Министерство образования и науки Российской Федерации

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

**Отчёт по учебной практике**

**Методы сортировки**

**Выполнил**: Зореев Михаил Владимирович студент группы 381806-1

**Проверил**: доцент кафедры МОСТ Кустикова Валентина Дмитриевна

Нижний Новгород

2018 г.

Оглавление

[1. Введение 3](#_Toc533364093)

[2. Постановка задачи 4](#_Toc533364094)

[3. Руководство пользователя 5](#_Toc533364095)

[4. Руководство программиста 8](#_Toc533364096)

[4.1 Структура программы 8](#_Toc533364097)

[4.2 Описание алгоритмов 8](#_Toc533364098)

[4.2.1 Сортировка выбором 8](#_Toc533364099)

[4.2.2 Сортировка вставками 8](#_Toc533364100)

[4.2.3 Сортировка пузырьком 9](#_Toc533364101)

[4.2.4 Сортировка подсчётом 9](#_Toc533364102)

[4.2.5 Быстрая сортировка 9](#_Toc533364103)

[4.2.6 Сортировка слиянием 10](#_Toc533364104)

[4.3 Описание функций 11](#_Toc533364105)

[5. Заключение 14](#_Toc533364106)

# Введение

Одной из основных функций современного компьютера является возможность работы с файлами. Для того чтобы упростить этот процесс полезно отсортировать файлы по какому-нибудь признаку, например, по размеру. Именно с этой целью был создан данный прототип файлового менеджера, позволяющий вывести на экран, отсортированный одним из 6 методов, список файлов, расположенных в интересуемой пользователя директории.

# Постановка задачи

Задача: Разработать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию/убыванию размера.

Входные данные: Путь до директории, в которой необходимо отсортировать содержимое, метод сортировки

Выходные данные: Отсортированный список имён файлов с указанием размера, время сортировки.

# Руководство пользователя

Для начала работы перейдите в папку с программой и запустите файл manager.exe после чего программа предложит вам ввести директорию, с которой вы хотите работать (Рис. 1).

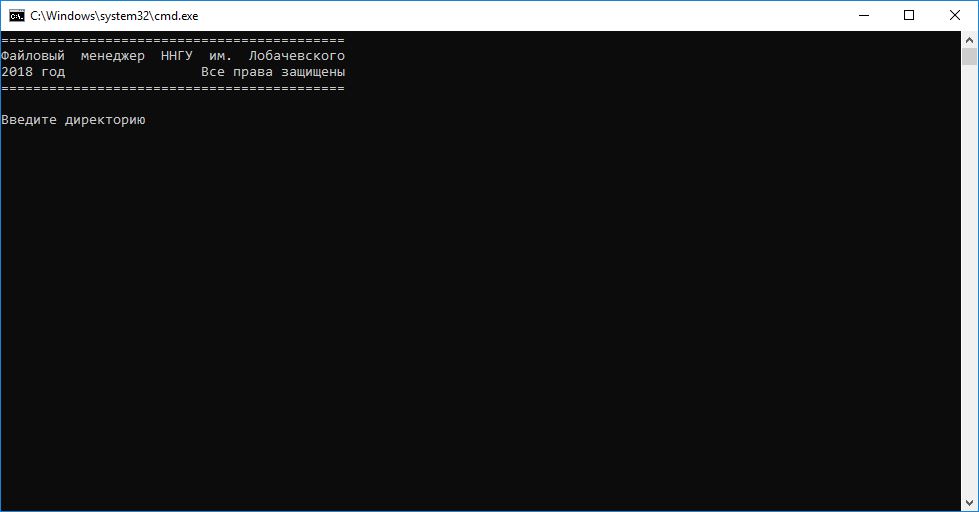


Рис. 1

Введите с помощью клавиатуры полный путь до папки и нажмите Enter. Если вы ошибётесь и введёте несуществующий путь или если выбранная папка окажется пустой программа сообщит вам об этом, выведя сообщение об ошибке (Рис. 2.). В таком случае повторите ввод.

