Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

## Отчет по лабораторной работе

## Прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоенных по возрастанию.

**Выполнил**:студент группы 381806-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Щекотилова Ю.А.

**Проверил**:

к.т.н., доцент кф. МАТ. ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кустикова В.Д.

Нижний Новгород

2018

# Введение

Для решения многих задач удобно сначала упорядочить данные по определенному признаку, так можно ускорить поиск некоторого объекта. Например, в преферансе игроки раскладывают карты по мастям и по значению. Так легче определить, каких карт не хватает. Или возьмем любой энциклопедический словарь - статьи в нем упорядочены в алфавитном порядке.

Перегруппирование заданного множества объектов в определенном порядке называют *сортировкой*.

Почему сортировке уделяется большое внимание? Вы это поймете, прочитав цитаты двух великих людей.

"Даже если бы сортировка была почти бесполезна, нашлась бы масса причин заняться ею! Изобретательные методы сортировки говорят о том, что она и сама по себе интересна как объект исследования." /Д. Кнут/

"Создается впечатление, что можно построить целый курс программирования, выбирая примеры только из задач сортировки." /Н. Вирт/

Отличительной особенностью сортировки является то обстоятельство, что эффективность алгоритмов, реализующих ее, прямо пропорциональна сложности понимания этого алгоритма. Другими словами, чем легче для понимания метод сортировки массива, тем ниже его эффективность.

# Постановка задачи

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

* Сортировка выбором
* Сортировка вставками
* Сортировка пузырьком
* Сортировка подсчетом
* Сортировка слиянием
* Сортировка Хоара (бытрая 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 программы на экран вы выводится текст, позволяющий написать путь к папке, в которой нужно сделать сортировку . (Рис. 1)



Рис. 1

Вводим ссылку на каталог. Допустим, что нам надо отстортировать файлы в каталоге D:\ Files. Далее программа предлагает выбрать сортировку или выйти из программы. (Рис. 2)

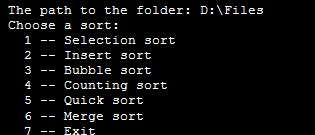


Рис. 2

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

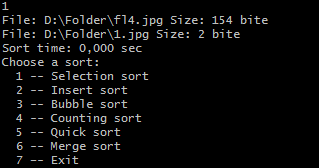


Рис. 3

Далее можно выбрать другой метод сортировки, или же выйти из программы.

# 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

### 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 1)

Таблица 1. 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 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 2)

Таблица 2. 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 |
| *-3* | *-1* | *4* | *-8* | *8* | 9 | 0 |
| *-3* | *-1* | *-8* | *4* | *8* | 9 | 0 |
| *-3* | *-8* | *-1* | *4* | *8* | 9 | 0 |
| *-8* | *-3* | *-1* | *4* | *8* | 9 | 0 |
| 6 | ***-8*** | ***-3*** | ***-1*** | ***4*** | ***8*** | ***9*** | 0 |
| *-8* | *-3* | *-1* | *4* | *8* | 0 | *9* |
| *-8* | *-3* | *-1* | *4* | 0 | *8* | *9* |
| *-8* | *-3* | *-1* | 0 | *4* | *8* | *9* |
| Отсортированный массив | ***-8*** | ***-3*** | ***-1*** | **0** | ***4*** | ***8*** | ***9*** |

### Сортировка пузырьком

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 3)

