Министерство науки и высшего образования РФ

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

Институт информационных технологий, математики и механики

Отчет по учебной практике

**Прототип файлового менеджера с функцией показа файлов в заданном каталоге**

Выполнил:

студент гр. 381806-1

Емельховская Е. Е.

Проверил:

Доцент кафедры МОСТ ИТММ

Кустикова В.Д.

Нижний Новгород

2018 г.

**Содержание**

[Введение 2](#_Toc533405990)

[1. Постановка задачи 3](#_Toc533405991)

[2. Руководство пользователя 4](#_Toc533405992)

[3. Руководство программиста 7](#_Toc533405993)

[3.1. Структура программы 7](#_Toc533405994)

[3.2. Описание алгоритмов 7](#_Toc533405995)

[3.3. Описание функций 13](#_Toc533405996)

[Заключение 18](#_Toc533405997)

[Литература 21](#_Toc533405998)

# Введение

Компьютер – неотъемлемая часть жизни человека. И каждый день нам приходится сталкиваться с проблемой поиска файлов. Каждый раз глаза «разбегаются в стороны» от огромного и беспорядочного списка файлов. В такой ситуации вряд ли получится быстро что-либо найти. Как же упростить этот процесс? Специально для этого в этой работе я осуществила работу файлового менеджера, сортирующего файлы по объёму. Данная программа разработана не только для дома, но и с перспективой использования в крупных компаниях в связи со своей актуальностью. Параллельно с этим проведем исследование зависимости времени сортировки от выбранного метода.

# Постановка задачи

Разработать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию размера, используя различный методы сортировок, а именно:

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

Входные данные:

* Путь до необходимой директории
* Метод сортировки

Выходные данные:

* Отсортированный по возрастанию список файлов(с размерами в байтах) в заданной директории
* Время сортировки

# Руководство пользователя

Чтобы включить программу необходимо нажать сочетание клавиш ctrl+F5.

После запуска программы пользователь видит окно (Рис. 1) с названием программы и просьбой ввести путь к папке. Ввод осуществляется через символы английского алфавита.

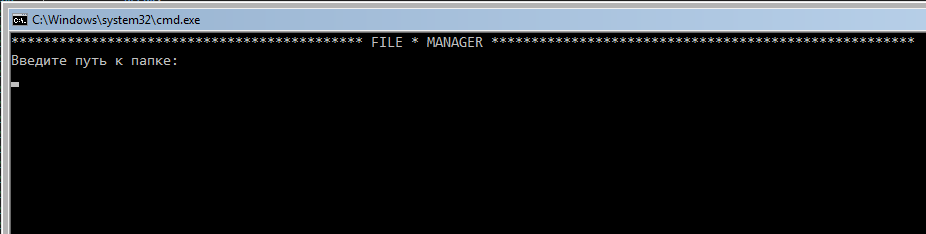


Рис. 1

Например, если у вас на компьютере есть папка Windows и вы хотите вывести файлы из нее, введите путь в таком формате: C:\Windows\ (Рис. 2).

После этого появится сообщение о количестве файлов.

Далее пользователю будет предложен список методов сортировки, из которых нужно выбрать один метод посредством ввода нужной цифры. Конец ввода завершается нажатием клавиши Enter.

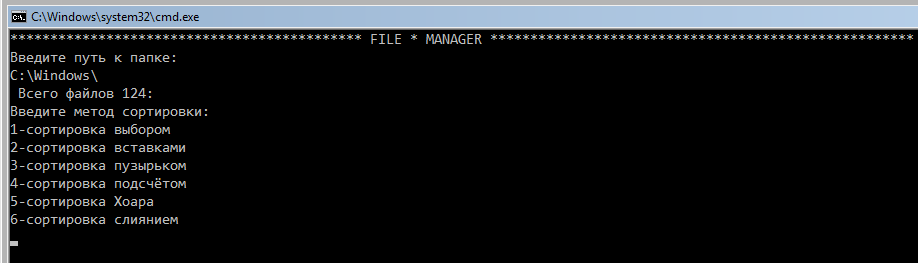


Рис. 2

После ввода отобразится список файлов, отсортированных по объему в байтах (Рис. 3). Кроме того, появится время выполнения данной сортировки. Для примера выберем сортировку выбором и введем цифру 1. Далее работа с программой не составит труда ☺