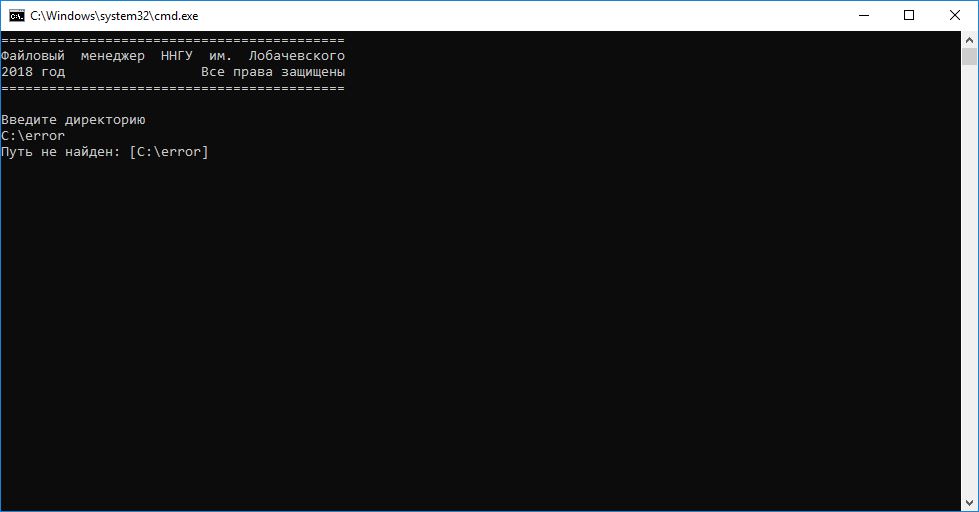


Рис. 2. Пример ошибки

В том случае, если папка по указанному адресу существует, и в ней имеется как минимум один файл, программа отобразит меню предлагающее выбрать одну из шести представленных сортировок (Рис. 3.). Для выбора сортировки введите цифру, указанную напротив неё и нажмите Enter. Следует обратить внимание, что если в папке имеются файлы больше 2 гигабайт, то сортировка подсчётом будет недоступна

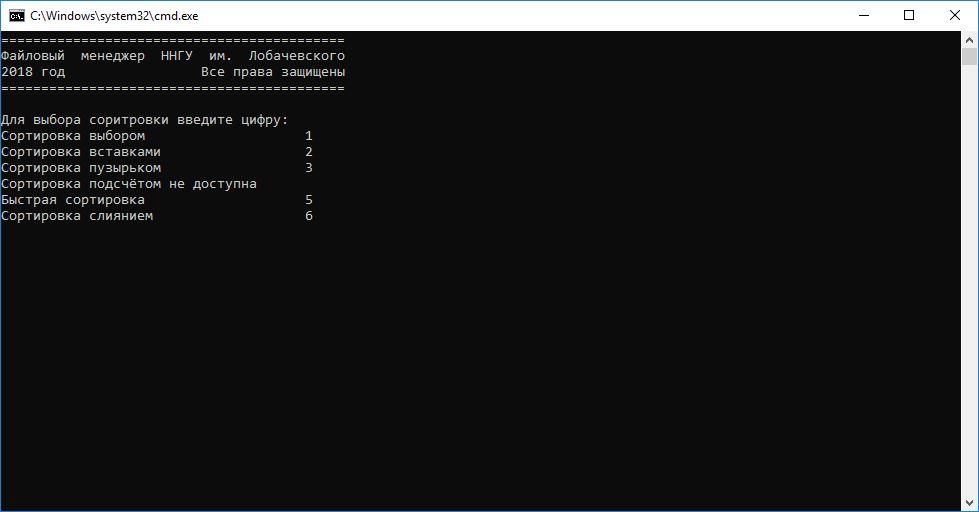


Рис. 3.

Далее вам будет предложено выбрать направление сортировки – по возрастанию или по убыванию (Рис. 4). Введите соответствующую цифру и нажмите Enter.

