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 размера**

**Выполнил:** студент группы 381806-1

Рустамов Азер Заур оглы

**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 6](#_Toc533366045)

[**Структура программы** 6](#_Toc533366046)

[**Описание алгоритма** 6](#_Toc533366047)

[Сортировка выбором – choose\_sort 6](#_Toc533366048)

[Сортировка вставками – insert\_sort 6](#_Toc533366049)

[Пузырьковая сортировка – bubble\_sort 7](#_Toc533366050)

[Сортировка подсчетом – counting\_sort 8](#_Toc533366051)

[Быстрая сортировка – quick\_sort 9](#_Toc533366052)

[**Описание функций** 10](#_Toc533366053)

[Подключение к директории с последующим формированием массивов названий файлов и их размеров 10](#_Toc533366054)

[Вывод меню команд 10](#_Toc533366055)

[Вывод отсортированного списка названий файлов с указанием размера и времени сортировки 10](#_Toc533366056)

[Сортировка выбором 10](#_Toc533366057)

[Сортировка вставками 11](#_Toc533366058)

[Пузырьковая сортировка 11](#_Toc533366059)

[Сортировка подсчетом 11](#_Toc533366060)

[Разбиение для быстрой сортировки 11](#_Toc533366061)

[Быстрая сортировка 12](#_Toc533366062)

[Функция слияния 12](#_Toc533366063)

[Сортировка слиянием 12](#_Toc533366064)

[Заключение 13](#_Toc533366065)

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

[4. Литература 15](#_Toc533366067)

# **Введение**

# **Постановка задачи**

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
* 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 [Рисунок 1]

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].

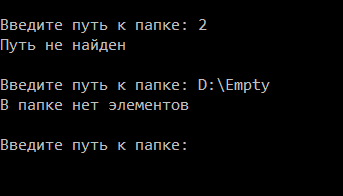


Рисунок 2. Пример неверного пути до директории и пути в пустую папку

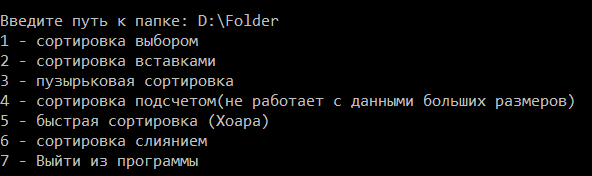
В случае корректно введенных данных на экране появится меню команд. Для выбора одной из команд необходимо ввести ее номер [Рисунок 3]. 

Рисунок 3. Меню команд

В случае введения номера команды от 1 до 6 будет выполнена выбранная сортировка и на экран будут выведены отсортированные названия файлов и их размеры в байтах, время сортировки и меню команд [Рисунок 4].

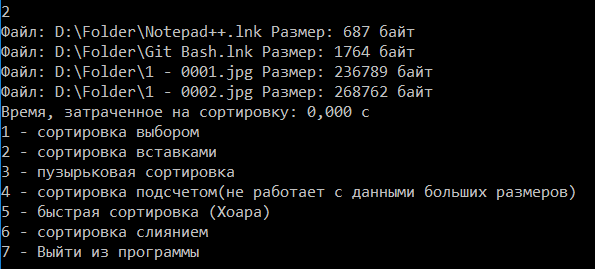


Рисунок 4. Пример выводимых данных

Для завершения работы с программой необходимо выбрать 7 команду меню команд – Exit [Рисунок 3].

# **Руководство программиста**

## **Структура программы**

Программа состоит из 1 файла *main.c,* который содержит функцию *main*, функции сортировок, функцию вывода меню команд, функцию ввода директории, функцию вывода отсортированных названий файлов, их размеров в байтах и времени сортировки.

