Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

**Отчёт по учебной практике**

«**Методы сортировки**»

Выполнил: студент 381806-3 группы

Зайцев А.Р.

Проверила: к.т.н., доцент каф. МОСТ ИИТММ Кустикова В.Д.

Нижний Новгород

2018

**Содержание**

[Введение 3](#_Toc533623228)

[1. Постановка задачи 4](#_Toc533623229)

[2. Руководство пользователя 5](#_Toc533623230)

[3. Руководство программиста 8](#_Toc533623231)

[3.1. Структура программы 8](#_Toc533623232)

[3.2. Описание алгоритма 8](#_Toc533623233)

[3.2.1. Сортировка пузырьком 8](#_Toc533623234)

[3.2.2. Сортировка вставками 9](#_Toc533623235)

[3.2.3. Сортировка выбором 10](#_Toc533623236)

[3.2.4. Сортировка подсчётом 11](#_Toc533623237)

[3.2.5. Быстрая сортировка (Хоара) 12](#_Toc533623238)

[3.2.6. Сортировка слиянием 13](#_Toc533623239)

[3.3. Описание функций 14](#_Toc533623240)

[Заключение 16](#_Toc533623241)

[Литература 17](#_Toc533623242)

# Введение

Нетрудно догадаться, что людям удобно работать с упорядоченными по каким-либо признакам списками данных. Именно для упорядочивания множеств и списков данных нужны сортировки.

Работа с файлами –неотъемлемый способ взаимодействия человека с операционной системой. Для того чтобы со списком файлов было удобнее работать, его необходимо отсортировать, например, по размеру.

Именно для этого и понадобилась программа «файловый менеджер», с помощью которой можно вывести на экран отсортированный одним из 6 методов сортировки набор файлов с указанием размеров.

# Постановка задачи

**Задача:** разработать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченным по возрастанию/убыванию размера.

**Входные данные:** путь до директории, в которой необходимо отсортировать содержимое, и метод сортировки.

**Выходные данные:** отсортированный список имён файлов с указанием размера и время сортировки.

# Руководство пользователя

На данные момент у меня получилось реализовать сортировку массива несколькими способами сортировки.

1. Запустите приложение Practice5.exe. При запуске программы перед вами на экране появится её название и сообщение с указанием ввести путь к папке (Рис1).

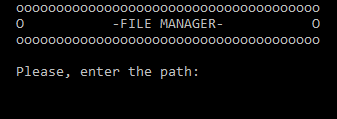


Рис 1. Программа после запуска

1. При неверном указании пути, программа выдаст сообщение, говорящее о том, что путь не найден. После чего вы сможете повторно ввести путь (Рис 2).

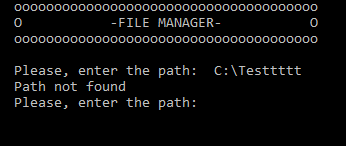


Рис 2. Неверный путь

1. При верном указании пути может появиться 2 сообщения. Первое, если выбранная папка окажется пустой (Рис 3). Второе, если в ней будут файлы (программа укажет их кол-во); одновременно с этим появится список сортировок на выбор (Рис 4).

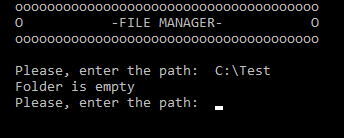


Рис 3. Пустая папка

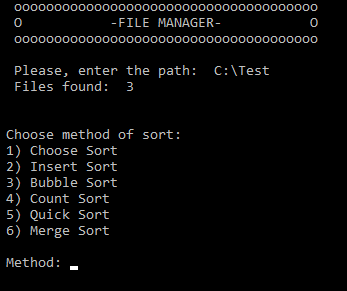


Рис 4. Список сортировок для непустой папки

1. После ввода желаемой сортировки перед вами в порядке возрастания размера появится список файлов из папки, а чуть ниже сообщение с затраченным на сортировку временем (Рис 5).

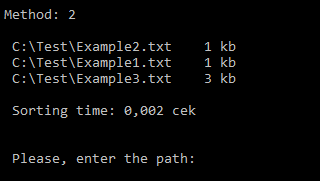


Рис 5. Отсортированный список

1. Также при достаточно большом размере файла сортировка вставками не сработает, программа предложит сменить сортировку (Рис 6).

