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 группы 381806-3

Лудина Д.А.

**Проверил:**

Доцент кафедры МОСТ ИИТММ

Кустикова В.Д.

Нижний Новгород

1. г.

**Содержание**

[Введение 3](#_Toc533556253)

[1 Постановка задач 4](#_Toc533556254)

[2 Руководство пользователя 5](#_Toc533556255)

[3 Руководство программиста 7](#_Toc533556256)

[3.1 Структура программы 7](#_Toc533556257)

[3.2 Описание алгоритмов 7](#_Toc533556258)

[3.2.1 Сортировка выбором 7](#_Toc533556259)

[3.2.2 Сортировка вставками 7](#_Toc533556260)

[3.2.3 Сортировка пузырьком 8](#_Toc533556261)

[3.2.4 Сортировка подсчетом 9](#_Toc533556262)

[3.2.5 Сортировка слиянием 10](#_Toc533556263)

[3.2.6 Быстрая сортировка(Хоара) 10](#_Toc533556264)

[3.3 Описание функций 11](#_Toc533556265)

[Заключение 13](#_Toc533556266)

[Литература 14](#_Toc533556267)

# Введение

На сегодняшний день одной из важнейших операций, применяемых каждым пользователем ПК, является сортировка массивов. В условиях повышения сложности решаемых в программировании задач, в которых необходимо работать с большими объёмами информации, куда выгоднее сначала упорядочивать используемый массив данных в нужном порядке, что значительно снизит затраты во времени при поиске какого-либо элемента из этого массива. Упорядочивание данных можно проводить по разным критериям в зависимости от того, с чем собирается работать пользователь или разработчик: упорядочивание по возрастанию/убыванию; в алфавитном порядке; по дате и времени и т.д.

В данной работе рассматривается прототип файлового менеджера, который с помощью разных сортировок, может отсортировать файлы по возрастанию/убыванию размер, находящиеся в папке, путь до которой указывает пользователь.

# Постановка задач

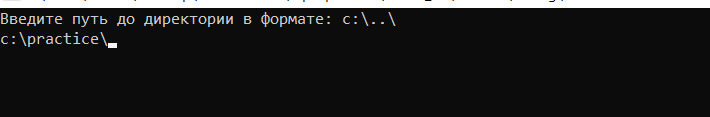
В данной работе необходимо реализовать программу файлового менеджера, которая отсортирует файлы по возрастанию/убыванию размера. Используя при этом различные методы сортировки, такие как: «пузырьком», выбором, вставками, слиянием, Хоара (так же известная, как «Быстрая»), подсчетом

Входные данные: путь до директории, в которой необходимо отсортировать содержимое, метод сортировки, выбор сортировки по возрастанию или убыванию

Выходные данные: отсортированный список имен файлов с указанием размера, время сортировки

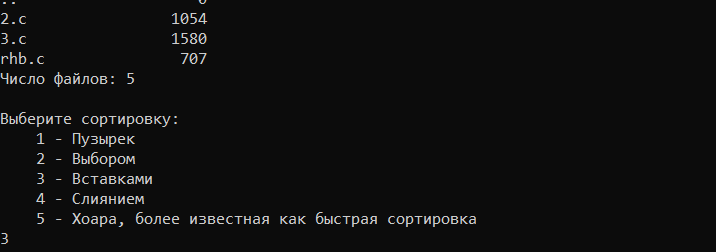
# Руководство пользователя

После запуска программы пользователю предлагается ввести путь по директории. Например, требуется отсортировать файл в каталоге «С:\practice\». В этом случае вводим в поле данный путь и нажимаем Enter.().



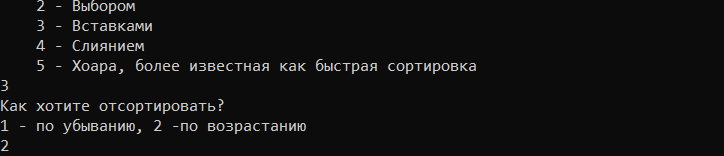
1. Ввод пути

На экране появиться список файлов с их размерами, общее количество файлов. Программа предложит выбрать вид сортировки. Нажмите цифру, соответствующую сортировке, которую вы хотите использовать, нажмите Enter.()