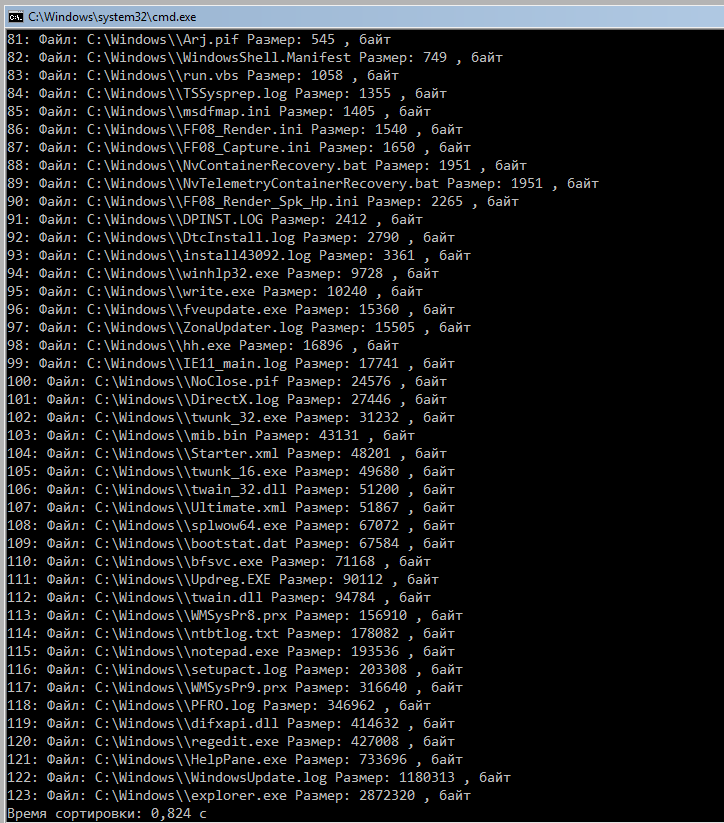


Рис. 3

# Руководство программиста

## Структура программы

Программа состоит из файла sortirovki.c.

## Описание алгоритмов

**Алгоритмы сортировок:**

P.S.Данные сортировки рассмотрены для ситуации сортировки по возрастанию

1. **Сортировка выбором**

Для сортировки выбором берется массив элементов, сначала находим для него минимальный элемент, меняем его местами с [0](нулевым элементом). Среди оставшихся элементов(нулевой элемент уже не трогаем) находим опять минимум, меняем его с [1](первым элементом). Продолжаем аналогичные операции, пока массив не закончится.

**Пример:**

Минимальный элемент (2), меняем его местами с нулевым элементом(7), ищем минимальный элемент уже начиная с первого, это будет (3), меняем ее с первым элементом (10) и так далее. (Рис. 4)



Рис.

1. **Сортировка вставками**

Чтобы отсортировать массив методом вставок, необходимо провести количество операций, равное количеству элементов в массиве. В каждой такой операции рассматривается подмассив. Изначально берем нулевой элемент. Это нулевой подмассив, можно считать, он уже отсортирован. Далее рассматриваем первый подмассив уже из двух элементов. Если последний элемент меньше, меняем их местами. Дальше подмассив опять увеличивается. В итоге получаем отсортированный подмассив и элемент, который мы должны сравнить с предыдущим элементом и если он больше- поменять местами. Все это делаем в цикле.

**Пример:**

Сразу рассмотрим подмассив из двух элементов(5,2). (2)<(5), меняем их местами.

Рассмотрим подмассив из трех элементов(2,5,4). (4)<(5), меняем их местами. Но (4)>(2), значит местами менять их не нужно. Далее аналогично.

(Рис. 5).

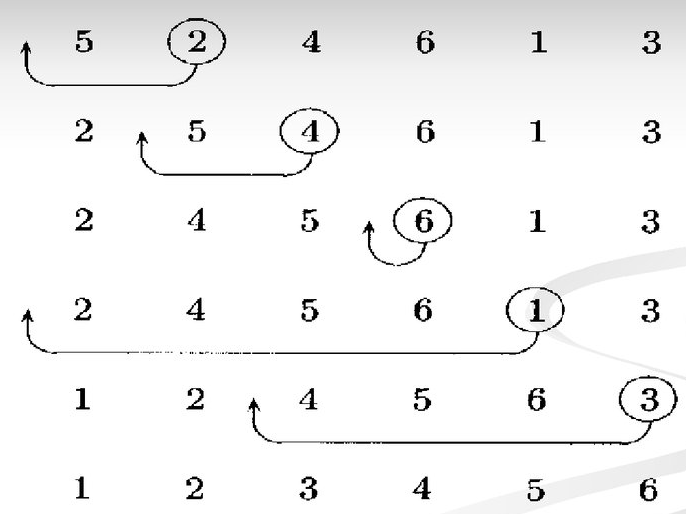


Рис. 5

3. **Пузырьковая сортировка**

Пузырьковая сортировка состоит из «всплытий». Один проход по массиву соответствует «всплытию» наибольшего элемента из неотсортированного подмассива. За один проход соседние элементы меняются местами в зависимости от того, кто из них больше.