Таблица 3. Пример сортировки пузырьком

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | *9* | *-3* | 8 | -8 | 0 |
| -1 | 4 | -3 | *9* | *8* | -8 | 0 |
| -1 | 4 | -3 | 8 | *9* | *-8* | 0 |
| -1 | 4 | -3 | 8 | -8 | *9* | *0* |
| 2 | *-1* | *4* | -3 | 8 | -8 | 0 | 9 |
| -1 | *4* | *-3* | 8 | -8 | 0 | 9 |
| -1 | -3 | *4* | *8* | -8 | 0 | 9 |
| -1 | -3 | 4 | *8* | *-8* | 0 | 9 |
| -1 | -3 | 4 | -8 | *8* | *0* | 9 |
| 3 | *-1* | *4* | -3 | -8 | 0 | 8 | 9 |
| -1 | *4* | *-3* | -8 | 0 | 8 | 9 |
| -1 | -3 | *4* | *-8* | 0 | 8 | 9 |
| -1 | -3 | -8 | *4* | *0* | 8 | 9 |
| 4 | *-1* | *-3* | -8 | 0 | 4 | 8 | 9 |
| -3 | *-1* | *-8* | 0 | 4 | 8 | 9 |
| -3 | -8 | *-1* | *0* | 4 | 8 | 9 |
| 5 | *-3* | *-8* | -1 | 0 | 4 | 8 | 9 |
| -8 | *-3* | *-1* | 0 | 4 | 8 | 9 |
| 6 | -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 4)

Таблица 4. 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 5. 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 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 6)

Таблица 6. 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* |
| 3 | *-3* | *-1* | *4* | *9* | *-8* | *0* | *8* |
| Отсортированный массив | -8 | -3 | -1 | 0 | 4 | 8 | 9 |

### Сортировка Хоара (быстрая сортировка)

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 7)

Таблица 7. 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 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## Описание функций

### Функция void SelectionSort(int \*a, ULONGLONG \*size, int n)

Функция реализует сортировку массива размеров файлов сортировкой выбором с изменением массива индексов. Принимает на вход массив индексов (\*a), массив размеров файлов (\*size) и количество файлов в каталоге (n).

### Функция void InsertSort(int \*a, ULONGLONG \*size, int n)

Функция реализует сортировку массива размеров сортировкой вставками с изменением массива индексов. В качестве входных данных принимает массив индексов (\*a), массив размеров файлов (\*size) и количество файлов в каталоге (n).

### Фунцкия void BubbleSort(int \*a, ULONGLONG \*size, int n)

Функция реализует сортировку массива размеров пузырьковой сортировкой с изменением массива индексов. Входными данными являются: массив индексов (\*a), массив размеров файлов (\*size) и количество файлов в каталоге (n).

### Функция void CountingSort(int \*a, ULONGLONG \*size, int n)

Функция реализует сортировку массива размеров сортировкой подчетом с изменением массива индексов. На вход принимает массив индексов (\*a), массив размеров файлов(\*size) и количество файлов в каталоге (n).

### Функция void QuickSplit(int \*a, ULONGLONG \*size, int \*i, int \*j, ULONGLONG p)

Вспомогательная функция для сортировки Хоара (быстрой сортировки). Принимает на вход массив размеров файлов (\*size), адреса первого и второго индексов (\*i и \*j соответственно) и ведущий элемент p.

### Функция void QuickSort(int \*a, ULONGLONG \*size, int n1, int n2)

Функция, сортирующая массив размеров файлов сортировкой Хоара (быстрой сортировкой) с изменением массива индексов. В качестве входных данных принимает массив индексов (\*a), массив размеров файлов (\*size) и индексы первого и второго элементов (n1 и n2соответственно).

### Функция void Merge(int \*a, ULONGLONG \*size, int l, int m, int r)

Вспомогательная функция для сортировки слиянием, обеспечивающая соединение массивов с их упорядочиванием. Входными данными являются массив индексов (\*a), массив размеров файлов (\*size) и индексы первого, среднего и последнего элементов (l, m и r соответственно).

### Функция void MergeSort(int \*a, ULONGLONG \*size, int l, int r)

Функция реализует сортировку массива размеров файлов сортировкой слиянием с изменением массива индексов. В качестве входных данных используются массив индексов (\*a), массив размеров файлов (\*size) и индексы первого и второго элементов (l и r соответственно).