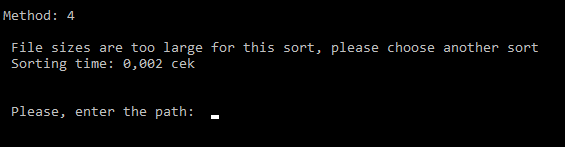


Рис 6. Просьба смены сортировки при большом размере файла

1. На (Рис 7) вы можете увидеть различие файлов по размеру и корректность работы программы.

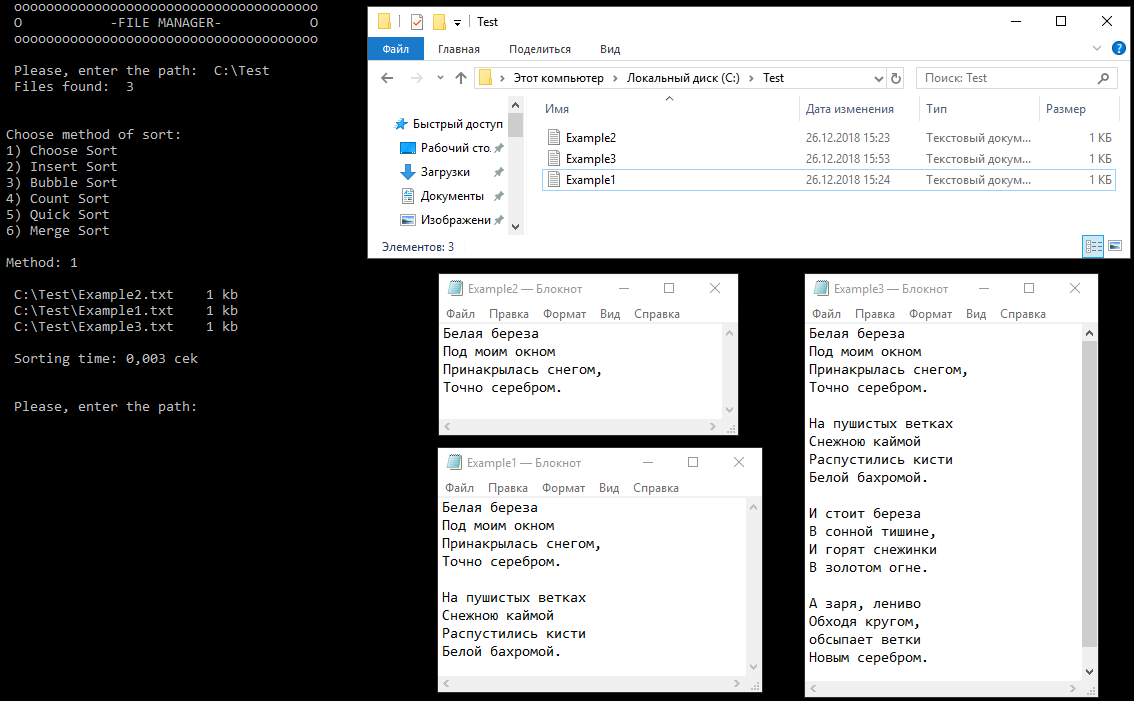


Рис 7. Показ файлов и работа программы целиком

# Руководство программиста

## Структура программы

Весь код программы содержится в файле «**File manager.c**».

Код программы состоит из отдельных функций, для упрощения написания программы и лучшей наглядности.

## Описание алгоритма

### Сортировка пузырьком

Расположим массив сверху вниз, от нулевого элемента к последнему. Идем по последовательности снизу - вверх, просматривая при том соседние пары элементов. Если элементы пары находятся в неправильном порядке, меняем их местами. После нулевого прохода вверху оказывается самый маленький элемент. Следующий проход делается до второго сверху элемента. Проходы совершаются до тех пор, пока в нем не останется только 1 элемент. Пример сортировки пузырьком приведен в (Таблица 1).

