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

# **Отчет по учебной практике**

**Методы сортировки**

Выполнил:

Студент группы 381806-3

Гурылев Н.С.

Проверил:

Доцент кафедры МОСТ ИИТММ

Кустикова В.Д.

Нижний Новгород

2018г.

# **Содержание**

[**1** **Отчет по учебной практике** 1](#_Toc533671122)

[**2** **Содержание** 2](#_Toc533671123)

[**3** **Введение** 3](#_Toc533671124)

[**4** **Постановка задачи** 4](#_Toc533671125)

[**5** **Руководство пользователя** 5](#_Toc533671126)

[**6** **Руководство программиста** 6](#_Toc533671127)

[**6.1** **Структура программы** 6](#_Toc533671128)

[**6.2** **Описание алгоритма** 6](#_Toc533671129)

[**6.2.1** **Пузырьковая сортировка** 6](#_Toc533671130)

[**6.2.2** **Сортировка выбором** 7](#_Toc533671131)

[**6.2.3** **Сортировка подсчетом** 7](#_Toc533671132)

[**6.2.4** **Сортировка вставками** 8](#_Toc533671133)

[**6.2.5** **Быстрая сортировка (Хоара)** 9](#_Toc533671134)

[**6.2.6** **Сортировка слиянием** 9](#_Toc533671135)

[**7** **Описание функций** 10](#_Toc533671136)

[**8** **Заключение** 13](#_Toc533671137)

[**9** **Литература** 14](#_Toc533671138)

# **Введение**

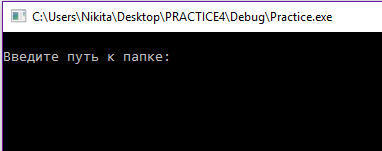
Под сортировкой понимается процесс перестановки объектов какого-то множества в определённом порядке. Цель сортировки – облегчить последующий поиск элементов в отсортированном множестве. Сортировка является примером огромного разнообразия алгоритмов, выполняющих одну и ту же задачу, многие из которых являются оптимальными, имея какое-то преимущество над остальными. Некоторые из этих сортировок и показаны в данной работе на примере файлового менеджера, сортирующего файлы по их размеру.

# **Постановка задачи**

В данной работе предполагается реализовать одну функцию файлового менеджера – сортировка файлов/директорий по возрастанию/убыванию объема хранимых в них данных. Необходимо оценить эффективность упорядочивания в зависимости от выбора алгоритма сортировки.

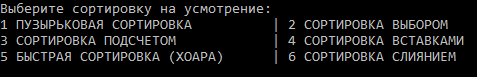
# **Руководство пользователя**

После запуска программы пользователю нужно ввести путь до директории, в которой нужно осуществить сортировку. Например, нужно отсортировать файлы в директории H:\Steam. Вводим в поле данный путь и нажимаем Enter.(рис.1)



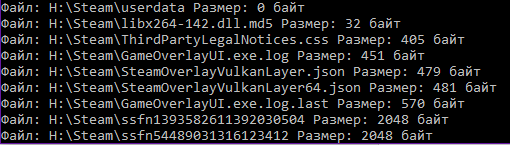
1. Ввод пути

Далее программа предлагает выбрать сортировку на усмотрение юзера.(рис.2)



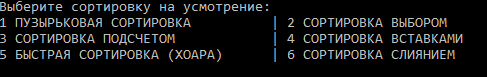
1. Выбор сортировки

Выбираем сортировку. Программа выводит на экран отсортированные файлы с их размерами в указанных единицах измерения. (рис.3)



1. Отсортированные файлы

Далее можно выбрать любую другую сортировку.(рис.4)



1. Меню сортировок

# **Руководство программиста**

## **Структура программы**

Программа состоит из функции сортировок, функции вывода меню команд, функции ввода директории, функции вывода отсортированных названий файлов, их размеров (в байтах) и времени сортировки и из файла main.

## **Описание алгоритма**

### **Пузырьковая сортировка**

На первом шаге сортировки проходим по массиву от 0 до предпоследнего элемента, попарно сравнивая соседние элементы (в этом примере 0 и 1, 1 и 2, 2 и 3, 3 и 4, 4 и 5, 5 и 6). Если первый элемент пары больше второго, меняем эти элементы местами. С каждым последующим шагом количество сравниваемых элементов уменьшается на 1.

Таблица **1**. Пример сортировки пузырьком

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 6 | 9 | 3 | -5 | 8 | 2 | -4 |
| 2 | 6 | 9 | 3 | -5 | 8 | 2 | -4 |
| 6 | 3 | 9 | -5 | 8 | 2 | -4 |
| 6 | 3 | -5 | 9 | 8 | 2 | -4 |
| 6 | 3 | -5 | 8 | 9 | 2 | -4 |
| 6 | 3 | -5 | 8 | 2 | 9 | -4 |
| 6 | 3 | -5 | 8 | 2 | -4 | 9 |
| 3 | 6 | -5 | 3 | 8 | 2 | -4 | 9 |
| 6 | -5 | 3 | 2 | 8 | -4 | 9 |
| 6 | -5 | 3 | 2 | -4 | 8 | 9 |
| 3 | -5 | 6 | 3 | 2 | -4 | 8 | 9 |
| -5 | 3 | 6 | 2 | -4 | 8 | 9 |
| -5 | 3 | 2 | 6 | -4 | 8 | 9 |
| -5 | 3 | 2 | -4 | 6 | 8 | 9 |
| 4 | -5 | 2 | 3 | -4 | 6 | 8 | 9 |
| -5 | 2 | -4 | 3 | 6 | 8 | 9 |
| 5 | -5 | -4 | 2 | 3 | 6 | 8 | 9 |
| Отсортированный массив | -5 | -4 | 2 | 3 | 6 | 8 | 9 |

