МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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 УЧЕБНОЙ ПРАКТИКЕ**

**Прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию (убыванию) размера**

**Выполнил:** студент группы 381806-1

Алибеков Мурад Рамазанович

**Проверил:** кандидат технических наук доцент кафедры МОСТ ИИТММ

Кустикова Валентина Дмитриевна

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc533366042)

[1. Постановка задачи 4](#_Toc533366043)

[2. Руководство пользователя 5](#_Toc533366044)

[3. Руководство программиста 7](#_Toc533366045)

**[Структура программы](#_Toc533366046)** [7](#_Toc533366046)

**[Описание алгоритма](#_Toc533366047)** [7](#_Toc533366047)

[Сортировка выбором – Choose\_Sort 7](#_Toc533366048)

[Сортировка вставками – Insert\_Sort 7](#_Toc533366049)

[Пузырьковая сортировка – Bubble\_Sort 8](#_Toc533366050)

[Сортировка подсчетом – Counting\_Sort 9](#_Toc533366051)

[Быстрая сортировка – Quick\_Sort 1](#_Toc533366052)0

**[Описание функций](#_Toc533366053)** [1](#_Toc533366053)1

[Вывод отсортированного списка названий файлов с указанием размера 1](#_Toc533366056)1

[Сортировка выбором 1](#_Toc533366057)1

[Сортировка вставками 11](#_Toc533366058)

[Пузырьковая сортировка 11](#_Toc533366059)

[Сортировка подсчетом 11](#_Toc533366060)

[Быстрая сортировка 12](#_Toc533366062)

[Функция слияния 12](#_Toc533366063)

[Сортировка слиянием 12](#_Toc533366064)

[Заключение 13](#_Toc533366065)

**[Анализ методов сортировок](#_Toc533366066)** [13](#_Toc533366066)

[Литература 15](#_Toc533366067)

# **Введение**

# Что общего в упорядоченном списке имен, в библиотеке с песнями, в размещении объектов в хронологическом порядке, в сортировке наших писем в алфавитном порядке по имени отправителя или названию темы или же по дате получения? Все они требуют сортировки элементов. Редко встретишь компьютерное приложение, в котором не использовалась бы сортировка; о сути дела нужда в сортировке предшествовала появлению цифровых компьютеров. Т.к. сортировка встречается почти что вездеб не удивительно, что с давних пор выискивали методы сортировки. Исследования методов сортировки ведутся до сих пор, они касаются как улучшения уже известных методов, так и разработки новых алгоритмов.**Постановка задачи**

Разработать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию (убыванию) размера. Для этого необходимо реализовать следующие методы сортировки:

* Сортировка выбором
* Сортировка вставками
* Пузырьковая сортировка
* Сортировка подсчетом
* Сортировка слиянием
* Быстрая сортировка

Входные данные:

* Путь до директории, в которой необходимо отсортировать содержимое
* Метод сортировки

Выходные данные:

* Отсортированный список имен файлов с указанием размера
* Время сортировки

# **Руководство пользователя**

После запуска программа на экран выводится запрос на введение пути до директории, в которой необходимо отсортировать содержимое [Рисунок1] .

Снимок экрана (15)

Рисунок 1. Запрос на ввод пути до директории

В случае некорректного ввода пути или указания пути в пустую директорию запрос на ввод пути повторится [Рисунок 2].

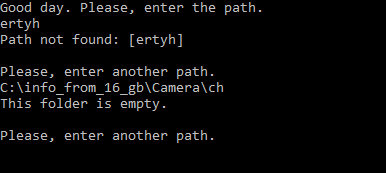


Рисунок 2. Пример неверного пути до директории и пути в пустую папку

В случае корректно введенных данных на экране появится список возможных сортировок. Для выбора одной из них необходимо ввести ее номер. Затем появится возможность выбора характеристики сортировки (по возрастанию (up) или по убыванию (down)). Для выбора характеристики необходимо ввести ее номер. В случае корректно введенных данных на экране появится время сортировки и список выбора дальнейших действий. [Рисунок 3].

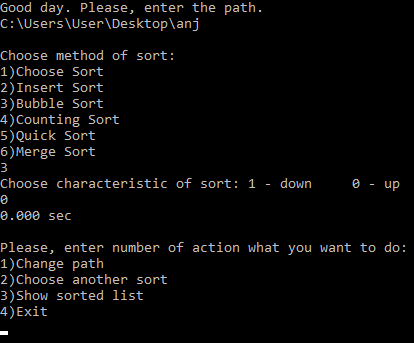


Рисунок 3. Список сортировок

В случае введения номера команды 2 на экране появится список возможных сортировок. [Рисунок 3].