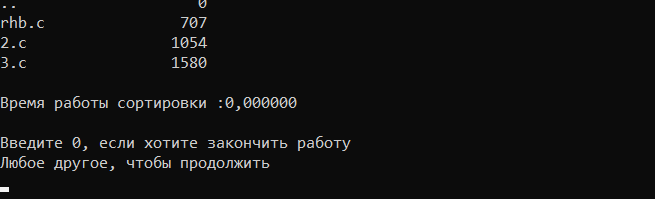
1. Вывод списка файлов

Далее программа предложит выбрать, как сортировать по убыванию или возрастанию.()



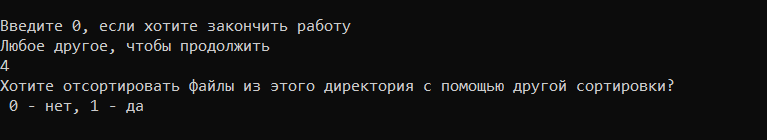
1. Выбор сортировки

Выведутся отсортированные файлы по размеру. Вам предложат продолжить или закончить работу.()



1. Условия выхода из программы

При введении любой другой цифры программа спрашивает из какой директории мы хотим отсортировать список файлов.()



1. Предложение о смене директории

Далее программа действует по такому же алгоритму.

# Руководство программиста

## Структура программы

Данная программа состоит из одного файла main.c, в котором описан весь ее код.

## Описание алгоритмов

### Сортировка выбором

Идея метода состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке.

Будем строить готовую последовательность, начиная с левого конца массива. Алгоритм состоит из n последовательных шагов, начиная от нулевого и заканчивая (n-1)-м.

На i-м шаге выбираем наименьший из элементов a[i] ... a[n] и меняем его местами с a[i]. Последовательность шагов при n=5 изображена на рисунке ниже.

1. Пример сортировки выбором

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

### Сортировка вставками

Суть алгоритма заключается в том, что на каждом шаге выбирается элемент из неотсортированной части массива, после чего он вставляется на нужную позицию в уже отсортированной части массива. Алгоритм будет работать до тех пор, пока не закончится массив. Выбор элемента из массива можно осуществлять произвольно, но удобнее брать их по порядку.

1. Пример сортировки вставками

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

### Сортировка пузырьком

Идея метода этой сортировки состоит в проходе по массиву снизу вверх. При этом сравниваются соседние элементы. Если они расположены неправильно, то их меняем местами. После нулевого прохода по массиву (при сортировке по возрастанию) самый «легкий» элемент оказывается «вверху» - отсюда аналогия с пузырьком. Следующий проход делается до второго элемента сверху, и второй по величине элемент будет поставлен на свое место. Далее делаются проходы по все уменьшающейся части массива до тех пор, пока не останется последовательность из одного элемента.

1. Пример сортировки пузырьком

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

### Сортировка подсчетом

При сортировке методом подсчета упорядоченная последовательность элементов создается на свободном участке памяти. Идея метода заключается в следующем: в отсортированной последовательности, элемент, занимающий позицию с номером К+1, превышает ровно К элементов, поэтому в процессе сортировки методом подсчета на каждом i-ом проходе мы попарно сравниваем i-й элемент со всеми элементами массива. Если установлено, что mass[i] > mass[j], то увеличиваем счетчик К на единицу (в начале К = 0). По окончании текущего прохода счетчик К указывает количество элементов, меньших, чем mass[i], поэтому элемент mass[i] занимает в отсортированной последовательности позицию К + 1

1. 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | 3 | 2 | 1 |

### Сортировка слиянием

Базируется на принципе «разделяй и властвуй». Массив делится пополам, и каждая его часть снова делится пополам до тех пор, пока он не будет состоять из 1 элемента (считается упорядоченным). Затем происходит слияние каждой упорядоченной части со своей половиной, сохраняя упорядоченность.

