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

Касьянычев Михаил Петрович

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

Кустикова Валентина Дмитриевна

Нижний Новгород

2018

Оглавление

[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 6](#_Toc533366048)

[Сортировка вставками 6](#_Toc533366049)

[Пузырьковая сортировка 7](#_Toc533366050)

[Сортировка подсчетом 8](#_Toc533366051)

[Быстрая сортировка 9](#_Toc533366052)

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

[Меню команд 10](#_Toc533366055)

[Вывод списка файлов 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**

**Задача:** разработать и реализовать прототип файлового менеджера с функцией показа файлов в заданном каталоге, упорядоченных по размеру.

**Входные данные:** путь к каталогу, к которому необходимо обратиться; способ сортировки.

**Выходные данные:** отсортированный по размеру список файлов в заданном каталоге.

# **Руководство пользователя**

После запуска программы 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].

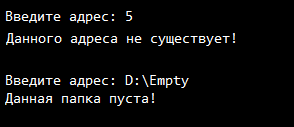


Рисунок 2. Пример неверного пути до директории и пути в пустую папку

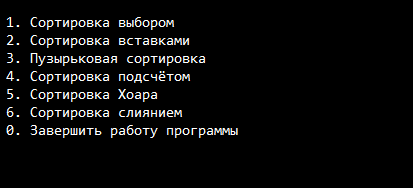
В случае корректно введенных данных на экране появится меню команд. Для выбора одной из команд необходимо ввести ее номер [Рисунок 3]. 

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].

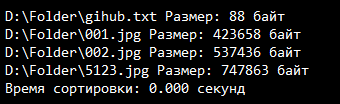


Рисунок 4. Пример выводимых данных

Для завершения работы с программой необходимо выбрать 0 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.*

1. Массив делится на отсортированную и неотсортированную часть
2. Ищется минимальный элемент в неотсортированной части.
3. Этот элемент меняется с первым из неотсортированной части, после этого отсортированная часть считается увеличившейся на этот элемент.
4. Описанные действия повторяются на один раз меньше, чем элементов в массиве.

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 итерация | 2 | 4 | 3 | 5 | 1 |
| 1 итерация | 1 | 4 | 5 | 3 | 2 |
| 2 итерация | 1 | 2 | 5 | 3 | 4 |
| 3 итерация | 1 | 2 | 3 | 5 | 4 |
| 4 итерация | 1 | 2 | 3 | 4 | 5 |

### *Сортировка вставками*

Сначала первый элемент массива считается полностью отсортированным.

Далее из оставшихся элементов берется первый и сравнивается с последним отсортированным. Если отсортированный элемент больше – они меняются местами.

Эту операцию повторяем до тех пор, пока не встретится число меньше или пока элемент не достигнет левого края.

Действия выше повторяем до тех пор, пока массив не будет полностью отсортирован.

Пример сортировки выбором рассмотрен в таблице 2 (полужирным выделен отсортированный фрагмент массива, курсивом – индексы).

1. Пример сортировки простыми 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*

1. Проходим массив с первого элемента до n-ого (где n – количество элементов в массиве)
2. Сравниваем попарно стоящие рядом элементы (1 и 2, 2 и 3, ..., n–1 и n)
3. Если первый элемент пары больше второго, меняем элементы местами
4. 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*

1. Создаем дополнительный массив и заполняем его 0
2. Проходим по всему исходному массиву и прибавляем 1 к элементу нового массива, индекс которого равен значению элемента исходного массива
3. 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*

1. Выбираем ведущий элемент
2. Все меньшие элементы перемещаем влево от него, большие – вправо
3. 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*

1. Рекурсивно разбиваем исходный массив на подмассивы из одного элемента
2. Рекурсивно попарно сливаем подмассивы
3. 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 \*\*dir)

* Назначение

Ввод пути к заданному каталогу

* Входные параметры

\*\*dir – 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(wchar\_t \*\*name, ULONGLONG \*size, int \*idx, int k)

* Назначение

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

* Входные параметры

\*idxe – массив индексов отсортированного массива размеров файлов

\*\*name – массив названий файлов

\*size – массив размеров файлов

k – количество файлов

* Выходные параметры

Отсутствуют

### *Сортировка выбором*

void choosesort(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 insertsort(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 bubblesort(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 countingsort(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 quicksplit(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 quicksort(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 mergesort(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**

В результате выполнения практической работы, был разработан прототип файлового менеджера, с возможностью отображения списка файлов, отсортированных с помощью одного из шести алгоритмов, в заданной папке.

В программе реализованы следующий сортировки: выбором, вставками, пузырьком, подсчётом, быстрая и слиянием. Чья сравнительная характеристика сложности представлена ниже.

|  |  |
| --- | --- |
| Сортировка | Сложность |
| Выбором | O(n2) |
| Вставками | O(n2) |
| Пузырьком | O(n2) |
| Подсчётом | O(n) |
| Быстрая | O(n⋅log n) |
| Слиянием | O(n⋅log n) |

# **Литература**