В случае введения номера команды 1 на экране появится поле для ввода нового пути. [Рисунок 1].

Для завершения работы с программой необходимо выбрать 4 команду меню команд – Exit [Рисунок 3].

В случае введения номера команды 3 на экран будут выведены отсортированные названия файлов и их размеры в байтах, а также меню команд [Рисунок 4].

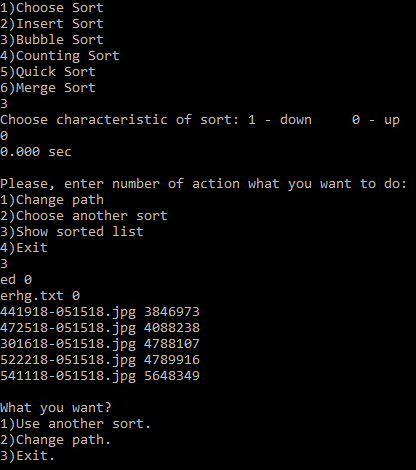


Рисунок 4. Пример выводимых данных

Для завершения работы с программой необходимо выбрать 3 команду меню команд – Exit [Рисунок 4].

# **Руководство программиста**

## **Структура программы**

Программа состоит из 1 файла *main.c,* который содержит функцию *main*, функции сортировок, функцию поиска количества файлов в директории, функцию заполняющую массивы имен файлов и их размеров и функцию вывода отсортированных названий файлов и их размеров в байтах.

## **Описание алгоритмов**

### *Сортировка выбором –* ***Choose\_Sort***

* Находим позицию минимального элемента в массиве
* Меняем минимальный элемент с первым
* Продолжаем для массива, начинающегося со следующего элемента [Таблица 1]

Таблица 1. Пример сортировки выбором

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Итерация | Массив | | | | | | | | Индекс минимального элемента | С каким элементом менять |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 1 | 2 | 4 | -1 | -2 | 10 | 1 | 9 | 0 | 3 | 0 |
| 2 | **-2** | 4 | -1 | 2 | 10 | 1 | 9 | 0 | 2 | 1 |
| 3 | **-2** | **-1** | 4 | 2 | 10 | 1 | 9 | 0 | 7 | 2 |
| 4 | **-2** | **-1** | **0** | 1 | 10 | 2 | 9 | 4 | 3 | 3 |
| 5 | **-2** | **-1** | **0** | **1** | 10 | 2 | 9 | 4 | 5 | 4 |
| 6 | **-2** | **-1** | **0** | **1** | **2** | 10 | 9 | 4 | 7 | 5 |
| 7 | **-2** | **-1** | **0** | **1** | **2** | **4** | 9 | 10 | 6 | 6 |
| 8 | **-2** | **-1** | **0** | **1** | **2** | **4** | **9** | 10 | 7 | 7 |
| - | **-2** | **-1** | **0** | **1** | **2** | **4** | **9** | **10** | - | - |

### *Сортировка вставками –* ***Insert\_Sort***

* Начинаем с массива из одного (первого) элемента, считая, что он отсортирован.
* Каждый следующий проход увеличиваем длину рассматриваемого массива на 1.
* Проходим массив от последнего элемента к первому.
* Пока (j-1)-й элемент больше j-того, меняем их местами [Таблица2].