**Пример:**

Первый проход: сравниваем нулевой(2) и первый(0) элементы, меняем их местами. Сравниваем первый(2) и второй(-2), меняем их местами. Сравниваем второй(2) и третий(5), не меняем их. Если продолжить, то окажется, что самый большой элемент «всплывет» наверх. Останется проделать такие же операции с оставшейся частью массива. (Рис. 6).

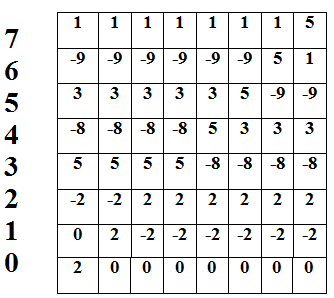


Рис. 6

1. **Сортировка подсчетом**

Для сортировки подсчетом создается дополнительный массив и обнуляется. Идея сортировки в том, чтобы пройтись по основному массиву, взять оттуда значение элемента(обозначим его\*) и увеличить значение нового массива с индексом \* на единицу.

**Пример:**

Диапазон значений массива от 0 до 5, значит создаем дополнительный массив размером 6. Проходимся по массиву. Нулевой элемент(0), значит увеличиваем на единицу нулевой элемент дополнительного массива. Дальше идет (2), увеличиваем второй элемент дополнительного массива на единицу и так далее.

(рис. 7).



Рис. 7

5. **Быстрая сортировка**

Выбираем ключевой средний элемент. Элементы меньше него убираем влево от него, элементы больше - вправо. Далее рекурсивно аналогично проделываем те же операции с оставшимися половинками.

**Пример:**

Сделаем элемент со значением (67) ключевым, а так же введем два счетчика L и R. Каждый раз когда элементы стоят на своем месте (если слева, то меньше, если справа, то больше) увеличиваем/уменьшаем счетчики, соответственно. Сначала двигаемся слева.(78)>(34), меняем их местами(т.к. они не на своих местах).(6)<(67), увеличиваем L. (44)<(76), а (82)>(76), поэтому меняем их местами. Когда закончим сортируем таким методом каждую половину отдельно. (Рис. 8).

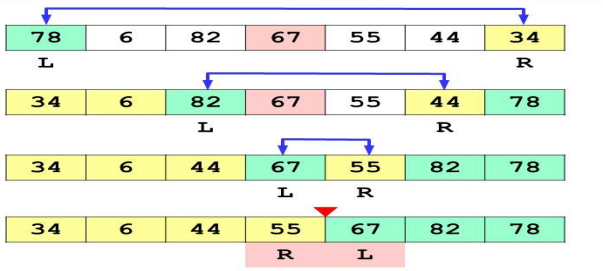


Рис. 8

6. **Сортировка слиянием**

В этой сортировке наша задача- рекурсивно разделить массив на подмассивы из одного элемента, а дальше сортируя, попарно их сливать в один.

**Пример:**

Дан такой массив. Рекурсивно разбиваем на подмассивы из одного элемента. Рассмотрим, например подмассивы(1267) и (0345). Возьмем 2 счетчика для каждого из них (i,j, например). Сейчас i=j=0. Берем (1) и (0), (1)>(0), тогда записываем (0) в следующий подмассив, j увеличиваем на единичку и теперь уже сравниваем (3) и (1) аналогично.

(рис. 9)

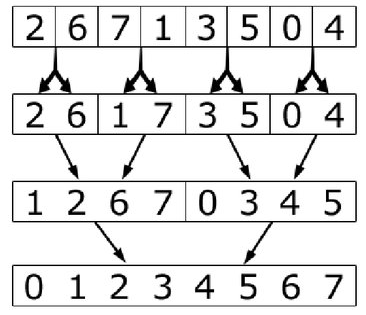


Рис. 9

## Описание функций

void INPUT\_PATH(wchar\_t \*\*sDir)

* **Назначение**

Ввод пути до каталога

* **Входные параметры**

\*\*sDir – указатель на массив, куда будет введен путь до каталога

* **Выходные параметры**

отсутствуют

int DIRECTORY(const wchar\_t \*sDir, wchar\_t \*\*\*fName, ULONGLONG \*\*fSize)