### Функция int FileDirectory(const wchar\_t \*sDir, ULONGLONG \*\*ssize, wchar\_t \*\*\*nname)

Функция проверяет правильность введенных данных, определяет количество файлов в директории, формирует массивов названий файлов и их размеров, а затем заполняет его. На вход принимает путь до каталога (\*sDir), массив размеров файлов (\*\*ssize) и адрес массива названий (\*\*\*nname). На выходе получаем количество файлов в каталоге (сount).

### Функция void input(wchar\_t \*\*strin)

Эта функция позволяет вводить путь к заданному каталогу. В качестве входных данных используется указатель на массив (\*\*strin). В этот указатель пользователем будет записано некоторое значение.

### Функция void CommandMenu()

Функция, выводящая на экран командное меню.

### Функция void Output(int \*index, wchar\_t \*\*nname, ULONGLONG \*ssize, int count)

Функция выводит на экран отсортированный список имен файлов с указанием их размера и времени сортировки. На вход принимает массив индексов отсортированного массива размеров файлов (\*index), массив названий файлов (\*\*nname), массив размеров файлов (\*ssize) и количество файлов (сount).

# 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)

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. Подбельский В. В., Фомин С.С. Курс программирования на языке Си. Учебник ДМК-Пресс -2012 104 с.

2. Сайт StudFiles (https://studfiles.net/preview/5707298/page:9/)

# Содержание

[Введение 2](#_Toc533581294)

[1 Постановка задачи 3](#_Toc533581295)

[2 Руководство пользователя 4](#_Toc533581296)

[3 Руководство программиста 5](#_Toc533581297)

[3.1 Структура программы 5](#_Toc533581298)

[3.2 Описание алгоритма 5](#_Toc533581299)

[3.2.1 Сортировка выбором 5](#_Toc533581300)

[3.2.2 Сортировка вставками 5](#_Toc533581301)

[3.2.3 Сортировка пузырьком 6](#_Toc533581302)

[3.2.4 Сортировка подсчетом 6](#_Toc533581303)

[3.2.5 Сортировка слиянием 7](#_Toc533581304)

[3.2.6 Сортировка Хоара (быстрая сортировка) 7](#_Toc533581305)

[3.3 Описание функций 8](#_Toc533581306)

[3.3.1 Функция void SelectionSort(int \*a, ULONGLONG \*size, int n) 8](#_Toc533581307)

[3.3.2 Функция void InsertSort(int \*a, ULONGLONG \*size, int n) 8](#_Toc533581308)

[3.3.3 Фунцкия void BubbleSort(int \*a, ULONGLONG \*size, int n) 8](#_Toc533581309)

[3.3.4 Функция void CountingSort(int \*a, ULONGLONG \*size, int n) 8](#_Toc533581310)

[3.3.5 Функция void QuickSplit(int \*a, ULONGLONG \*size, int \*i, int \*j, ULONGLONG p) 8](#_Toc533581311)

[3.3.6 Функция void QuickSort(int \*a,ULONGLONG \*size,int n1,int n2)….8](#_Toc533581312)

[3.3.7 Функция void Merge(int \*a, ULONGLONG \*size, int l, int m, int r) 9](#_Toc533581313)

[3.3.8 Функция void MergeSort(int \*a, ULONGLONG \*size, int l, int r) 9](#_Toc533581314)

[3.3.9 Функция int FileDirectory(const wchar\_t \*sDir, ULONGLONG \*\*ssize, wchar\_t \*\*\*nname) 9](#_Toc533581315)

[3.3.10 Функция void input(wchar\_t \*\*strin) 9](#_Toc533581316)

[3.3.11 Функция void CommandMenu() 9](#_Toc533581317)

[3.3.12 Функция void Output(int \*index, wchar\_t \*\*nname, ULONGLONG \*ssize, int count) 9](#_Toc533581318)

[Заключение 10](#_Toc533581319)

[Литература 11](#_Toc533581320)