Таблица 2. Пример сортировки вставками

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Итерация | Массив | | | | | | | | Индекс сравниваемого элемента | Нужно ли менять с предыдущим? |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 1 | **5** | 9 | 8 | 2 | 6 | 3 | 1 | 4 | 1 | NO |
| 2 | **5** | **9** | 8 | 2 | 6 | 3 | 1 | 4 | 2 | YES |
| **5** | 8 | **9** | 2 | 6 | 3 | 1 | 4 | 1 | NO |
| 3 | **5** | **8** | **9** | 2 | 6 | 3 | 1 | 4 | 3 | YES |
| **5** | **8** | 2 | **9** | 6 | 3 | 1 | 4 | 2 | YES |
| **5** | 2 | **8** | **9** | 6 | 3 | 1 | 4 | 1 | YES |
| 2 | **5** | **8** | **9** | 6 | 3 | 1 | 4 | 0 | NO |
| 4 | **2** | **5** | **8** | **9** | 6 | 3 | 1 | 4 | 4 | YES |
| **2** | **5** | **8** | 6 | **9** | 3 | 1 | 4 | 3 | YES |
| **2** | **5** | 6 | **8** | **9** | 3 | 1 | 4 | 2 | NO |
| 5 | **2** | **5** | **6** | **8** | **9** | 3 | 1 | 4 | 5 | YES |
| **2** | **5** | **6** | **8** | 3 | **9** | 1 | 4 | 4 | YES |
| **2** | **5** | **6** | 3 | **8** | **9** | 1 | 4 | 3 | YES |
| **2** | **5** | 3 | **6** | **8** | **9** | 1 | 4 | 2 | YES |
| **2** | 3 | **5** | **6** | **8** | **9** | 1 | 4 | 1 | NO |
| 6 | **2** | **3** | **5** | **6** | **8** | **9** | 1 | 4 | 6 | YES |
| **2** | **3** | **5** | **6** | **8** | 1 | **9** | 4 | 5 | YES |
| **2** | **3** | **5** | **6** | 1 | **8** | **9** | 4 | 4 | YES |
| **2** | **3** | **5** | 1 | **6** | **8** | **9** | 4 | 3 | YES |
| **2** | **3** | 1 | **5** | **6** | **8** | **9** | 4 | 2 | YES |
| **2** | 1 | **3** | **5** | **6** | **8** | **9** | 4 | 1 | YES |
| 1 | **2** | **3** | **5** | **6** | **8** | **9** | 4 | 0 | NO |
| 7 | **1** | **2** | **3** | **5** | **6** | **8** | **9** | 4 | 7 | YES |
| **1** | **2** | **3** | **5** | **6** | **8** | 4 | **9** | 6 | YES |
| **1** | **2** | **3** | **5** | **6** | 4 | **8** | **9** | 5 | YES |
| **1** | **2** | **3** | **5** | 4 | **6** | **8** | **9** | 4 | YES |
| **1** | **2** | **3** | 4 | **5** | **6** | **8** | **9** | 3 | NO |
| - | **1** | **2** | **3** | **4** | **5** | **6** | **8** | **9** | - | - |

### *Пузырьковая сортировка –* ***Bubble\_Sort***

* Проходим массив с первого элемента до n-ого (где n – количество элементов в массиве).
* Сравниваем попарно стоящие рядом элементы (1 и 2, 2 и 3, ..., n–1 и n).
* Если первый элемент пары больше второго, меняем элементы местами.
* Для каждого следующего прохода количество элементов уменьшается на 1 [Таблица 3].

Таблица 3. Пример пузырьковой сортировки

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Итерация | Массив | | | | | | | | Индексы сравниваемых элементов | Нужно ли менять элементы местами? |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 1 | 8 | 6 | 4 | 2 | 9 | 5 | 3 | 7 | 0, 1 | YES |
| 6 | 8 | 4 | 2 | 9 | 5 | 3 | 7 | 1, 2 | YES |
| 6 | 4 | 8 | 2 | 9 | 5 | 3 | 7 | 2, 3 | YES |
| 6 | 4 | 2 | 8 | 9 | 5 | 3 | 7 | 3, 4 | NO |
| 6 | 4 | 2 | 8 | 9 | 5 | 3 | 7 | 4, 5 | YES |
| 6 | 4 | 2 | 8 | 5 | 9 | 3 | 7 | 5, 6 | YES |
| 6 | 4 | 2 | 8 | 5 | 3 | 9 | 7 | 6, 7 | YES |
| 2 | 6 | 4 | 2 | 8 | 5 | 3 | 7 | **9** | 0, 1 | YES |
| 4 | 6 | 2 | 8 | 5 | 3 | 7 | **9** | 1, 2 | YES |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | **9** | 2, 3 | NO |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | **9** | 3, 4 | YES |
| 4 | 2 | 6 | 5 | 8 | 3 | 7 | **9** | 4, 5 | YES |
| 4 | 2 | 6 | 5 | 3 | 8 | 7 | **9** | 5, 6 | YES |
| 3 | 4 | 2 | 6 | 5 | 3 | 7 | **8** | **9** | 0, 1 | YES |
| 2 | 4 | 6 | 5 | 3 | 7 | **8** | **9** | 1, 2 | NO |
| 2 | 4 | 6 | 5 | 3 | 7 | **8** | **9** | 2, 3 | YES |
| 2 | 4 | 5 | 6 | 3 | 7 | **8** | **9** | 3, 4 | YES |
| 2 | 4 | 5 | 3 | 6 | 7 | **8** | **9** | 4, 5 | NO |
| 4 | 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 0, 1 | NO |
| 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 1, 2 | NO |
| 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 2, 3 | YES |
| 2 | 4 | 3 | 5 | 6 | **7** | **8** | **9** | 3, 4 | NO |
| 5 | 2 | 4 | 3 | 5 | **6** | **7** | **8** | **9** | 0, 1 | NO |
| 2 | 4 | 3 | 5 | **6** | **7** | **8** | **9** | 1, 2 | YES |
| 2 | 3 | 4 | 5 | **6** | **7** | **8** | **9** | 2, 3 | NO |
| 6 | 2 | 3 | 4 | **5** | **6** | **7** | **8** | **9** | 0, 1 | NO |
| 2 | 3 | 4 | **5** | **6** | **7** | **8** | **9** | 1, 2 | NO |
| 7 | 2 | 3 | **4** | **5** | **6** | **7** | **8** | **9** | 0, 1 | NO |
| - | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | - | - |

### *Сортировка подсчетом –* ***Counting\_Sort***

* Создаем дополнительный массив и обнуляем его.
* Проходим по всему исходному массиву и прибавляем 1 к элементу нового массива, индекс которого равен значению элемента исходного массива.
* Перезаписываем исходный массив, при этом записываем элемент равный индексу нового массива столько раз, сколько равен элемент нового массива с этим индексом [Таблица 4].

Таблица 4. Пример сортировки подсчетом

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *min = 0*  *max = 3* | Сортируемый массив | | | | | | | | Дополнительный массив | | | |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *0* | *1* | *2* | *3* |
| Итерации | 3 | 2 | 3 | 0 | 1 | 1 | 3 | 3 | 0 | 0 | 0 | 0 |
| 1 | ↑ |  |  |  |  |  |  |  | 0 | 0 | 0 | 1 |
|  | ↑ |  |  |  |  |  |  | 0 | 0 | 1 | 1 |
|  |  | ↑ |  |  |  |  |  | 0 | 0 | 1 | 2 |
|  |  |  | ↑ |  |  |  |  | 1 | 0 | 1 | 2 |
|  |  |  |  | ↑ |  |  |  | 1 | 1 | 1 | 2 |
|  |  |  |  |  | ↑ |  |  | 1 | 2 | 1 | 2 |
|  |  |  |  |  |  | ↑ |  | 1 | 2 | 1 | 3 |
|  |  |  |  |  |  |  | ↑ | 1 | 2 | 1 | 4 |
| 2 | 0 | 2 | 3 | 0 | 1 | 1 | 3 | 3 | ↑ |  |  |  |
| **0** | 1 | 1 | 0 | 1 | 1 | 3 | 3 |  | ↑ |  |  |
| **0** | **1** | **1** | 2 | 1 | 1 | 3 | 3 |  |  | ↑ |  |
| **0** | **1** | **1** | **2** | 3 | 3 | 3 | 3 |  |  |  | ↑ |
| - | **0** | **1** | **1** | **2** | **3** | **3** | **3** | **3** | - | | | |

### *Быстрая сортировка –* ***Quick\_Sort***

* Выбираем ведущий элемент.
* Все меньшие элементы перемещаем влево от него, большие – вправо.
* Рекурсивно продолжаем для левой и правой половин, пока размер подмассива не станет меньше [Таблица 5].

Таблица 5. Пример быстрой сортировки

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | Массив | | | | | | | |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 1 | 3 | 7 | 4 | 6 | 4 | 5 | 8 | 7 |
| 3 | 5 | 4 | 6 | 4 | 7 | 8 | 7 |
| 2 | 3 | 5 | 4 | 4 | **6** | 7 | 8 | 7 |
| 3 | 4 | 4 | **5** | 7 | 7 | **8** |
| 3 | 3 | 4 | 4 | 7 | 7 |
| - | **3** | **4** | **4** | **5** | **6** | **7** | **7** | **8** |

*Сортировка слиянием* – *Merge\_Sort*

* Рекурсивно разбиваем исходный массив на подмассивы из одного элемента.
* Рекурсивно попарно сливаем подмассивы.
* На каждом этапе слияния упорядочиваем подмассивы [Таблица 6].