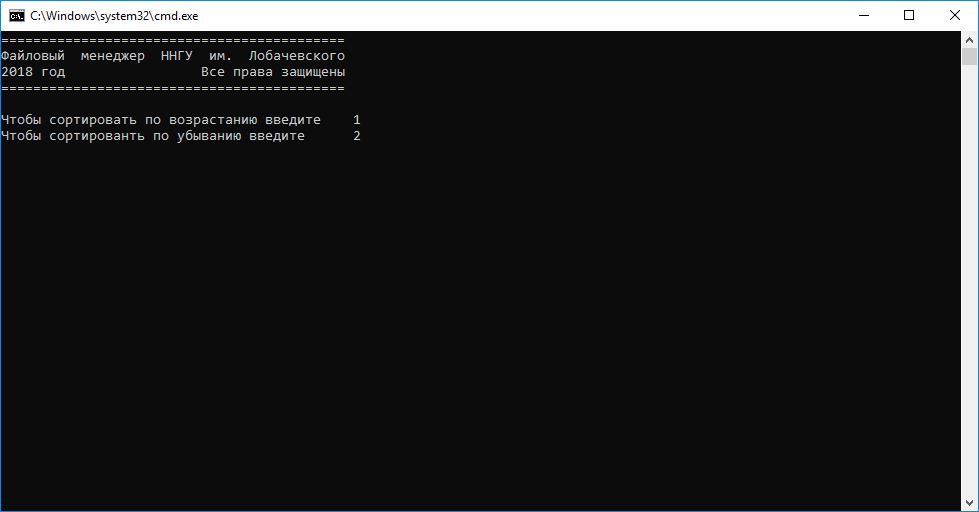


Рис. 4

После завершения вычислений программа выведет на экран отсортированный список файлов, с указанием их размера в байтах. Дальше вы сможете отсортировать файлы, используя другой алгоритм, сменить директорию или выйти из программы (Рис. 5). Для этого введите указанную напротив нужного действия цифру и нажмите Enter.

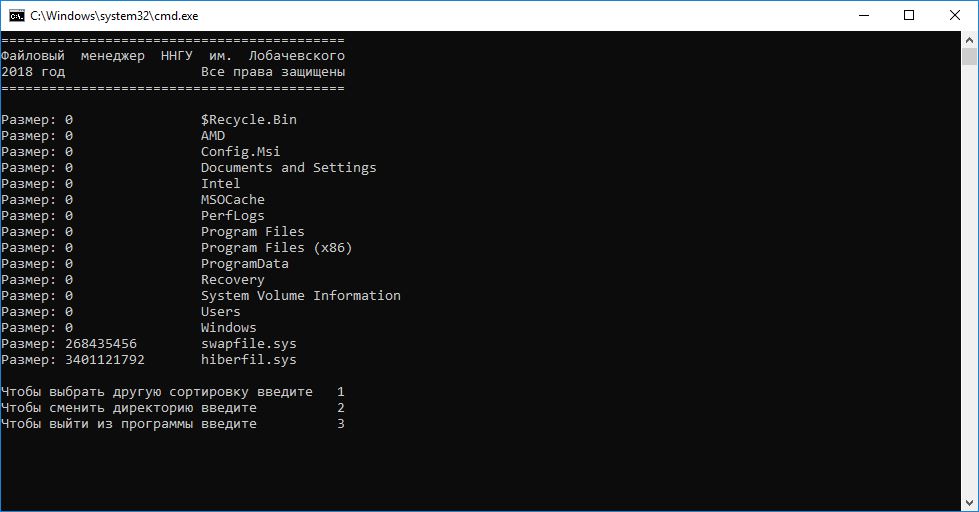


Рис. 5

# Руководство программиста

## Структура программы

Весь программный код расположен в файле main.c

## Описание алгоритмов

### Сортировка выбором

1. Массив делится на отсортированную и неотсортированную часть
2. Ищется минимальный элемент в неотсортированной части.
3. Этот элемент меняется с первым из неотсортированной части, после этого отсортированная часть считается увеличившейся на этот элемент.
4. Описанные действия повторяются на один раз меньше чем элементов в массиве.