## **Описание алгоритма**

### *Сортировка выбором – choose\_sort*

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

Таблица 1. Пример сортировки выбором

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

### *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 | Нет |
| 2 | **5** | **9** | 8 | 2 | 6 | 3 | 1 | 4 | 2 | Да |
| **5** | 8 | **9** | 2 | 6 | 3 | 1 | 4 | 1 | Нет |
| 3 | **5** | **8** | **9** | 2 | 6 | 3 | 1 | 4 | 3 | Да |
| **5** | **8** | 2 | **9** | 6 | 3 | 1 | 4 | 2 | Да |
| **5** | 2 | **8** | **9** | 6 | 3 | 1 | 4 | 1 | Да |
| 2 | **5** | **8** | **9** | 6 | 3 | 1 | 4 | 0 | Нет |
| 4 | **2** | **5** | **8** | **9** | 6 | 3 | 1 | 4 | 4 | Да |
| **2** | **5** | **8** | 6 | **9** | 3 | 1 | 4 | 3 | Да |
| **2** | **5** | 6 | **8** | **9** | 3 | 1 | 4 | 2 | Нет |
| 5 | **2** | **5** | **6** | **8** | **9** | 3 | 1 | 4 | 5 | Да |
| **2** | **5** | **6** | **8** | 3 | **9** | 1 | 4 | 4 | Да |
| **2** | **5** | **6** | 3 | **8** | **9** | 1 | 4 | 3 | да |
| **2** | **5** | 3 | **6** | **8** | **9** | 1 | 4 | 2 | Да |
| **2** | 3 | **5** | **6** | **8** | **9** | 1 | 4 | 1 | Нет |
| 6 | **2** | **3** | **5** | **6** | **8** | **9** | 1 | 4 | 6 | Да |
| **2** | **3** | **5** | **6** | **8** | 1 | **9** | 4 | 5 | Да |
| **2** | **3** | **5** | **6** | 1 | **8** | **9** | 4 | 4 | Да |
| **2** | **3** | **5** | 1 | **6** | **8** | **9** | 4 | 3 | Да |
| **2** | **3** | 1 | **5** | **6** | **8** | **9** | 4 | 2 | Да |
| **2** | 1 | **3** | **5** | **6** | **8** | **9** | 4 | 1 | Да |
| 1 | **2** | **3** | **5** | **6** | **8** | **9** | 4 | 0 | Нет |
| 7 | **1** | **2** | **3** | **5** | **6** | **8** | **9** | 4 | 7 | Да |
| **1** | **2** | **3** | **5** | **6** | **8** | 4 | **9** | 6 | Да |
| **1** | **2** | **3** | **5** | **6** | 4 | **8** | **9** | 5 | Да |
| **1** | **2** | **3** | **5** | 4 | **6** | **8** | **9** | 4 | Да |
| **1** | **2** | **3** | 4 | **5** | **6** | **8** | **9** | 3 | Нет |
| - | **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 | Да |
| 6 | 8 | 4 | 2 | 9 | 5 | 3 | 7 | 1, 2 | Да |
| 6 | 4 | 8 | 2 | 9 | 5 | 3 | 7 | 2, 3 | Да |
| 6 | 4 | 2 | 8 | 9 | 5 | 3 | 7 | 3, 4 | Нет |
| 6 | 4 | 2 | 8 | 9 | 5 | 3 | 7 | 4, 5 | Да |
| 6 | 4 | 2 | 8 | 5 | 9 | 3 | 7 | 5, 6 | Да |
| 6 | 4 | 2 | 8 | 5 | 3 | 9 | 7 | 6, 7 | Да |
| 2 | 6 | 4 | 2 | 8 | 5 | 3 | 7 | **9** | 0, 1 | Да |
| 4 | 6 | 2 | 8 | 5 | 3 | 7 | **9** | 1, 2 | Да |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | **9** | 2, 3 | Нет |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | **9** | 3, 4 | Да |
| 4 | 2 | 6 | 5 | 8 | 3 | 7 | **9** | 4, 5 | Да |
| 4 | 2 | 6 | 5 | 3 | 8 | 7 | **9** | 5, 6 | Да |
| 3 | 4 | 2 | 6 | 5 | 3 | 7 | **8** | **9** | 0, 1 | Да |
| 2 | 4 | 6 | 5 | 3 | 7 | **8** | **9** | 1, 2 | Нет |
| 2 | 4 | 6 | 5 | 3 | 7 | **8** | **9** | 2, 3 | Да |
| 2 | 4 | 5 | 6 | 3 | 7 | **8** | **9** | 3, 4 | Да |
| 2 | 4 | 5 | 3 | 6 | 7 | **8** | **9** | 4, 5 | Нет |
| 4 | 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 0, 1 | Нет |
| 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 1, 2 | Нет |
| 2 | 4 | 5 | 3 | 6 | **7** | **8** | **9** | 2, 3 | Да |
| 2 | 4 | 3 | 5 | 6 | **7** | **8** | **9** | 3, 4 | Нет |
| 5 | 2 | 4 | 3 | 5 | **6** | **7** | **8** | **9** | 0, 1 | Нет |
| 2 | 4 | 3 | 5 | **6** | **7** | **8** | **9** | 1, 2 | Да |
| 2 | 3 | 4 | 5 | **6** | **7** | **8** | **9** | 2, 3 | Нет |
| 6 | 2 | 3 | 4 | **5** | **6** | **7** | **8** | **9** | 0, 1 | Нет |
| 2 | 3 | 4 | **5** | **6** | **7** | **8** | **9** | 1, 2 | Нет |
| 7 | 2 | 3 | **4** | **5** | **6** | **7** | **8** | **9** | 0, 1 | Нет |
| - | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | - | - |

### *Сортировка подсчетом – counting\_sort*

* Создаем дополнительный массив и заполняем его 0
* Проходим по всему исходному массиву и прибавляем 1 к элементу нового массива, индекс которого равен значению элемента исходного массива
* Заново заполняем 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** |

* + 1. 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**

### *Подключение к директории с последующим формированием массивов названий файлов и их размеров*

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

Вывод меню командvoid input(wchar\_t \*\*strin)

* Назначение

Ввод пути к заданному каталогу

* Входные параметры

\*\*strin – указатель на массив, в который будет записано введенное пользователем значение.

* Выходные параметры

### *Вывод меню команд*

void menu()

* Назначение

Вывод на экран меню команд

* Входные параметры

Отсутствуют

* Выходные параметры

Отсутствуют

### *Вывод отсортированного списка названий файлов с указанием размера и времени сортировки*

void output(int \*Idxes, wchar\_t \*\*names, ULONGLONG \*sizes, int count)

* Назначение

Вывод на экран отсортированного списка имен файлов с указанием размера и времени сортировки

* Входные параметры

\*Idxes – массив индексов отсортированного массива размеров файлов

\*\*names – массив названий файлов

\*sizes – массив размеров файлов

filesCount – количество файлов

* Выходные параметры

Отсутствуют

### *Сортировка выбором*

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

* Назначение

Сортировка массива размеров файлов сортировкой выбором с изменением массива индексов

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

n – количество файлов в каталоге

* Выходные параметры

Отсутствуют

### *Сортировка вставками*

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

* Назначение

Сортировка массива размеров файлов сортировкой вставками с изменением массива индексов

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

n – количество файлов в каталоге

* Выходные параметры

Отсутствуют

### *Пузырьковая сортировка*

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

* Назначение

Сортировка массива размеров файлов пузырьковой сортировкой с изменением массива индексов

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

n – количество файлов в каталоге

* Выходные параметры

Отсутствуют

### *Сортировка подсчетом*

int counting\_sort(int \*a, ULONGLONG \*size, int n)

* Назначение

Сортировка массива размеров файлов сортировкой подсчетом с изменением массива индексов

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

n – количество файлов в каталоге

* Выходные параметры

Отсутствуют

### *Разбиение для быстрой сортировки*

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

### *Быстрая сортировка*

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

n1 – индекс первого элемента

n2 – индекс последнего элемента

* Выходные параметры

Отсутствуют

### *Функция слияния*

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

* Назначение

Функция необходимая для сортировки слиянием. Соединяет подмассивы, упорядочивая их.

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

l – индекс первого элемента

m – индекс среднего

r – индекс последнего элемента

* Выходные параметры

Отсутствуют

### *Сортировка слиянием*

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

* Назначение

Сортировка массива размеров файлов сортировкой слиянием с изменением массива индексов

* Входные параметры

\*a – массив индексов массива размеров файлов

\*size – массив размеров файлов

l – индекс первого элемента

r – индекс последнего элемента

* 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.

1. Сложность методов сортировки

|  |  |  |
| --- | --- | --- |
| 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**