### **Сортировка выбором**

В массиве находим позицию минимального элемента, этот минимальный элемент меняем с первым, далее в неотсортированной части снова ищем минимальный элемент. И так далее, до тех пор, пока весь массив не будет отсортирован.

Таблица 2. Пример сортировки выбором

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | -1 | 4 | 9 | -3 | 8 | -8 | 0 |
| 2 | -8 | 4 | 9 | -3 | 8 | -1 | 0 |
| 3 | -8 | -3 | 9 | 4 | 8 | -1 | 0 |
| 4 | -8 | -3 | -1 | 4 | 8 | 9 | 0 |
| 5 | -8 | -3 | -1 | 0 | 8 | 9 | 4 |
| 6 | -8 | -3 | -1 | 0 | 4 | 9 | 8 |
| 7 | -8 | -3 | -1 | 0 | 4 | 8 | 9 |
| Отсортированный массив | -8 | -3 | -1 | 0 | 4 | 8 | 9 |

### **Сортировка подсчетом**

Для сортировки подсчетом нужно создать вспомогательный массив (Таблица 5), заполнив его нулями. Проходим по всему исходному массиву и прибавляем 1 к элементу нового массива, индекс которого равен значению элемента исходного массива. После заполнения вспомогательного массива заполняем исходный массив, записывая элемент равный индексу нового массива столько раз, сколько элементов нового массива с этим индексом.

Таблица 3. Пример сортировки подсчетом

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 2 | 1 | 0 | 1 | 2 | 3 | 2 |
| *0* | 1 | 0 | 1 | 2 | 3 | 2 |
| 0 | *1* | *1* | 1 | 2 | 3 | 2 |
| 0 | 1 | 1 | *2* | *2* | *2* | 2 |
| 0 | 1 | 1 | 2 | 2 | 2 | *3* |
| Отсортированный массив | 0 | 1 | 1 | 2 | 2 | 2 | 3 |

Таблица 4. Вспомогательная таблица для сортировки подсчетом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Значение элемента  Элементы | Вспомогательный массив | | | |
| 0 | 1 | 2 | 3 |
|  | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0,1 | 0 | 1 | 1 | 0 |
| 0,1,2 | 1 | 1 | 1 | 0 |
| 0,1,2,3 | 1 | 2 | 1 | 0 |
| 0,1,2,3,4 | 1 | 2 | 2 | 0 |
| 0,1,2,3,4,5 | 1 | 2 | 2 | 1 |
| 0,1,2,3,4,5,6 | 1 | 2 | 3 | 1 |

### **Сортировка вставками**

В массиве выделяем массив из первого элемента. Будем считать, что он отсортирован. На каждом следующем шаге увеличиваем длину рассматриваемого массива на 1 и пока элемент j меньше элемента с индексом j-1 будем менять их местами.

Таблица 5. Пример сортировки вставками

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | ***-1*** | 4 | 9 | -3 | 8 | -8 | 0 |
| 2 | ***-1*** | ***4*** | 9 | -3 | 8 | -8 | 0 |
| 3 | ***-1*** | ***4*** | ***9*** | -3 | 8 | -8 | 0 |
| *-1* | *4* | -3 | *9* | 8 | -8 | 0 |
| *-1* | -3 | *4* | *9* | 8 | -8 | 0 |
| *-3* | -1 | *4* | *9* | 8 | -8 | 0 |
| 4 | ***-3*** | ***-1*** | ***4*** | ***9*** | 8 | -8 | 0 |
| *-3* | *-1* | *4* | 8 | *9* | -8 | 0 |
| 5 | ***-3*** | ***-1*** | ***4*** | ***8*** | ***9*** | -8 | 0 |
| *-3* | *-1* | *4* | *8* | *-8* | 9 | 0 |

### **Быстрая сортировка (Хоара)**

В массиве выберем ведущий элемент (можно выбрать среднее арифметическое значений элементов, центральный элемент или же любой другой). Слева от него кладем все те элементы массива, которые меньше ведущего элемента, все большие элементы кладем в правую часть. Далее выбираем ведущий элемент в правой и левой частях, и продолжаем делать до тех пор, пока размер одной из частей не станет меньше.

