесты сортировок поставить так, чтобы показать теоретическую сложность представленных сортировок.

# Метод решения

1. Сортировка вставками

На вход подаётся массив чисел. Сортировка состоит из вложенного цикла. Внешний и внутренний циклы начинаются со второго элемента массива. Во внутреннем цикле происходит сравнение второго элемента с первым. Если второй оказался меньше (при упорядочивании по возрастанию), то элементы меняются местами. Далее внешний и внутренний циклы принимают значение третьего элемента. Во внутреннем цикле происходит сравнение этого элемента с предыдущими. Если он оказывается меньше предыдущего, элементы меняются местами. Алгоритм повторяется до последнего элемента в массиве.

1. Сортировка Хоара (быстрая сортировка)

На вход подается массив чисел. Сначала определяется опорный элемент (в данной реализации середина массива). Далее относительно этого элемента производятся сравнения и сортировка левой и правой частей. По итогу первого прохода цикла, массив разбивается на две части: в левой меньшие элементы, в правой большие. Далее рекурсивно вызывается эта же функция для левой и правой частей по отдельности, левая и правая части сортируются по такому же принципу. После сортировки массивы разбиваются ещё на две части, и снова так же сортируются. Цикл повторяется до полной сортировки всего массива.

1. Сортировка слиянием

На вход подается массив чисел. В функции MergeSort сначала проверяется условие существования нескольких элементов в массиве. Дальше проверяется, является ли заданный массив двухэлементным, если является, то элементы сортируются (в данном случае по возрастанию). В функции MergeSort рекурсивно вызывается эта же функция для правой половины массива и для левой половины массива. В итоге, массив делится на пары и сортируется до тех пор, пока не кончатся элементы в массиве. В функции Merge происходит слияние упорядоченных пар. Упорядоченный массив отдельно записывается во временный массив SortArr. В функции сравниваются элементы из левой и правой частей массивов. В зависимости от того, какой элемент меньший, он записывается во временный массив. Сравнения и записи повторяются до тех пор, пока либо левая граница массива не дойдет до середины, либо пока правая граница не дойдет до конца. Последним шагом упорядоченные данные из временного массива перезаписываются в основной массив.

1. Поразрядная сортировка

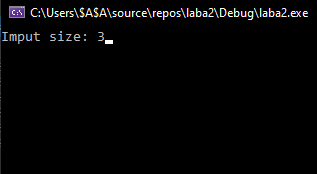
Сначала создаётся вспомогательный массив, подсчитывающий количество разрядов (функция CreateCounters). Далее в сортировке (функция RadixPass) одного разряда алгоритм аналогичен алгоритму сортировки подсчетом, при этом количество разрядов было подсчитано ранее. На вход самой сортировки (функция RadixSort) приходят входной, выходной массивы и массив counters. Подсчитываются все разряды, выделяется количество разрядов, которые подсчитаны для конкретного значения, и для определенного разряда вызывается сортировка подсчетом, реализованная в RadixPass. Далее алгоритм повторяется для остальных подчитанных разрядов.

# Руководство пользователя

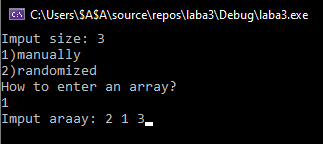
Описание работы с вашей программы, что должен сделать пользователь.

Программа должна получить на вход следующие данные:

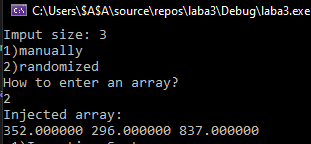
1. Размер массива



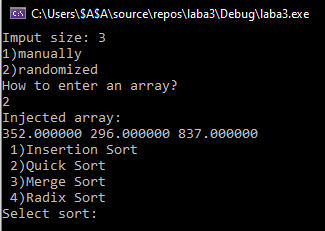
1. Выбор типа заполнения массива (вручную/случайными числами)
2. При заполнении вручную, программа потребует ввести массив



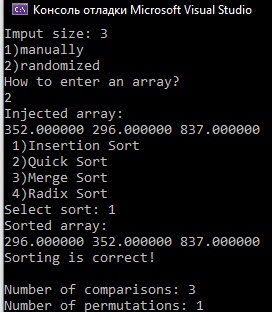
1. При заполнении случайными числами массив заполнится автоматически



1. Далее программа потребует выбрать тип сортировки из списка



1. Программа выводит отсортированный массив, подтверждение корректности сортировки. В зависимости от выбранной сортировки, выводит количество перестановок и сравнений.



# Описание программной реализации

Структура проекта, какие есть файлы и что в них содержится.

1. Сортировка подсчетом

Функция void InsertionSort принимает значения массива и его размера.

1. Сортировка Хоара

Функция void QuickSort принимает значения массива, номера начального и конечного элементов.

1. Сортировка слиянием

Функция void Merge

Функция void MergeSort принимает значение массива, номера начального и конечного элементов.

1. Поразрядная сортировка

Функция void CreateCounters

Функция void RadixPass

Функция void RadixSort принимает значения входного, выходного и вспомогательного массивов и размер входного массива

1. Проверка корректности

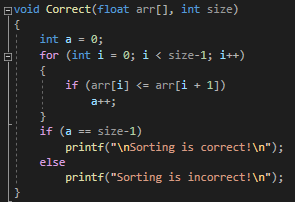
Функция void Correct принимает значение массива и его размер, выводит подтверждение корректности сортировки.

1. Реализация программы

Функция int main() (см. приложение) запрашивает на ввод размер массива, сам массив (либо случайно заполненный), номер сортировки из списка. Выводит отсортированный выбранной сортировкой массив, подтверждение о корректности (из функции void Correct), количество сравнений и перестановок.

# Подтверждение корректности

Для подтверждения корректности в программе в программе реализована функция void Correct. В функции проверяется, больше или равен ли каждый следующий элемент массива, по сравнению с предыдущим.



# Результаты экспериментов

Данные экспериментов, сгенерированные случайно:



**Сортировка вставками**

Количество сравнений в сортировке вставками равно n\*(n-1)/2. Количество перестановок зависит от упорядоченности массива. В случае, когда массив уже упорядочен, количество перестановок – 0 (лучший случай). Сложность В случает обратно отсортированного массива (худший случай), количество перестановок максимально и равно количеству сравнений (n\*(n-1)/2 в общем случае). Сложность . Средняя сложность алгоритма .



**Быстрая сортировка**

Лучший случай сортировки Хоара – отсортированный массив. При этом, количество перестановок равно нулю. Сложность. Худший случай сортировки – когда опорный элемент максимален. Сложность . Средняя сложность .



**Сортировка слиянием**

Количество перестановок (копий) всегда неизменно (при одинаковых размерах). Сложность в лучшем случае: . Худший случай достигается, когда каждый элемент массива сравнивается хотя бы один раз (например: 0,2,4,6,1,3,5,7) Сложность . Средняя сложность .



**Поразрядная сортировка**

Так как поразрядная сортировка линейная её сложность всегда где *k* – количество разрядов.

# Заключение

Из вышеописанных экспериментов, видна зависимость количества сравнений/перестановок и сложностью алгоритма. Чем меньше в алгоритме перестановок и сравнений, тем быстрее работает сортировка. Видно, что при определенных условиях задания массива некоторые сортировки работают медленнее или быстрее. Линейные сортировки работают за одинаковое время.

# Приложение