1. Пример сортировки слиянием

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | Массив | | | | | | | Описание |
| 1 | 5 | 9 | 1 | -2 | 0 | 2 | -4 | Ввод массива |
| 2 | 5 | 9 | 1 | -2 | 0 | 2 | -4 | Разбили на две части |
| 3 | 5 | 9 | 1 | -2 | 0 | 2 | -4 | Разбили каждую часть еще на две части |
| 4 | 5 | 9 | 1 | -2 | 0 | 2 | -4 | Остались массивы по одному элементу. Начинаем сливать их |
| 5 | 5 | 9 | -2 | 1 | -4 | 0 | 2 |  |
| 6 | -2 | 1 | 5 | 9 | -4 | 0 | 2 |  |
| 7 | -4 | -2 | 0 | 1 | 2 | 5 | 9 | Окончательное слияние |

### Быстрая сортировка(Хоара)

Метод основан на подходе "разделяй-и-властвуй". Общая схема такова:

1. из массива выбирается некоторый опорный элемент a[i],
2. запускается разделение массива, которое перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо,
3. теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правому
4. для обоих подмассивов, если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

В конце получится полностью отсортированная последовательность.

1. Пример быстрой сортировки

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

## Описание функций

Реализована структура fail, которая хранит данные о файле: его размер и имя (максимум на 32 символа).

**Функция «Compare»**

Описание: сравнивает два элемента в большую или меньшую сторону, в зависимости от параметра k.

Принимает на вход две переменные типа fail (a и b), которые нужно сравнить по их размеру, и целочисленную переменную k (определяет, как сравнивать).

На выход выдает 0 при (k = 1 и a < b ) или (k = 2 и a > b ), выдает 1 при (k = 1 и a > b ) или (k = 2 и a < b ).

**Функция «Bubble\_sort»**

Описание: реализация сортировки пузырьком.

Принимает на вход массив типа fail a, размер массива size (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «Choose»**

Описание: реализация сортировки выбором.

Принимает на вход массив типа fail a, размер массива size (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «Insert\_sort»**

Описание: реализация сортировки вставками.

Принимает на вход массив типа fail a, размер массива size (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «Merge\_sort»**

Описание: реализация сортировки слиянием.

Принимает на вход массив типа fail a, индекс левой границы l (long), индекс правой границы r (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «Merge»**

Описание: реализует слияние двух упорядоченных массивов.

Принимает на вход массив типа fail a, индекс левой границы l (long), индекс середины mid (long), индекс правой границы r (long), порядок слияния k (int) (по убыванию или возрастанию элементов). В результате получается слитый массив a.

**Функция «Quick\_sort»**

Описание: реализация сортировки Хоара.

Принимает на вход массив типа fail a, размер массива size (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «Count\_sort»**

Описание: сортировка подсчетом

Принимает на вход массив типа fail a, размер массива size (long), порядок сортировки k (int). В результате выполнения функции, массив a становится упорядоченным.

**Функция «NumOfFiles»**

Описание:нахождение количества файлов в директории

Входные параметры: const wchar\_t \*sDir – путь к директории

Выходные параметры: количество файлов в директории

**Функция «ListDirectoryContents»**

Описание:формирование массива типа struct fail и заполнение данного массива

Входные параметры: const wchar\_t \*sDir – путь к директории struct fail a[] – массив который необходимо заполнить, int n - длинна массива

Выходные параметры: 0 – если директория найдена, 1 – если директория не найдена

# Заключение

Для определения наиболее быстрой сортировки массивов был проведён эксперимент, в котором измерялось время, за которое каждая сортировка упорядочивает файлы по размеру. Для сортировки использовалось 6 методов, которые может выбрать пользователь. В таблице представлено сравнение времени работы и значения память, необходимые для разных видов сортировок.

1. Сложность методов сортировки

|  |  |  |
| --- | --- | --- |
| **Методы сортировки** | **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) |

# Литература

1. Сайт Algolist. Сортировка выбором – <http://algolist.manual.ru/sort/select_sort.php>.
2. Сайт vscode Сортировка подсчетом - <https://vscode.ru/prog-lessons/sortirovka-metodom-podscheta-na-si.html>
3. Сайт Algolist. Сортировка пузырьком - <http://algolist.manual.ru/sort/bubble_sort.php>.
4. Сайт Algolist. Сортировка вставками - <http://algolist.manual.ru/sort/insert_sort.php>
5. Сайт Algolist. Сортировка слиянием - <http://algolist.manual.ru/sort/merge_sort.php>
6. Сайт Algolist. Сортировка Хоара - <http://algolist.manual.ru/sort/shell_sort.php>