Таблица 6. Пример сортировки Хоара

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | -1 | 4 | 9 | *-3* | 8 | -8 | 0 |
|  | -1 | -8 | 0 | *-3* | 4 | 9 | 8 |
| 2 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

### **Сортировка слиянием**

Идея сортировки слияния заключается в том, что сначала мы рекурсивно делим исходный массив на массивы, состоящие из одного элемента, а затем на каждом новом шаге, упорядочивая массивы, рекурсивно попарно сливаем их, приводя в виду исходного массива.

Таблица 7. Пример сортировки слиянием

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Индекс  Шаг | Массив | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | *-1* | *4* | 9 | -3 | 8 | -8 | 0 |
| -1 | 4 | *9* | *-3* | 8 | -8 | 0 |
| -1 | 4 | -3 | 9 | *8* | *-8* | 0 |
| -1 | 4 | -3 | 9 | -8 | 8 | *0* |
| 2 | *-1* | *4* | *-3* | *9* | -8 | 8 | 0 |
| -3 | -1 | 4 | 9 | *-8* | *8* | *0* |

# **Описание функций**

void bubble(int \*a, ULONGLONG \*size, int n)

Назначение: сортировка пузырьком для массива указанного размера.

Входные параметры: size – массив размеров файлов, которые нужно упорядочить; a – количество упорядочиваемых файлов; n – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void choose(int \*a, ULONGLONG \*size, int n)

Назначение: сортировка выбором для указанного массива указанного размера.

Входные параметры: size – массив размеров файлов, которые нужно упорядочить; a – количество упорядочиваемых файлов; n – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void count\_sort(int \*a, ULONGLONG \*size, int n)

Назначение: сортировка подсчетом для массива указанного размера.

Входные параметры: size – массив размеров файлов, которые нужно упорядочить; a – количество упорядочиваемых файлов; n – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

void insert(int \*a, ULONGLONG \*size, int n)

Назначение: сортировка простыми вставками для массива указанного размера.

Входные параметры: size – массив размеров файлов, которые нужно упорядочить; a – количество упорядочиваемых файлов; n – массив индексов файлов.

Выходные параметры: указатель на массив индексов файлов.

int\* quick\_sort(int \*a, ULONGLONG \*size, int n1, int n2)

Назначение: быстрая сортировка для массива указанного размера.

Входные параметры: size – массив размеров файлов, которые нужно упорядочить; a – количество упорядочиваемых файлов; n1 – первый индекс; n2 – последний индекс.

Выходные параметры: указатель на массив индексов файлов.

void merge\_sort(int\* a, ULONGLONG\* size, int l, int m, int r)

Назначение: вспомогательная функция для сортировки слиянием. Соединяет подмассивы.

Входные параметры: a - массив индексов; size - массив размеров файлов; l – индекс первого элемента; m – индекс среднего элемента; r – индекс последнего элемента.

void mergesort(int\* a, ULONGLONG\* size, int l, int r)

Назначение: сортировка слиянием для массива.

Входные параметры: a - массив индексов; size - массив размеров файлов; l – индекс первого элемента; r – индекс последнего элемента.

Выходные параметры: указатель на массив индексов файлов.

void print\_menu()

Назначение: вывод командного меню.

Входные параметры: -

Выходные параметры: -

void quick\_split(int \*a, ULONGLONG \*size, int \*i, int \*j, ULONGLONG p)

Назначение

Функция необходимая для быстрой сортировки. Все элементы, меньшие опорного, перемещаются влево от него, большие – вправо.

Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

\*i – адрес первого индекса

\*j – адрес второго индекса

p – опорный элемент

Выходные параметры

Отсутствуют

void quick\_sort(int \*a, ULONGLONG \*size, int n1, int n2)

Назначение

Сортировка массива размеров файлов быстрой сортировкой с изменением массива индексов

Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

Выходные параметры

Отсутствуют

int ListDirectoryContents(const wchar\_t \*sDir, ULONGLONG \*\*sizes, wchar\_t \*\*\*names)

Назначение

Определение корректности введенных данных, определение количества файлов в директории, формирование и заполнение массивов названий файлов и их размеров

Входные параметры

\*sDir – путь до каталога

\*\*sizes – массив размеров файлов

\*\*\*names – адрес массива названий

Выходные параметры

count – количество файлов в каталоге

# **Заключение**

Создан прототип файлового менеджера Прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по возрастанию. Реализованы 6 функций сортировок: сортировка выбором, сортировка вставками, сортировка пузырьком, сортировка подсчетом, сортировка слиянием и сортировка Хоара.

Теоретическая оценка сложности каждого метода сортировки

Таблица 8. Теоретическая оценка сложности каждого метода сортировки

|  |  |  |
| --- | --- | --- |
| Метод сортировки | Затраченное время | Память |
| Сортировка выбором | O(n2) | O(1) |
| Сортировка вставками | O(n2) | O(1) |
| Пузырьковая сортировка | O(n2) | O(1) |
| Сортировка подсчетом | O(n) | (max(n) – min(n)) + 1 |
| Сортировка слиянием | O(n·log2 n) | O(1) |
| Сортировка Хоара | O(n·log2 n) | O(n) |

# **Литература**

1. Сайт StudFiles (<https://studfiles.net/preview/5707298/page:9/>)
2. <https://tproger.ru/translations/sorting-for-beginners/>