Таблица 1. Пример сортировки пузырьком

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **Сравниваются** | **Замена места** |
| Исходная последовательность | 5 | 9 | 4 | 2 | 1 | 5 |  |  |
| 0 | 5 | 9 | 4 | 2 | **1** | **5** | 5 и 1 | Нет |
| 5 | 9 | 4 | **2** | **1** | 5 | 1 и 2 | Да |
| 5 | 9 | **4** | **1** | 2 | 5 | 1 и 4 | Да |
| 5 | **9** | **1** | 4 | 2 | 5 | 1 и 9 | Да |
| **5** | **1** | 9 | 4 | 2 | 5 | 1 и 5 | Да |
| 1 | 1 | 5 | 9 | 4 | **2** | **5** | 5 и 2 | Нет |
| 1 | 5 | 9 | **4** | **2** | 5 | 2 и 4 | Да |
| 1 | 5 | **9** | **2** | 4 | 5 | 2 и 9 | Да |
| 1 | **5** | **2** | 9 | 4 | 5 | 2 и 5 | Да |
| 2 | 1 | 2 | 5 | 9 | **4** | **5** | 5 и 4 | Нет |
| 1 | 2 | 5 | **9** | **4** | 5 | 4 и 9 | Да |
| 1 | 2 | **5** | **4** | 9 | 5 | 4 и 5 | Да |
| 3 | 1 | 2 | 4 | 5 | **9** | **5** | 5 и 9 | Да |
| 1 | 2 | 4 | **5** | **5** | 9 | 5 и 5 | Нет |
| 4 | 1 | 2 | 4 | 5 | 5 | 9 | - | - |

### Сортировка вставками

Будем считать, что первый элемент последовательности отсортирован, а оставшаяся часть последовательности нет. Сравниваем a[i] элемент с a[i + 1], и если a[i + 1] меньше отсортированного, то меняем их местами. Если же a[i + 1] больше a[i], то считаем, что он теперь тоже отсортирован. Проделываем те же действия с оставшимися элементами. Пример сортировки вставками приведен в (Таблица 2).

Таблица 2. Пример сортировки вставками

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **Сравниваются** | **Замена места** |
| Исходная последовательность | 5 | 9 | 4 | 2 | 1 | 5 |  |  |
| 0 | **5** | **9** | 4 | 2 | 1 | 5 | 5 и 9 | Нет |
| 1 | 5 | **9** | **4** | 2 | 1 | 5 | 9 и 4 | Да |
| **5** | **4** | 9 | 2 | 1 | 5 | 4 и 5 | Да |
| 2 | 4 | 5 | **9** | **2** | 1 | 5 | 9 и 2 | Да |
| 4 | **5** | **2** | 9 | 1 | 5 | 2 и 5 | Да |
| **4** | **2** | 5 | 9 | 1 | 5 | 2 и 4 | Да |
| 3 | 2 | 4 | 5 | **9** | **1** | 5 | 9 и 1 | Да |
| 2 | 4 | **5** | **1** | 9 | 5 | 1 и 5 | Да |
| 2 | **4** | **1** | 5 | 9 | 5 | 1 и 4 | Да |
| **2** | **1** | 4 | 5 | 9 | 5 | 1 и 2 | Да |
| 4 | 1 | 2 | 4 | 5 | **9** | **5** | 9 и 5 | Да |
| 1 | 2 | 4 | **5** | **5** | 9 | 5 и 5 | Нет |
| 1 | 2 | **4** | **5** | 5 | 9 | 5 и 4 | Нет |
| 1 | **2** | **4** | 5 | 5 | 9 | 4 и 2 | Нет |
| **1** | **2** | 4 | 5 | 5 | 9 | 2 и 1 | Нет |
| 1 | 2 | 4 | 5 | 5 | 9 | - | - |

### Сортировка выбором

Идея метода состоит в том, чтобы создавать отсортированную последовательность путем присоединения элементов в правильном порядке.

На i-том шаге выбирается наименьший элемент из a[i] до a[n]. Он меняется местами с a[i]. На n – 1 шаге вся последовательность, кроме a[n] уже отсортирована. А a[n] уже стоит на своем последнем месте, так как все меньшие элементы ушли влево.

Пример сортировки выбором приведен в (Таблица 3).

Таблица 3. Пример сортировки выбором

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6** | | | | | | **min** | **min ‹—› a[i]** |
| Исходная последовательность | 5 | 9 | 4 | 2 | 1 | 5 |  |  |
| 0 | **1** | 9 | 4 | 2 | **5** | 5 | 1 | 5 |
| 1 | 1 | **2** | 4 | **9** | 5 | 5 | 2 | 9 |
| 2 | 1 | 2 | **4** | 9 | 5 | 5 | 4 | - |
| 3 | 1 | 2 | 4 | **5** | **9** | 5 | 5 | 9 |
| 4 | 1 | 2 | 4 | 5 | **5** | **9** | 5 | 9 |
| 5 | 1 | 2 | 4 | 5 | 5 | 9 | - | - |

### Сортировка подсчётом

Находим min и max в сортируемой последовательности. Создаем дополнительный массив (пусть он называется С) размером (max – min) + 1 и заполняем его нулями. Последовательно пройдем по заданному массиву и запишем в С[i] количество чисел равных i. Теперь достаточно пройти по массиву С и для каждого i в заданный массив записать последовательно число С[i] раз. Пример сортировки подсчетом приведен в (Таблица 4).