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 итерация | 2 | 4 | 3 | 5 | 1 |
| 1 итерация | 1 | 4 | 5 | 3 | 2 |
| 2 итерация | 1 | 2 | 5 | 3 | 4 |
| 3 итерация | 1 | 2 | 3 | 5 | 4 |
| 4 итерация | 1 | 2 | 3 | 4 | 5 |

### Сортировка вставками

1. Массив делится на отсортированную и неотсортированную часть, на первой итерации отсортированная часть состоит только из первого элемента массива.
2. Берётся первый элемент из неотсортированной части, в том случае, если этот элемент меньше чем расположенный слева от него, то они меняются местами, эти действия повторяются пока выбранный элемент не займёт своё место.
3. Описанные действие повторяются для каждого элемента из неотсортированного массива, пока тот не опустеет.

Таблица 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 итерация | 2 | 4 | 3 | 5 | 1 |
| 1 итерация | 2 | 4 | 3 | 5 | 1 |
| 2 итерация | 2 | 3 | 4 | 5 | 1 |
| 3 итерация | 2 | 3 | 4 | 5 | 1 |
| 4 итерация | 1 | 2 | 3 | 4 | 5 |

### Сортировка пузырьком

1. Идём от начала массива сравнивая соседние элементы, если k элемент окажется больше k+1 (в случае с сортировкой по возрастанию), то они меняются местами.
2. В результате этого самый большой элемент переместится к концу массива.
3. Описанные действия повторяем на один раз меньше чем элементов в массиве

Таблица 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 итерация | 2 | 4 | 3 | 5 | 1 |
| 1 итерация | 2 | 3 | 4 | 1 | 5 |
| 2 итерация | 2 | 3 | 1 | 4 | 5 |
| 3 итерация | 2 | 1 | 3 | 4 | 5 |
| 4 итерация | 1 | 2 | 3 | 4 | 5 |

### Сортировка подсчётом

1. Ищется максимальное и минимальное значение в массиве
2. Используя их создаётся временный массив размера max-min+1
3. Производится проход по массиву, в ходе которого в значение в [k – min] ячейке временного массива увеличивается на 1 каждый раз, когда в изначальном массиве встречается значение k.
4. Массив создаётся заново, для этого совершается проход по временному массиву в ходе которого в изначальный массив помещается значение k+min столько раз, какое значение записано в k ячейке временного массива.

Таблица 4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Входной массив | 6 | 4 | 7 | 2 | 3 | 3 | 5 |
| Временный массив | 1 | 2 | 1 | 1 | 1 | 1 |  |
| Результат | 2 | 3 | 3 | 4 | 5 | 6 | 7 |

### Быстрая сортировка

1. Любым способом выбираем ведущий элемент.
2. Переставляем значения в массиве, чтобы те, что меньше ведущего оказались слева, а те, что больше – справа (в случае с сортировкой по возрастанию).
3. После чего аналогично сортируем левую и правую часть, при этом массив из одного элемента считаем уже отсортированным, таким образом в какой-то момент все подмассивы окажутся отсортированными, тогда же будет отсортирован и изначальны массив.

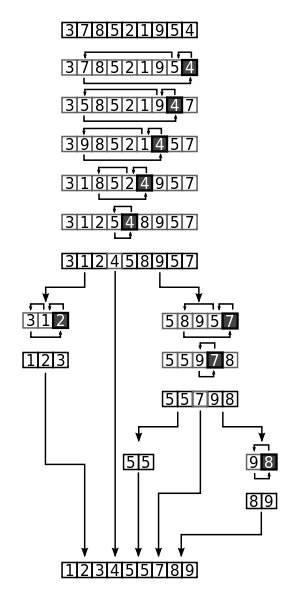


Рис. 6.

### Сортировка слиянием

1. Массив делится на две части, который сортируются тем же алгоритмом, при этом массивы из одного элемента считаются отсортированными.
2. После этого происходит слияние отсортированных подмассивов, размер которых постепенно увеличивается, пока не будет изначальный.