Таблица 6. Пример сортировки слиянием

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | Массив | | | | | | | |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* |
| 1 | 2 | 8 | 4 | 1 | 0 | 2 | 8 | 3 |
| 2 | 8 | 4 | 1 | 0 | 2 | 8 | 3 |
| 2 | 8 | 1 | 4 | 0 | 2 | 8 | 3 |
| 2 | 8 | 1 | 4 | 0 | 2 | 8 | 3 |
| 2 | 2 | 8 | 1 | 4 | 0 | 2 | 3 | 8 |
| 1 | 2 | 4 | 8 | 0 | 2 | 3 | 8 |
| 3 | 1 | 2 | 4 | 8 | 0 | 2 | 3 | 8 |
| - | **0** | **1** | **2** | **2** | **3** | **4** | **8** | **8** |

## **Описание функций**

### *Вывод на экран отсортированного списка названий файлов с указанием размера:*

void output(ULL sizes[], ULL ind[], word \*names)

* Входные параметры

\*sizes - “массив” размеров файлов

\*ind - “массив” индексов

\*names - “массив слов”(имена файлов)

* Выходные параметры отсутствуют

### *Сортировка выбором*

void Choose\_Sort(ULL a[] , ULL n)

* Входные параметры

\*a - “массив”-клон размеров файлов

n - длина этого массива

* Выходные параметры отсутствуют

### *Сортировка вставками*

void Insert\_Sort(ULL a[], ULL n)

* Входные параметры

\*a - “массив”-клон размеров файлов

n - длина этого массива

* Выходные параметры отсутствуют

### *Пузырьковая сортировка*

void Bubble\_Sort(ULL a[], ULL n)

* Входные параметры

\*a - “массив”-клон размеров файлов

n - длина этого массива

* Выходные параметры отсутствуют

### *Сортировка подсчетом*

void Counting\_Sort(ULL a[], ULL n)

* Входные параметры

\*a - “массив”-клон размеров файлов

n - длина этого массива

* Выходные параметры отсутствуют

### *Быстрая сортировка*

void Quick\_Sort(ULL a[], ULL left, ULL right)

* Входные параметры

\*a - “массив”-клон размеров файлов

left - индекс левой границы “подмассива” a[]

right - индекс правой границы “подмассива” a[]

* Выходные параметры отсутствуют

### *Функция слияния двух упорядоченных “подмассивов” в массиве*

void Merge(ULL \*a, ULL left, ULL mid, ULL right)

* Входные параметры

\*a - “массив”-клон размеров файлов

left - индекс левой границы “подмассива” a[]

right - индекс правой границы “подмассива” a[]

* Выходные параметры отсутствуют

### *Сортировка слиянием*

void Merge\_Sort(ULL a[], ULL left, ULL right)

* Входные параметры

\*a - “массив”-клон размеров файлов

left - индекс левой границы “подмассива” a[]

right - индекс правой границы “подмассива” a[]

* Выходные параметры отсутствуют

# *Поиск количества файлов*

# ULL Count\_of\_files(const wchar\_t \*sDir)

* Входные параметры

\*sDir - приведенное к типу wchar\_t слово, обозначающее путь до директории

* Выходные параметры

Целое число типа long long int

# *Заполнение данными массивов с именами и размерами файлов*

int GetInfoAboutFiles(const wchar\_t \*sDir, char \*\*\*names, ULL \*sizes)

* Входные параметры

\*sDir - приведенное к типу wchar\_t слово, обозначающее путь до директории

\*\*\*name - указатель на массив слов - названий файлов

\*sizes - “массив” размеров файлов

* Выходные параметры отсутствуют

# **Заключение**

Разработан прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию размера. Реализованы следующие методы сортировок:

• Сортировка выбором

• Сортировка вставками

• Пузырьковая сортировка

• Сортировка подсчетом

• Сортировка слиянием

• Быстрая сортировка

## **Анализ методов сортировок**

В таблице ниже приведена сравнительная характеристика сложности работы данных алгоритмов.

Таблица 7. Пример сортировки слиянием

|  |  |  |
| --- | --- | --- |
| Методы сортировки | Средняя сложность алгоритма | |
| Время выполнения | Память |
| Сортировка выбором | O(n2) | O(1) |
| Сортировка простыми вставками | O(n2) | O(1) |
| Пузырьковая сортировка | O(n2) | O(1) |
| Сортировка подсчетом | O(n) | max(n) – min(n) + 1 |
| Быстрая сортировка | O(n·log2 n) | O(1) |
| Сортировка слиянием | O(n·log2 n) | O(n) |

На графиках ниже показана зависимость времени сортировки от количества элементов для разных оценок. По горизонтальной оси располагается количество элементов.

# **Литература**

* <https://ru.wikipedia.org/>
* Курс лекций по Основам Программирования 1-ого семестра в 2018-2019 учебных годах направления ФИИТ в Институте информационных технологий, математики и механики в ННГУ им. Лобачевского.
* П. Луридас. Алгоритмы для начинающих. Теория и практика для разработчика. — Издательство “Эксмо”, 2018. — 607 с. — ISBN: 978-5-04-089834-3