* **Назначение**

Определение количества файлов в директории, формирование и заполнение массивов названий файлов и их размеров

* **Входные параметры**

\*sDir – путь до каталога

\*\*fSize – адрес массива размеров файлов

\*\*\*name – адрес массива названий

* **Выходные параметры**

i – количество файлов в каталоге

void OUTPUT(int \*fIndex, wchar\_t \*\*fName, ULONGLONG \*fSize, int kol)

* **Назначение**

Вывод на экран списка имен файлов и их размерами

* **Входные параметры**

\*fIndex – массив индексов отсортированного массива размеров файлов

\*fSize – массив размеров файлов

\*\*fName – массив названий файлов

kol- количество файлов

* **Выходные параметры**

Отсутствуют

void SORT\_CHOICE(ULONGLONG \*sizes, int \*idxes, int kol)

* **Назначение**

Сортировка выбором (тут и всюду далее меняем массив индексов)

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

kol – количество файлов

* **Выходные параметры**

Отсутствуют

void SORT\_INSERT(ULONGLONG \*sizes, int \*idxes, int kol)

* **Назначение**

Сортировка вставками

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

kol – количество файлов

* **Выходные параметры**

Отсутствуют

void SORT\_BUBBLE(ULONGLONG \*sizes, int \*idxes, int kol)

* **Назначение**

Пузырьковая сортировка

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

kol – количество файлов

* **Выходные параметры**

Отсутствуют

void SORT\_COUNT(ULONGLONG \*sizes, int \*idxes, int kol)

* **Назначение**

Сортировка подсчетом с учетом слишком больших размеров

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

kol – количество файлов

* **Выходные параметры**

Отсутствуют

void SORT\_HOARA2(ULONGLONG \*sizes, int \*idxes, int \*i, int \*j, ULONGLONG m)

* **Назначение**

Одна из функций быстрой сортировки, необходимая для переноса меньших влево и больших вправо

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

\*i – адрес первого индекса

\*j – адрес второго индекса

m – ключевой элемент

* **Выходные параметры**

Отсутствуют

void SORT\_HOARA1(ULONGLONG \*sizes, int \*idxes, int n1, int n2)

* **Назначение**

Одна из функций быстрой сортировки для рекурсивного продолжения сортировки

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

n1 – индекс первого элемента

n2 – индекс последнего элемента

* **Выходные параметры**

Отсутствуют

void SORT\_MERGE2(ULONGLONG \*sizes, int \*idxes, int l, int m, int r)

* **Назначение**

Функция для сортировки слиянием, соединяющая подмассивы, упорядочивая их.

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

l – индекс первого(левого) элемента

m – индекс среднего

r – индекс последнего(правого) элемента

* **Выходные параметры**

Отсутствуют

void SORT\_MERGE1(ULONGLONG \*sizes, int \*idxes, int l, int r)

* **Назначение**

Функция сортировки слиянием для рекурсивного деления

* **Входные параметры**

\*idxes – массив индексов массива размеров файлов

\*fSize – массив размеров файлов

l –индекс первого элемента

r –индекс последнего элемента

* **Выходные параметры**

Отсутствуют

# Заключение

Таким образом, в данной работе был создан прототип файлового менеджера, который может отсортировать файлы из введенного каталога по возрастанию размеров файлов шестью сортировками.

**Анализ методов сортировок**

|  |  |  |  |
| --- | --- | --- | --- |
| Кол-во файлов: | 124 | 534 | 1002 |
| Сортировка выбором | 0 | 0,001 | 0,015 |
| Сортировка вставками | 0 | 0,001 | 0,015 |
| Сортировка пузырьком | 0,001 | 0,006 | 0,044 |
| Сортировка подсчетом | 3,567 | 6,123 | - |
| Быстрая сортировка | 0 | 0 | 0,001 |
| Сортировка слиянием | 0 | 0 | 0,002 |

Было проведено исследование путем сравнения времени, затрачиваемого на каждую из сортировок при введении одного и того же пути.

# Исходя из графика и таблицы можно сделать вывод о том, что сортировка подсчетом самая неэффективная. Также можно составить список быстроты работы сортировок по убыванию:

1. Быстрая сортировка
2. Сортировка подсчетом
3. Сортировка слиянием
4. Сортировка вставками
5. Сортировка выбором
6. Пузырьковая сортировка

# Литература

Интернет – ресурсы:

* Картинки алгоритмов - https://yandex.ru/images/