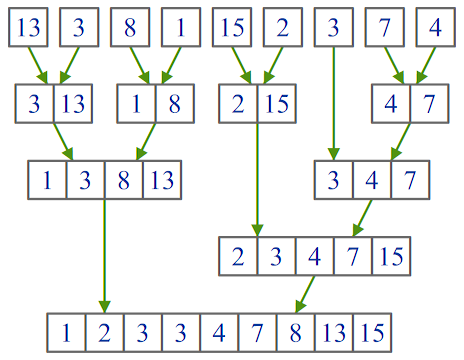


Рис. 7

## Описание функций

void choose\_sort(LONGLONG \*size, int \*index, int n)

Назначение: реализация сортировки выбором.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), n – количество файлов.

Выходные параметры: index – массив положений.

void insert\_sort(LONGLONG \*size, int \*index, int n)

Назначение: реализация сортировки вставками.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), n – количество файлов.

Выходные параметры: index – массив положений.

void bubble\_sort(LONGLONG \*size, int \*index, int n)

Назначение: реализация сортировки пузырьком.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), n – количество файлов.

Выходные параметры: index – массив положений.

void counting\_sort(LONGLONG \*size, int \*index, LONGLONG min, LONGLONG max, int n)

Назначение: реализация сортировки подсчётом.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), min – размер самого маленького файла, max - размер самого тяжёлого файла, n – количество файлов.

Выходные параметры: index – массив положений.

void quick\_sort(LONGLONG \*size, int \*index, int first, int last)

Назначение: Реализация быстрой сортировки.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), first – индекс первого элемента, last – индекс последнего элемента.

Выходные параметры: index – массив положений.

void merge\_sort(LONGLONG \*size, int \*index, int left, int right)

Назначение: Реализация быстрой сортировки.

Входные параметры: \*size – указатель на массив с размерами файлов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), left – индекс первого элемента, right – индекс последнего элемента.

Выходные параметры: index – массив положений.

int file\_counter(wchar\_t \*sDir);

Назначение: Подсчёт количества файлов в папке.

Входные параметры: sDir – строка, содержащая в себе путь до папки.

Выходные параметры: количество файлов в папке или -1 если она не найдена.

int file\_reader(wchar\_t \*sDir, LONGLONG \*size, char \*\*name, LONGLONG\* min, LONGLONG\* max);

Назначение: Чтение информации об именах файлов и их размерах.

Входные параметры: sDir – строка, содержащая в себе путь до папки, \*size – указатель на массив размеров файлов, \*\*name указатель на массив имён фалов, \*min – указатель на переменную содержащую в себе размер самого маленького файла, \*max – указатель на переменную содержащую в себе размер самого большого файла.

void printer(LONGLONG \*size, wchar\_t \*\*name, int \*index, int n, int mode);

Назначение: Вывод отсортированного списка файлов.

Входные параметры: \*size – указатель на массив с размерами файлов, \*\*name указатель на массив имён фалов, \*index – указатель на массив положений (указывает какой файл на какой позиции, относительно других, находится), n – количество файлов, mode – направление сортировки (по возрастанию или по убыванию).

void head\_printer();

Назначение: Вывод шапки программы.

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

Назначение: реализация ввода пути до папки с клавиатуры.

Входные параметры \*\*path – указатель на строку, в которой должен храниться путь.

Выходные параметры: path – строка, содержащая в себе введённый пользователем путь до папки.

# Заключение

В результате выполнения практической работы, был разработан прототип файлового менеджера, с возможностью отображения списка файлов, отсортированных с помощью одного из шести алгоритмов, в заданной папке.

В программе реализованы следующий сортировки: выбором, вставками, пузырьком, подсчётом, быстрая и слиянием. Чья сравнительная характеристика сложности представлена ниже.

|  |  |
| --- | --- |
| Сортировка | Сложность |
| Выбором | O(n2) |
| Вставками | O(n2) |
| Пузырьком | O(n2) |
| Подсчётом | O(n) |
| Быстрая | O(n⋅log n) |
| Слиянием | O(n⋅log n) |