Таблица 4. Пример сортировки подсчётом

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность, n = 6,**  **min = 0, max = 4** | | | | | | **Дополнительный массив** | | | | |
| Исходная последовательность | 3 | 2 | 1 | 4 | 0 | 1 | 0 | 1 | 2 | 3 | 4 |
| 1 | + |  |  |  |  |  | 0 | 0 | 0 | 1 | 0 |
|  | + |  |  |  |  | 0 | 0 | 1 | 1 | 0 |
|  |  | + |  |  |  | 0 | 1 | 1 | 1 | 0 |
|  |  |  | + |  |  | 0 | 1 | 1 | 1 | 1 |
|  |  |  |  | + |  | 1 | 1 | 1 | 1 | 1 |
|  |  |  |  |  | + | 1 | 2 | 1 | 1 | 1 |
| 2 | **0** | 2 | 1 | 4 | 0 | 1 | + |  |  |  |  |
| **0** | **1** | 1 | 4 | 0 | 1 |  | + |  |  |  |
| **0** | **1** | **1** | 4 | 0 | 1 |  |  | + |  |  |
| **0** | **1** | **1** | **2** | 0 | 1 |  |  |  | + |  |
| **0** | **1** | **1** | **2** | **3** | 1 |  |  |  |  | + |
|  | **0** | **1** | **1** | **2** | **3** | **4** |  |  |  |  |  |

### Быстрая сортировка (Хоара)

Общий алгоритм быстрой сортировки выглядит следующим образом:

Выбираем ведущий элемент (можно взять средний элемент или выбрать случайным образом).

Переставляем значения в массиве таким образом, чтобы элементы, меньшие ведущего, оказались слева, а большие – справа.

Сортируем правую и левую часть таким же методом.

Массив считается отсортированным, когда в каждой части останется по одному элементу.

Пример быстрой сортировки приведён в (Таблица 5).

Таблица 5. Пример быстрой сортировки

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность n = 7** | | | | | | |
| Исходная последовательность | 3 | 9 | 7 | 6 | 8 | 4 | 2 |
| 0 | 3 | 9 | 7 | **6** | 8 | 4 | 2 |
| 3 | 2 | 7 | **6** | 8 | 4 | 9 |
| 3 | 2 | 4 | **6** | 8 | 7 | 9 |
| 1 | 3 | **2** | 4 | 6 | 8 | **7** | 9 |
| **2** | 3 | 4 | 6 | **7** | 8 | 9 |
|  | 2 | 3 | 4 | 6 | 7 | 8 | 9 |

### Сортировка слиянием

Последовательность делится пополам. Далее продолжаем делить каждую часть пополам, пока в каждой части не останется по одному элементу. Одиночный элемент является отсортированным. Далее последовательно соединяем отсортированные части.

В (Таблица 6) приведён пример сортировки слиянием.

Таблица 6. Пример сортировки слиянием

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ итерации** | **Последовательность n = 8** | | | | | | | |
| Исходная последовательность | 7 | 6 | 2 | 3 | 1 | 5 | 6 | 0 |
| 0 | 7 | 6 | 2 | 3 | 1 | 5 | 6 | 0 |
| 7 | 6 | 2 | 3 | 1 | 5 | 6 | 0 |
| 7 | 6 | 2 | 3 | 1 | 5 | 6 | 0 |
| 7 | 6 | 2 | 3 | 1 | 5 | 6 | 0 |
| 1 | 6 | 7 | 2 | 3 | 1 | 5 | 0 | 6 |
| 2 | 3 | 6 | 7 | 1 | 5 | 0 | 6 |
| 2 | 2 | 3 | 6 | 7 | 0 | 1 | 5 | 6 |
| - | 0 | 1 | 2 | 3 | 5 | 6 | 6 | 7 |

## Описание функций

void Input(wchar\_t \*\*path)

**Назначение:** Запрос у пользователя пути к нужной папке для сортировки

**Входные данные:** \*\*path – указатель на массив, в который будет записана строка с директорией

int ListDirectoryContetns(const wchar\_t \*sDir, wchar\_t \*\*\*fNames, ULONGLONG \*\*fSizes)

**Назначение:** Получение имен и размеров файлов, подсчет количества файлов

**Входные данные:** \*sDir – строка с полученной директорией, \*\*\*fNames – массив для записи имён файлов, \*\*fSizes – массив для записи размеров файлов

**Выходные данные:** Количество файлов в папке, число -1 при несуществующей директории

void Output(int \*ind, int count\_files, wchar\_t \*\*fNames, ULONGLONG \*size)

**Назначение:** Вывод отсортированного списка файлов с размерами

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*\*fNames – массив с именами файлов, \*size – массив с размерами файлов

void Choose\_Sort(int count\_files, ULONGLONG \*size, int \*ind)

**Назначение:** Выполнение сортировки выбором

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void Insert\_Sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки вставками

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void Bubble\_Sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки «пузырьком»

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

int Count\_Sort(ULONGLONG \*size, int count\_files, int \*ind)

**Назначение:** Выполнение сортировки подсчетом

**Входные данные:** \*ind - массив для индексов отсортированных файлов, count\_files – количество файлов в папке, \*size – массив с размерами файлов

void Quick\_Split(int \*ind, ULONGLONG \*size, int \*i, int \*j, ULONGLONG p)

**Назначение:** Подготовка к быстрой сортировке

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, i и j – индексы крайних элементов сортируемого фрагмента, p – опорный элемент, по которому будет производиться разделение массива

void Quick\_Sort(ULONGLONG \*size, int \*ind, int n1, int n2)

**Назначение:** Выполнение быстрой сортировки

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, n1 и n2 – индексы крайних элементов сортируемого фрагмента

void Merge(ULONGLONG \*size, int lb, int mid, int ub, int \*ind)

**Назначение:** Подготовка к сортировке слиянием

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, lb и ub – индексы крайних элементов сортируемого фрагмента, mid – индекс среднего элемента сортируемого фрагмента

void Merge\_Sort(ULONGLONG \*size, int lb, int ub, int \*ind)

**Назначение:** Выполнение сортировки слиянием

**Входные данные:** \*ind - массив для индексов отсортированных файлов, \*size – массив с размерами файлов, lb и ub – индексы крайних элементов сортируемого фрагмента

void main()

**Назначение:** Основная функция

**Входные данные:** -

# Заключение

В ходе выполнения практической работы «Методы сортировки» был разработан прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по размеру. Для сортировки было использовано 6 различных методов:

1. Сортировка «пузырьком»
2. Сортировка вставками
3. Сортировка выбором
4. Сортировка подсчётом
5. Быстрая сортировка (Хоара)
6. Сортировка слиянием

В (Таблица 7) ниже приведена оценка сложности каждого метода сортировки.

Таблица 7. Оценка сложности методов сортировки

|  |  |  |
| --- | --- | --- |
| Методы сортировки | Средняя сложность алгоритма | |
| Время выполнения | Память |
| Сортировка «пузырьком» | O(n2) | O(1) |
| Сортировка вставками | O(n2) | O(1) |
| Сортировка выбором | O(n2) | O(1) |
| Сортировка подсчетом | O(n) | O(n) |
| Быстрая сортировка | O(nlog n) | O(log n) |
| Сортировка слиянием | O(nlog n) | O(n) |

На графиках ниже представлена зависимость времени работы сортировки от количества сортируемых элементов.

# Литература

**Источники:**

<https://vscode.ru/prog-lessons/sortirovka-puzyirkom-na-si.html>

<https://vscode.ru/prog-lessons/sortirovka-vstavkami-na-si.html>

<https://vscode.ru/prog-lessons/sortirovka-metodom-vyibora-na-yazyike-si.html>

<https://vscode.ru/prog-lessons/sortirovka-metodom-podscheta-na-si.html>

<https://ru.wikipedia.org/wiki/Быстрая_сортировка>

<https://forkettle.ru/vidioteka/programmirovanie-i-set/algoritmy-i-struktury-dannykh/108-sortirovka-i-poisk-dlya-chajnikov/1015-sortirovka-sliyaniem-merge-sort>